

Rational Build Forge



Information Center

Version 7.1.2.2 This information is under development and can change at any time; the released version will be available on ibm.com.

Note

Before using this information and the product it supports, read the information in "Notices," on page 499.

This edition applies to version milestone 7.1.2.2 of Rational Build Forge and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 2003, 2011.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. About IBM Rational Build

Forge	1
Build Forge product editions	1
Process automation	1
Build Forge concepts and objects	3

Chapter 2. Getting started with Build

Forge	7
Accessing and using the console	7
Filtering and sorting lists	9
Creating a hello world project	10
Setting up a server	10
Defining the project	12
Running the job	13
Viewing the job results	13
Project samples	14

Chapter 3. Notices 15

Chapter 4. Release Notes 17

Release notes - <i>IBM Rational Build Forge version 7.1.2</i>	17
Release notes - <i>IBM Rational Build Forge version 7.1.2.1</i>	22
Release notes - <i>IBM Rational Build Forge version 7.1.2.2</i>	25

Chapter 5. Requirements 29

Management Console requirements	29
Installation Manager requirements	30
Database requirements	30
Agent requirements	31
Agent version interoperability	32
Web client requirements	32
Licensing requirements for product editions	33
Specifying a license file during installation	33
Configuring a Rational license server for Build Forge	33
Obtaining license keys and setting up a Rational licensing server	35
Networking requirements for IPv6 support	36
Integration requirements for other products	37
National language support requirements	39
Language settings for the Management Console and Agent	40
International data support for the database host	41
Changing user language preference in the Management Console	41
Determining the language/charset for UNIX/Linux hosts	41
Determining the language code page for Windows hosts	41

Chapter 6. Planning the installation . . . 43

Components	43
----------------------	----

Types of deployments	44
Installation methods	44
Configuration options	44
Integrations with other products	45

Chapter 7. Pre-installation setup. . . . 47

International data setup	47
Database setup	48
DB2 Express setup	49
DB2 setup	51
Microsoft SQL Server setup	53
MySQL setup	57
Oracle setup	59
Security setup	65

Chapter 8. Installing the Management

Console 67

Starting Installation Manager with launchpad	67
Starting launchpad from the product DVDs	68
Starting launchpad from a downloaded package	68
Installation steps in Installation Manager	69
Post-Installation checklist	73
Increasing the number of file handles for Linux	74

Chapter 9. Alternative installation

methods 75

Installing using your own components	75
Prerequisites	75
International data support	75
Database installation and configuration	76
Apache HTTP Server installation and configuration	77
PHP installation and configuration	79
Apache Tomcat installation and configuration	82
Manually installing Installation Manager	84
Starting Installation Manager	84
Specifying the repository URL	84
Performing a silent installation of product components	85
Creating a response file with Installation Manager	85
Installing and running Installation Manager in silent mode	86
Performing a silent upgrade of product components	87
Creating an update response file with Installation Manager	87
Running the update installation in silent mode	88
Installing the Build Forge system on VMware	89
Installing the Management Console on Linux on System z	89

Chapter 10. Configuring additional features in Management Console . . . 95

Build Forge configuration file (buildforge.conf)	95
--	----

Updating the buildforge.conf file	95
Buildforge.conf reference	96
Configuring the Management Console to use an alternate port.	97
Configuring redundancy	98
About redundancy	98
Installing redundant systems	98
Working with redundancy	99
Enabling IPv6 network support.	99
Modifying httpd.conf.	100
Setting up the FLEXlm client	100
Security features	101
Secure login	101
Implementing single sign-on	102
Enabling SSL and HTTPS	115
Enabling password encryption.	132

Chapter 11. Installing agents 143

Installing the agent on Windows platforms	143
User mode agent	144
Performing a silent agent installation on Windows operating systems	144
Installing the agent on UNIX and Linux systems	145
Installing the agent on System i platforms.	146
Installing and running the agent on System z platforms.	147
Troubleshooting the agent installation on z/OS	148
Running an agent	149
Running an agent on Windows	150
Running an agent on UNIX, Linux, and MacOS	150
Running the agent on System i	150
bfagent reference	152
Configuring the agent	152
Locating the agent configuration file.	152
Changing the agent port.	153
Configuring a different shell	153
Running agent commands on a Network Share File System (Windows)	153
Trigger variables and agent performance	154
bfagent.conf reference	155
Troubleshooting agents	162
Testing host name resolution	162
Testing the connection	163
Troubleshooting an agent on Windows	164
Troubleshooting an agent on UNIX, Linux, or MacOS	164

Chapter 12. Post-installation tasks 167

Starting and stopping the engine	167
Starting and stopping the engine on Windows	167
Starting and stopping the engine on UNIX or Linux systems	167
Setting up users	168
Root user.	168
Creating and editing users	168
Verifying the installation	170
Server authentication.	170
Configuring servers	170
Creating a test project	171
Running projects	171

Troubleshooting common problems	172
---	-----

Chapter 13. Upgrading from a previous version 175

Upgrading a version 7.1 console	175
Performing an update installation	175
Upgrading a version 7.0.2.x console	176
Upgrading Agents.	176

Chapter 14. Uninstalling product components 179

Using Installation Manager to uninstall the product	179
Manually uninstalling the product if Installation Manager install fails	180
Post-uninstallation cleanup of Build Forge files	180
Post-installation removal of the provided DB2 Express database	180
Using Installation Manager to reinstall the product and use an existing DB2 Express	181
Uninstalling the Windows Build Forge Agent.	181
Uninstalling a UNIX or Linux Build Forge agent	182

Chapter 15. Administration 185

About administration.	185
Access groups	185
Access overview	187
Access example: giving a group the ability to run jobs	188
Team and project security plan	188
Managing access properties.	189
Creating and editing users	189
Root user.	192
API users.	192
Permissions	192
Permissions exercise	193
LDAP integration	194
About LDAP integration	194
LDAP domain properties	195
Tasks	197
System configuration settings	200
Messages.	209
Translated messages	210
Subscribing to the RSS data feed for jobs status	210
Filtering the RSS data feed for system messages	211
Security panel	211
Keystore panel	213
SSL panel.	214
SSO panel	215
Managing licenses.	215
Entering a new license key	216
Managing the engine.	216
Pausing the engine	216
Starting and stopping the engine.	216
Managing the database	217
Deleting the database log file	217
Error messages	217
No active steps.	217
License key is invalid or Build Forge license key is corrupt or missing.. . . .	217
A database license is required	218

Chapter 16. Servers. 219

About server resources	219
About the Servers panel.	220
Creating a server	221
Testing a server.	222
Enabling and disabling a server	222
Limiting concurrent jobs on a server.	222
Resetting the job count	223
Resetting the job count to zero on a server	223
Resetting the job count to zero on all servers	223
RSS data feed for server status	224
Server authentication	224
About server authentications	224
Creating server authentications	225
Overriding server authentication	226
Allowing the use of restricted server authentication	226
Selectors	227
About selectors.	227
Selector setup practices	228
Selector variable types	228
Selector variable comparison rules	229
Selector assessment of eligible server resources	230
Collectors	230
About collectors	230
About the Collectors panel	230
Creating collectors.	231
Manifests and dynamic server selection	232
Viewing manifests.	233
Refreshing the manifest manually	234
Setting the update frequency of a server manifest	234
Example setups for static and dynamic server selection	234
Properties reference	235
Using snapshots to create new instances of a selector	240
Selector snapshots overview	240
Selector snapshot planning	242
Creating a selector snapshot from an existing selector or selector snapshot	242
Changing the default selector snapshot.	243
Changing the snapshot name for a selector snapshot	243
Accessing and viewing snapshots in a selector snapshot set.	244
Deleting a selector snapshot	244

Chapter 17. Working with environments 247

About environments	247
Environment inheritance	247
Special cases for inheritance	248
Project variable changes made when starting a job	248
About variables	249
Interpretation of variables in steps	250
Interpretation of undeclared variables	250
About the Environments panel	251
Details tab	251

Snapshot tab	254
Creating an environment	254
Using variables.	254
Creating pulldowns for a variable	254
Including other environments	255
Changing variable values during step execution	255
Mapping Windows drives	256
Using dot commands in variables	257
System variables reference	258
Trigger variables reference	261
Environment snapshots	265
Environment snapshot overview	265
Environment snapshot planning	267
Creating an environment snapshot	267
Changing the default environment snapshot	268
Changing an environment snapshot name.	268
Accessing and viewing snapshots	269
Deleting an environment snapshot	269

Chapter 18. Working with projects 271

About projects	271
About the Projects panel.	271
Changing project properties	274
Copying a project	276
Deleting a project	276
Making steps stick with a server	277
Chains: conditional execution of another project or library.	277
Environment variable inheritance in chained projects	278
Canceling chained projects when wait enabled	278
Defining tags	279
Editing the tag format for a project	279
Synchronizing tags	280
System-defined variables for tags.	281
Creating or editing tag variables	281
Libraries	282
About libraries	282
Copying a library	283
Log filters	283
About log filters	283
Creating a log filter	284
Assigning a log filter to a step.	285
Filter patterns	285
Filter actions	286
Filter notification	288
Error thresholds	288
Error and warning counts	288
Classes	288
About classes	288
Setting up notification	291
About notification templates	292
Configuring your SMTP server	292
Setting notification properties of projects and steps	293
Notification exercise	293
Customizing notification templates	293
Notification for inlined projects	296
Using snapshots to create new instances of a project.	296
Project snapshot overview	296

Project snapshot planning	298
Project snapshot options	298
Verifying and editing access groups for snapshot permissions	299
Creating a project snapshot from an existing project or project snapshot	300
Changing the default project snapshot	301
Changing the snapshot name for a project snapshot	301
Accessing and viewing snapshots in a project snapshot set	302
Starting a job for the default project snapshot	302
Starting a job for a nondefault project snapshot	303
Deleting a project snapshot	303
Chapter 19. Working with steps	305
About steps	305
About the Steps panel	305
Adding a step	309
Editing a step	309
Disabling a step	309
Additional step operations	310
Controlling execution flow	310
How steps run	311
Inlines: including the steps of a project or library	312
Pass Chains and Fail Chains for steps	313
Threading: running steps in parallel	313
Broadcasting a step to multiple servers	314
Conditional step execution	315
While loop execution	316
Condition functions	316
Launching projects from steps	318
Customizing log output	319
Labeling log output for a step	319
Highlighting step output as a color or active link	320
Working with job data	320
Embedding build numbers in project files	320
Changing the build tag during a job	321
Changing environment variable values during a job	321
Using Registers	323
Project registers	324
Copying files to and from server resources in a step	325
Enabling file copying on a server resource	325
Getting a file from a server	325
Putting a file on a server	326
Troubleshooting step processing	326
Job does not process any step commands after an ANT build command	326
Commands in a step after a Windows batch command are not run	327
Dot command reference	327
Dot command syntax	327
.bom	328
.bomexport	329
.break	329
.bset	330
.buildstatus	331

.date	331
.defect	333
.drill	333
.edit	335
.email	335
.export	335
.get	336
.load	336
.lock	339
.mkdir	339
.monitor	339
.pack	340
.pop	340
.poptag	341
.purge	341
.push	341
.put	342
.rem	342
.report	342
.retag	343
.retry	343
.rget	343
.rmdir	343
.rput	344
.run and .runwait	344
.scan	346
.semget	347
.semput	347
.set	347
.sleep	348
.snapshot	348
.source	349
.stop	350
.strsub	350
.test	351
.tset	351
.unlock	352

Chapter 20. Working with jobs 353

About jobs	353
About the Home panel	353
About the Jobs panel	353
Running jobs and viewing results	355
Starting jobs	355
Viewing job results	357
RSS data feed for jobs status	358
Restarting failed jobs	358
Using the Bill of Materials	359
Scheduling jobs	361
Schedule parameters	363
Scheduling purges for classes of job	365
Administering jobs	365
Locking jobs	366
Deleting jobs	366
Working directories for jobs	367
Semaphores	369

Chapter 21. Working with reports 373

About reports	373
-------------------------	-----

Prerequisites for displaying data in report output.	373
Access permissions for displaying data in reports	374
Exporting results for built-in reports to a CSV file	374
Standard reports for Performance	374
Viewing job performance statistics for projects	374
Viewing job duration times for a project	375
Viewing step and server performance by project	375
Predefined reports for Queries.	376
Viewing selector utilization history	376
Viewing the current server manifests by server	377
Viewing job pass/fail/warning results	377
Viewing server utilization for jobs in a date range	378
Searching for a job file using its MD5 value	379
Creating reports with Quick Report	379
Report group permissions for Quick Report	380
Report type reference for Quick Report.	380
Report format and presentation reference for Quick Report	385
Sample reports reference	390
Creating a report using a provided report type	391
Adding report output to the job BOM	392
Modifying and managing reports in Quick Report.	392

Chapter 22. Working with utilities. 395

Accessing and running command-line utilities	395
Exporting projects	395
bfexport reference	395
Using .export	397
Importing projects.	398
Importing projects and other objects using the Import utility	398
bfimport reference.	400
How access groups are assigned to imported objects.	403
Access group assignment and security	403
Renaming and replacing objects on import	403

Chapter 23. Linking to Web resources in the UI Config tab. 407

Chapter 24. Build Catalyst 409

Supported operating systems	409
Installation overview	409
Installing Build Catalyst on Linux and Solaris operating systems	410
Installing Build Catalyst on Windows	411
Build Catalyst examples.	412
rafmake utility reference.	414
Build Catalyst environment variables	417

Chapter 25. Direct integrations with Build Forge 421

Chapter 26. Adaptor integrations 423

Integrating with Rational ClearCase	423
Integrating with Rational ClearQuest	426
Integrating with CVS.	430
Integrating with Perforce	432
Integrating with StarTeam	433
Integrating with Subversion	434
Adaptor Concepts	435
About adaptors.	435
About adaptor links	436
Adaptor templates.	438
Adaptors and projects	438
Adaptors and environment variables	438
Adaptors and notification	439
Adaptors and job execution	439
Adaptors and job results	439
Adaptor requirements	439
Setting up an adaptor	440
Setting up an adaptor: task summary	440
Creating an adaptor from an adaptor template	443
Creating an environment for the adaptor	443
Adding an adaptor to a project	445
Testing the adaptor	446
Setting the adaptor log level	447
Manually starting adaptor-linked projects	447
Quick start for adaptor-linked projects	448
Customizing Adaptor Templates	448
Resetting adaptor templates	449
Modifying adaptor templates	449
Modifying a template for a single adaptor.	449
Creating a new adaptor template.	450
Creating multiple entry point adaptors.	451
Example: enabling email notification	452
Example: removing change details from a BOM report	453
Adaptor reference	454
Dot commands for adaptors	454
Adaptor template structure.	454
Adaptor XML reference	455
adduser	455
bom	456
bomformat	456
command	456
env.	457
execute	457
field	457
integrate	458
interface	458
match	459
notify	459
onproject	459
ontempenv	460
PROJECT_INTERFACE	460
relate	460
resultsblock	460
run.	461
section	462
setenv	462
step	463

Chapter 27. Rational Team Concert integration 465

Installing Rational Team Concert Server Extension	465
Installing the client plug-in for Rational Team Concert	466
Alternate Installation when SSL is enabled	468
Configuring the Rational Team Concert adaptor	469

Chapter 28. Other IDE integrations 471

About IDE integrations	471
Special variables for test projects	471
Plug-ins for Eclipse and Rational Application Developer	472
Installing the plug-ins for Eclipse or Rational Application Developer	473
Alternate Installation when SSL is enabled	474
Using plug-ins for Eclipse and Rational Application Developer	475
Plug-in for Rational Team Concert	476
Using the Rational Team Concert plug-in	476
Troubleshooting the Rational Team Concert plug-in	477

Chapter 29. WebSphere integrations 479

Using WebSphere Application Server instead of Apache Tomcat	479
Using IBM HTTP Server instead of Apache HTTP Server	481
Edit the IBM HTTP Server configuration file	482
Identify Proxy Server in PHP	482
Configuring SSL for IHS	483

Chapter 30. Working with APIs 485

Creating a Build Forge user for API programs	485
Java client API	485
Getting the Java client API package	486
Setting up the the Java client API	486
Perl client API	486
Getting the Perl client API package	487
Setting up the Perl client API	487

Chapter 31. Determining the Management Console version number 489

Chapter 32. Executable commands installed with the product 491

Chapter 33. Glossary 493

access group.	493
adaptor	493
agent	493
archive	493
BOM	493
class	493
clobber	494
collector	494
database	494
dynamic	494
engine.	494
environments	494
handshake	494
interceptor	494
interface	495
job	495
library.	495
Lightweight Directory Access Protocol	495
manifest	495
Management Console.	495
notification templates.	495
plug-in	495
project.	496
selector	496
semaphore	496
server	496
services	497
snapshot	497
static	497
step	497
step log	497
threading.	497
user	497

Appendix. Notices 499

Trademarks	501
------------	-----

Chapter 1. About IBM Rational Build Forge

This section describes product editions and basic concepts.

Build Forge product editions

The following editions of the product are available:

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

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

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

Process automation

Process automation enables businesses to choreograph processes across disparate applications, people, and systems to remove inefficiencies, optimize costs, ensure compliance, and boost productivity.

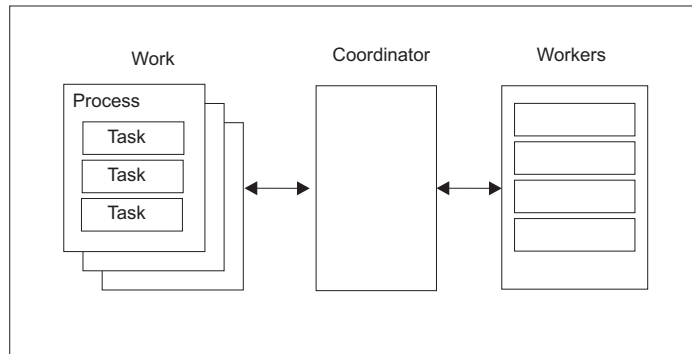
Build Forge automates, orchestrates, manages, and tracks processes within the assembly line of product development. It is commonly used to automate software build and package processes.

This section describes a human model for processes and maps to Build Forge features.

Human model

These roles make up a human-based model of process automation.

- **Work:** the available work to be done is a set of documented *processes*. Each process is made up of *tasks*.
- **Coordinator:** the *coordinator* "runs" the work. The coordinator selects a process to run, and then examines the tasks in order. For each task, the process defines which worker should perform it. The coordinator hands the task to the worker and waits for the result.
- **Workers:** the *worker* performs the task defined in the process.



Build Forge model

The human-based model maps to the Build Forge model as follows:

- **Work:** a work process is defined in a Build Forge *project* object. Each project contains one or more *step* objects. Steps in a project correspond to tasks in a process in the human model.
- **Coordinator:** the coordinator is the running Build Forge software. It contains a component called the process engine. When you start a project in Build Forge, the process engine runs it as a *job* object. A job is simply a running project.
- **Workers:** workers are host computers. They are represented in Build Forge by *server* objects. The host must have an *agent* installed on it.

This is a very simplified model.

- Build Forge contains many other object types to support process automation. Users, authorizations, and other objects that support servers and projects are all stored in the database. Parts of Build Forge are also stored in the database, for example UI widgets. The database must be running for Build Forge to run.
- Projects and steps can be configured to run in many different ways to support complex processes with complex dependencies and workflow. Projects can run other projects, and steps themselves can run projects.
- Steps and projects can dynamically select servers, based on criteria data that is part of the server definition. For example, a step could choose which server to run on based on whether it was running Windows or UNIX/Linux. Servers can also be configured as a pooled resource, so a step would choose a server based on its current availability or load.

Build Forge components to install

To use Build Forge, these components must be installed:

- Database: Build Forge uses the database to store all of the objects that it uses.
- Build Forge: a set of technologies that govern access to the database, run UI code, and perform the work of running jobs.

- Agents: you install agent software on each host that will perform work.
- Web Browser: users typically access Build Forge through a console presented in a web browser.

Build Forge concepts and objects

This section gives you an overview of key concepts and objects and how they are related. Links to more detailed explanations are included for convenience.

Users and roles

To access Build Forge, users need to have a user object set up for them. In production systems this is done by configuring Build Forge to access an identity management system, LDAP.

Build Forge uses an authorization system to control user access to all objects. *Access group* objects contain lists of *permissions*. A *user* belongs to one or more access groups.

You can create your own access groups or modify the ones provided. The access groups defined in the system are:

- Build Engineer - defines processes (creates projects and steps)
- System Manager - administers servers and other system-wide features
- Security - administers users, access, and security
- Developer - helps develop processes, runs jobs, views results
- Operator - copies projects, runs jobs, views results
- Guest - copies projects, runs jobs, views results

Servers

A *server* object defines a place where projects and steps can run. Projects and steps can use the same server or select one independently.

Server objects represent hosts where work is performed. The host must be running a Build Forge agent to receive the work from the system and return results.

Other objects are related to servers and need to be set up before defining projects and steps:

- Server Authentications: a *server authentication* stores login information for the server to use to access the host specified by the server. A server authentication must be created before creating the server that uses it. In the server definition, you choose the server authentication from the list of all server authentications defined in the system.
- Collectors: a *collector* object gathers specified properties of a server. The data is stored in a *manifest*. Servers have a default set of properties assigned. These built-in properties include information about the host architecture, network connections, and resources (CPU, memory, load). You can add other properties by defining collector objects. A collector must be created before adding it to the server. In the server definition, you choose the collector from the list of all collectors defined in the system.

- **Selectors:** a *selector* object defines how a server is selected for use by a project or step. A selector must be created before adding it to the project that uses it. In the project definition, you choose the selector from the list of all selectors defined in the system.
- **Environments:** an *environment* object is a set of variables that can be used by a step. During a job, the variables are set on the server host before the step is run. Environments can be associated with server objects, project objects, and step objects. When the same variable is set to different values in different environments, an inheritance scheme determines which value is used. An environment must be created before adding it to a server, project, or step. In those object definitions, you choose the environment from the list of all environments defined in the system.

Environments

An *environment* is a set of variables. Environments can be specified for server, project, and step objects. When a step runs, environments set at each of those objects are combined to provide variables for the step to use.

Variables can be changed as a step runs. (See “Changing variable values during step execution” on page 255.) The scope of the change can be local to the step, local to the project, or permanent (the variable is changed in the stored environment).

Predefined system variables are available as well as variables you define.

Projects

A *project* defines work to be done in a process. When a project is started, it runs as a job.

The work to be done is contained in the list of steps.

Other objects are related to projects:

- **Selectors:** a *selector* determines where the project will be started. If a selector is not specified for the project, then it cannot run independently and is called a library. The selector must already be defined to assign it to a job.
- **Environments:** an *environment* object is a set of variables that can be used by a step. During a job, the variables are set on the server host before the step is run. Environments can be associated with server objects, project objects, and step objects. An inheritance scheme determines which values are used if the same variable is set to different values. An environment must be created before creating the server, project, or step that uses it. In those object definitions, you choose the environment from the list of all environments defined in the system.
- **Notification templates:** a *notification template* defines how to send out notifications about job activity (job start, job pass, job fail, others). A notification object defines who to notify through access groups. You specify the location of your SMTP server in a system setting.
- **Classes:** a *class* object is used to group projects for maintenance purposes. Typically classes are used to purge or archive completed jobs periodically. All jobs that have run using the project are affected by the class.
- **Adaptors:** an *adaptor* defines an integration with an external system, typically a source code management system. Several sample adaptor templates are provided

as a starting point. You need to configure them further to manage the connection and perform specified actions in the system. See Chapter 26, “Adaptor integrations,” on page 423.

- Adaptor links: an *adaptor link* defines the relationship between an adaptor and the project that uses it.

Steps

A *step* defines the smallest unit of work to be done. (See “About steps” on page 305.) Its key component is its Command property, which includes a command to be run on the selected server.

The Command property can also be used to run dot commands. Dot commands are commands that run on the process engine and provide additional functionality.

Other objects are related to projects:

- Selectors: a step can have its own selector. If not specified it uses the selector of its project.
- Environments: a step can have its own environment. The environments provided by the server, the project, and the step are combined. By default they are applied in that order, so that any variables defined by the step's environment take precedence over definitions of the same variable. Precedence can be controlled in system settings.
- Log Filters: a step can be assigned a log filter in its Result property. A *log filter* object is used to specify conditions that indicate whether the step passes or fails. You use Perl expressions to scan the log for a particular pattern. Normally the exit status of the command is used, but log filters provide an alternate means. A log filter must be created before specifying it in a step. In the Result property for the step, you choose the log filter from the list of all log filters defined in the system.
- Notification templates: a *notification template* defines how to send out notifications about step activity (step start, step pass, step fail, others). A notification object defines who to notify through access groups. You specify the location of your SMTP server in a system setting.
- Build Catalyst: a step can run rafmake, the key utility in Build Catalyst. Build Catalyst provides the means to accelerate make-based builds. Build Catalyst must be installed on the same host where the make builds are run, in addition to a Build Forge agent.

Jobs

A job is a running project. When the job is started, the process engine queues it and then runs it. You can check its status in the Jobs panel. When it completes, the following information is available:

- Results: you can review the results of all steps by opening the job. You can also open a running job to monitor its progress.
- Step log: the step log records extensive information about how a step was run, including information about the manifest and environment settings as well as execution results. Click the step results link in the Results page to view the log.
- Bill of Materials: the Bill of Materials (BOM) contains information about job steps, step manifests, and a job audit log. You can use the .bom dot command to format additional information for the BOM and write data to it. You can use the .scan dot command to add baselines and checkpoints to the BOM.

You can cancel and restart jobs. You can add projects to the Schedule to have them run at scheduled times.

Chapter 2. Getting started with Build Forge®

These topics provide a quick introduction to the system.

Installation

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

1. Check requirements
 - “Database requirements” on page 30
 - “Management Console requirements” on page 29
 - “Agent requirements” on page 31
2. Complete the following steps:
 - a. Chapter 6, “Planning the installation,” on page 43 (stand-alone, normal production, distributed, and so on)
 - b. Chapter 7, “Pre-installation setup,” on page 47 (international data, database, and security)
 - c. Chapter 8, “Installing the Management Console,” on page 67
 - d. Chapter 11, “Installing agents,” on page 143

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

By default you are redirected to another URL to enter your credentials, then redirected back after you successfully log in. The URL is for the services

component. It provides a secure servlet for authentication. Its use is configurable and is not used if it is turned off or if LDAP is used for user authentication. For more information, see “Secure login” on page 101.

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 194.

User sessions

A user session is tracked by a session ID that is initially generated in a cookie. Once generated, the session ID is stored in the database. Every user interaction is checked against the session ID.

In general, you can have multiple browser windows open using the same login, but they must be on the same host and use the same type of browser (FireFox, Internet Explorer).

A session closes under any of the following conditions:

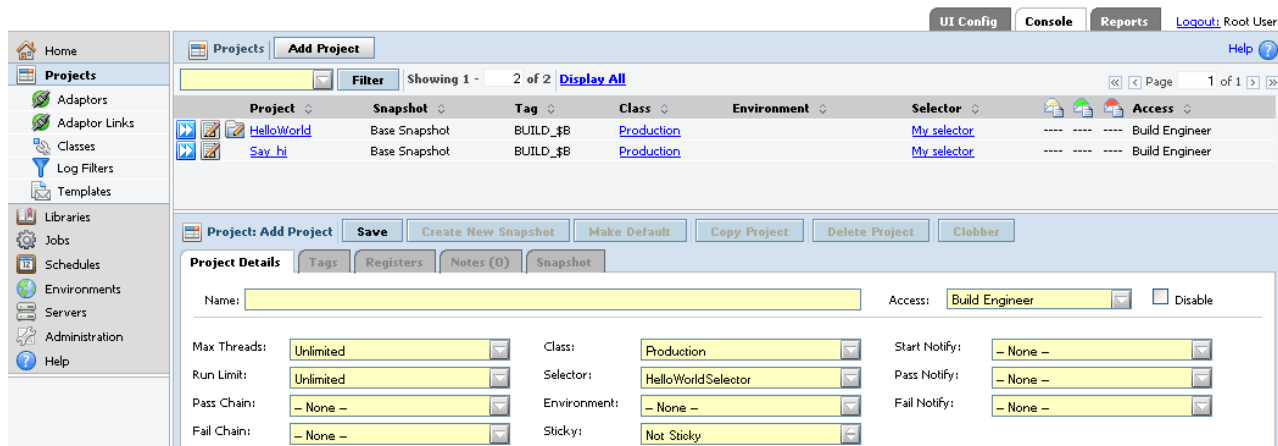
- The same user account is used to log in from another host.
- The same user account is used to log in from the same host but uses a different type of browser.
- The user closes the browser window or tab that contains the Management Console session.
- The session time limit is reached. The time limit is set in the Auto-Logoff system setting. By default it is set to 0, which means there is no time limit.

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.

Using the console window



Click a tab in the upper right to view an application. You can click **UI Config**, **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 right:
- 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 and sorting lists

You can quickly filter or sort lists.

To filter a list, do the following:

1. Enter text in the Filter text box. The entry is case-sensitive.
2. Click **Filter**.
3. View results. You can choose to view all results or paginate them.

Tips:

- Filters work on all columns where the entries are links (blue and underlined). You cannot filter on columns whose values are displayed in plain black text.

- To filter only on a single column, enter the column name before the filtering string. For example, if you have a Selector column, you can enter 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. To delete a filter string, highlight the string and click the **trash can** icon to the right of the string.
- A **Display All** choice always appears in the list of filter strings.
- If you enter filter text and no entry matches the filter, no entries are shown.

To sort a list, do the following:

1. Click on the double arrows next to the column label.
2. View results. You can choose to view all results or paginate them.

Tip:

- Click the double arrows again to reverse the order of the sort.

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.

The following items provide an overview of the tasks needed to create a "Hello World" project.

1. Set up a server:
 - a. Create a server authentication (login name and password) so that Rational Build Forge can access and run commands on a server
 - b. Create the server definition to indicate the server with an agent installed
 - c. Create a selector for the HelloWorld project to use to determine the server on which to run
 - d. Test the server connection to ensure the Master Console and the agent on the server can communicate with each other
2. Define a project:
 - a. Name the project HelloWorld and use the selector to specify the server on which to run the project
 - b. Create a step to run the command that displays the Hello World message
3. Run the job:
 - a. Start an instance of the project, also known as a job
 - b. Click the **Jobs** item in the left panel to check job status
4. View the job results:
 - a. In the completed jobs list, click the job tag to display a list of the job's steps (or single step in the case of the HelloWorld example)
 - b. Click the job's step to display the step log
 - c. Find in the step log the command's Hello World output

Setting up a server

This topic describes how to set up a server to use in a "Hello World" exercise to verify that the build system is set up properly.

Creating a server authentication

Rational Build Forge uses a server authentication (login name and password) to connect to a server with an agent installed.

About this task

You must create a server authentication for a server before you define that server as a resource for Rational Build Forge to use. The privileges associated with the login name and password determine the privileges that the Rational Build Forge projects use when they run on the server.

Procedure

1. In the left panel of Rational Build Forge, click **Servers** → **Server Auth**.
The Management Console displays a blank Server Authentication Details panel at the bottom.
2. Click **Add Server Authentication**.
3. In **Name**, enter a name for the authentication. Use the login name for the authentication name.
4. In **Login**, specify a login name for an account on the server.

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

5. In **Password**, type the password.
6. In **Verified**, retype the password.
7. Click **Save Server Authentication**.

Results

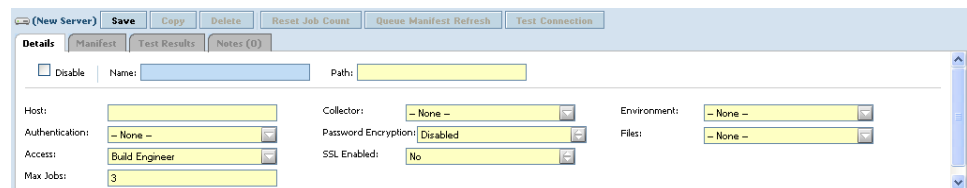
The new server authentication is displayed in the top panel and is available to use when you define a server.

Creating a server

A server in the Management Console represents a host where you can run projects or steps.

Procedure

1. In the left panel of Rational Build Forge, click **Servers**.
2. Click **Add Server**.
3. Provide the server details.



- a. **Name:** Give the server the name HelloWorldServer. You will use this same name later when you define a selector.
- b. **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.

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.

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

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

- d. **Authentication:** Select the server authentication to use with this server.
- e. Leave the other fields at their default values.

4. Click **Save**. Your new server is displayed in the server list at the top of the content panel.
5. To verify that you have correctly configured the server, select your server in the list and then click **Test Connection**.

The system reports errors if it cannot communicate with the server. If you receive an error, ensure the agent is running on the server.

Creating a selector

Selectors determine the server on which to run a project or step.

About this task

Create a selector so that the HelloWorld project can determine where to complete its steps.

Procedure

1. In the left panel of Rational Build Forge, click **Servers** → **Selectors**.
2. Click **Add Selector**.
3. In **Name**, type HelloWorldSelector.
4. Click **Save**. The bottom portion of the panel changes so you can define the selector property that determines which servers to select.
5. Set the server on which to run using the server name specified in "Creating a server" on page 11. With **Name** set to BF_NAME and Operator set to EQ (the defaults), in **Value**, type HelloWorldServer.
6. Click **Save**.

Defining the project

This topic describes how to create a simple project as part of a "Hello World" exercise to verify that the build system is set up properly.

Creating a project Before you begin

You must set up a server and a selector to use in your HelloWorld project definition.

Procedure

1. Select **Projects**. The **Project Details** panel is displayed at the bottom of the main content panel.
2. In the **Name** field, type HelloWorld.

3. Set **Selector** to HelloWorldSelector.
4. Click **Save**. The system displays the empty step list for the project and a blank **Step Details** page. Define the step as explained in “Creating a step.”

Creating a step Before you begin

To define the first step in the HelloWorld project, you must have started to define the project.

Procedure

1. On the **Step Details** page in the **Name** field, type EchoHelloWorld.
2. In the **Command** field, enter a command that writes Hello World 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**. The step is displayed in the step list.

Running the job

This topic describes how to start a job (instance of a running project) and check its status as part of a "Hello World" exercise to verify that the build system is set up properly.

Starting the job Before you begin

You must define the HelloWorld project.

Procedure

1. Select **Projects** to redisplay the project list.
2. Click the **Fast Start** icon  next to the HelloWorld project.

Checking job status Procedure

1. Click **Jobs** and then the **Running** tab to see the HelloWorld job listed as running. After the job completes, it moves to the page for the **Completed** tab.

Note: If the HelloWorld job is not listed, continue to the next step.

2. Click the **Completed** tab. The HelloWorld job is displayed in the list of completed jobs.

Viewing the job results

This topic describes how to view the job results as part of a "Hello World" exercise to verify that the build system is set up properly.

Before you begin

You must define and run the HelloWorld project.

Viewing the completed job list

Procedure

1. Click **Jobs**.
2. Click the **Completed** tab. The completed jobs list displays the HelloWorld job as having completed successfully.

Viewing the step log

Procedure

1. Click **Jobs**.
2. Click the **Completed** tab.
3. Click the job tag for the HelloWorld job. The default job tag for an initial job is BUILD_1. The system displays the names of the job's steps. In this example, the job has only one step, EchoHelloWorld.
4. Click the EchoHelloWorld step to examine its log. 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 that output in the step logs. The log has many sections; the relevant one is the final EXEC section. You can display only the EXEC section by clearing all the check boxes, selecting the EXEC check box, and clicking the Refresh link. The results of the command are shown below.

```
80 04/19/10 11:06AM EXEC Locale set to 'English_United States.1252'
253 04/19/10 11:06AM EXEC Locale set to 'English_United States.1252'
354 04/19/10 11:06AM EXEC Performing variable expansion on command line
356 04/19/10 11:06AM EXEC start [C:\buildforgeprojects\HelloWorld\BUILD_1@mcsystem]
357 04/19/10 11:06AM EXEC Hello World
358 04/19/10 11:06AM EXEC end [C:\buildforgeprojects\HelloWorld\BUILD_1@mcsystem]
```

Results

This project demonstrates that you have configured your system correctly, that projects can successfully access a server, run, and generate output on a server. You can replace the echo command 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:

`<bfinstall>/samples/projects/`

By default, the Build Forge installation directory, or `<bfinstall>`, is C:\Program Files\IBM\Build Forge on Windows and /opt/buildforge on UNIX and Linux.

To use a sample project, do the following:

- Import it into the Management Console, using the `bfimport` command.
- Run the project.

Chapter 3. Notices

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.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department BCF
20 Maguire Road
Lexington, MA 02421
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Chapter 4. Release Notes

This section contains the release notes for each release of the product. It may also contain entries to record where the Information Center was updated between releases. The Release Notes contain the following:

- A list of new and changed features for the release.
- References to requirements, installation instructions, support, and other product information.

Release notes - *IBM Rational Build Forge version 7.1.2*

Build Forge version 7.1.2 is available. Compatibility, installation, and other getting-started issues are addressed.

© Copyright International Business Machines Corporation 2003, 2010. All rights reserved.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

- “Description”
- “New and changed features”
- “APARs fixed in this release” on page 21
- “Announcement” on page 21
- “Compatibility issues with earlier versions” on page 22
- “System requirements” on page 22
- “Installation” on page 22
- “Known problems” on page 22

Description

IBM® Rational® Build Forge® is an adaptive process execution framework that automates, orchestrates, manages, and tracks all the processes between each handoff within the assembly line of software development, creating an automated software factory. IBM Rational Build Forge integrates into your current environment and supports major development languages, scripts, tools, and platforms. With it, you can continue to use your existing investments while adding valuable capabilities around process automation, acceleration, notification, and scheduling.

New and changed features

The following list indicates the new and changed features in this release. For the latest list of features, see the New & Noteworthy page for the GA release on Jazz.net (account required).

- Behavior changes
 - Improved purge performance
 - Improved schedule performance

- Improved manifest refresh and server test performance
- Improved build performance

Logging now takes place in the services layer. For builds with at least several thousand log lines, this change improves build performance.
- Time-zone fixes and improved daylight-saving and standard-time handling
- Users: The list of time zones has been completely changed and a (DST) descriptor has been added to the ones that use daylight savings time
- Schedules: When a schedule is edited, it now displays the time zone of the user that owns the schedule as well as the next run time according to the current user
- Jobs:
 - A job that fails but has Continue On Fail set now has the result Failed But Continued. Previously, the result would have been Warning. A job with a Warning result because of filter matches still produces a Warning result.
 - Jobs that are marked for purging but are not yet being purged are now marked Waiting For Purging. Jobs that are currently being purged are still marked Purging.
- Install:
 - During an Upgrade Install, you are now taken to the Database Configuration page (which is preloaded with the values from buildforge.conf) so that the system can verify that the database is up before starting the Upgrade.
 - When upgrading a version 7.1 console, there are additional tasks to perform to enable performance enhancements in version 7.1.2. For details, see the Upgrading page for the GA release on Jazz.net (account required).
- Using your own application server

Installation no longer prompts you for a location for the rbf-services.war file when you choose to use your own application server. Instead, it always places the rbf-services.war file in the following directory, based on operating system:

C:\Program Files\IBM\Build Forge\PrepForExternal (Windows)
 /opt/buildforge/PrepForExternal (UNIX and Linux)
- Trigger variable - New

Increase agent performance and reduce step log size
 (_SUPPRESS_ENV_OUTPUT)

The trigger variable _SUPPRESS_ENV_OUTPUT can suppress the display in a step log of ENV entries (variables and their values) before a step command is run. By default, _SUPPRESS_ENV_OUTPUT is not set and all variable values in the environment are printed before a step command is run. You can set the variable to the following values:

 - ALWAYS: always omit the ENV messages
 - Any other value: omit the ENV messages. However, if the command fails, the ENV messages are printed after the command message. This information may be useful in debugging the command execution failure.
- System configuration settings - New
 - Control authentication of RSS data feeds (Disable Authentication for XML Feeds)

With the setting Disable Authentication for XML Feeds, you control whether RSS data feeds are authenticated.
 - Improve system performance through the capping of the number of purges (Max Simultaneous Purges)

With the setting Max Simultaneous Purges, you can control how many purges run simultaneously. You can purge as many builds as you like, but no more builds than the value in Max Simultaneous Purges will be simultaneously deleted.

- Control launch of adaptor links attached to a chain project (Run Chain Links)

With the setting Run Chain Links, you control whether a launched chain project also launches any attached adaptor links.

- Control how often servers are checked (Server Test Frequency)

With the Server Test Frequency setting, you control how long (in minutes) the system waits before automatically testing and refreshing the manifest data for servers. The default is 120 minutes (2 hours). A value of 0 means do not check servers.

During these checks, the system contacts servers to verify that:

- The servers are still reachable
- The login information for the servers is correct
- The manifest data for the servers is current

Any server that has not been tested within this time interval is automatically retested as quickly as possible. Manual server test requests initiated from the management console take precedence over these automated tests.

This setting also determines how testing is spread out over time to avoid testing a large number of servers at the same time. The system distributes the testing evenly over the interval. The system will test a server even if that server has already been tested within the current interval. For example, using the default setting of 120 in an installation with 10,000 servers, the system would attempt to test $10,000 / 120 = 83$ (rounded down) servers every minute, even if they have not gone the full 120 minutes without being tested. If the Server Test Frequency setting is changed to 1440 minutes (1 day), then only 6 servers are tested at a time, and the effort is spread throughout the day.

- Control how many times a step attempts to connect to an agent (Step Max Retries)

With the setting Step Max Retries, you control how many times a step attempts to connect to an agent if the first attempt fails. The step fails if it does not connect in the specified number of retries.

- Indicate whether to stop a project if one of its threaded steps fails (Terminate Threads)

With the setting Terminate Threads, you indicate whether, when a threaded step fails, all other active thread blocks in the same project are stopped.

- System configuration settings - Renamed
 - Agent Connect Timeout: Renamed to Server Usage Connect Timeout

- System configuration settings - Removed

The following settings were defunct and were therefore removed from the product.

- Active server refresh interval
- Console hostname
- Default _MAXJOBS
- Expandable Step Log
- GDD Secondary Site Failover
- Inactive server refresh interval
- Lock Manager Address

- Log All Changes
- Manifest check interval
- Quickreport user sub directory for custom datasources
- QuickReport User sub directory for saved reports
- QuickReport User sub directory for uploaded XML files
- Secondary Site Timeout
- Server Self-heal time
- Server Test Connect Timeout
- Site-specific selector weight
- Tab Stop
- Update Time Vars on Restart
- Use System Language for Agent Communication
- Option for installing Rational Automation Framework for WebSphere

Installing Rational Build Forge through Installation Manager now provides an option to include Rational Automation Framework for WebSphere. You must have the Rational Build Forge and Rational Automation Framework for WebSphere files in place for Installation Manager to properly locate and install all the software.

Review the Rational Automation Framework for WebSphere documentation for preinstallation requirements before installing.
- Rational Automation Framework for WebSphere
 - Faster transfer performance

This release provides faster transfer performance for initial transfer of shared components to target systems plus faster performance for any subsequent partial transfer of updated files. It also includes faster transfer of installation media to target systems for installation and patch actions.

Faster transfer performance has been achieved through code improvements and by leveraging functionality provided by IBM® Tivoli® Remote Execution and Access (RXA).

No command modifications are required to take advantage of faster initial transfer of shared components or faster partial transfer of updated files.

Faster transfer of installation media is targeted at users who are employing a framework server (local media) setup or a partially shared setup to manage media. These users can see faster transfer times when running installation and patching actions with the -m option. Note that media transfer improvements do not apply to shared file system users because this media setup does not transfer media to target systems.
 - Serviceability: New logging framework for message consistency.
 - Consumability:
 - Dynamic classpath support for Java and Jython enables users writing custom actions to easily incorporate reusable Java and Jython code in custom actions.
 - Installation Manager now presents Rational Automation Framework for WebSphere as an installation option. The prerequisite to specify the RAFW_ENABLED environment variable is no longer required.
 - New target system support:
 - WebSphere Virtual Enterprise V6.1
 - WebSphere Process Server V6.2
 - WebSphere Enterprise Service Bus V6.2

- WebSphere Services Registry and Repository V6.2
- WebSphere Application Server V6.0 Feature Packs
Web 2.0
- WebSphere Application Server V6.1 Feature Packs
Web 2.0
Enterprise Java Beans 3.0 (EJB 3.0)
Web Services
- WebSphere Application Server V7.0 Feature Packs
Web 2.0
XML
Communication Enabled Applications (CEA)
Service Component Architecture (SCA)
OSGi Applications and Java Persistence API

For additional information about these features, see the Rational Automation Framework for WebSphere Information Center.

- Rational Build Forge documentation now available as an Information Center
The Build Forge Help, Installation Guide, and Release Notes are packaged in a single Information Center.
Engineering the help and documentation as an Information Center provides the following benefits:
 - Single presentation of all topics
 - Improved search capability
 - Content organization compatible with the organization of other Rational products
 - Ability to generate PDFs of selected groups of topics
 - Home page with links to other Information Centers and resources
 The Information Center is installed as a WAR file in the `<bfinstall>/Apache/Tomcat/webapps` directory.
The Information Center will also be available on a public IBM web site.

APARs fixed in this release

For the latest list of APARs fixed in this release, see the Release Notes page for the GA release on Jazz.net (account required).

Announcement

The release announcement for this version is available at www.ibm.com/common/ssi/index.wss. See the announcement for the following information:

- Detailed product description, including a description of new features
- Hardware and software requirements
- Software services and support availability
- Packaging and ordering details
- International compatibility information

Compatibility issues with earlier versions

The database table architecture changed in version 7.1. Build Forge configuration data and historical data need to be migrated from the old database table architecture.

The requirements for upgrading to Version 7.1.2 depend on your current version.

- 7.1.0.x and 7.1.1.x: if you are running any of these versions, you can use the Update feature of Installation Manager to upgrade.
- 7.0.2.x: if you are running any of these versions, you must first upgrade to Version 7.1.1.3, and then upgrade to Version 7.1.2.

System requirements

For information about hardware and software compatibility, see Chapter 5, “Requirements,” on page 29.

Installation

For step-by-step installation instructions, see the topics immediately following Chapter 5, “Requirements,” on page 29.

Known problems

Known problems are documented in the form of individual TechNotes in the Support knowledge base at <http://www.ibm.com/software/rational/support/>. As problems are discovered and resolved, the IBM Support team updates the knowledge base. By searching the knowledge base, you can quickly find workarounds or solutions.

- Reported issues, Rational Build Forge Version 7.1.2.0
<http://www.ibm.com/support/search.wss?rs=3099&tc=SS2MGB&q=RN7.1.2.0&rankprofile=8>
- Reported issues, Rational Build Forge Version 7.1.1
<http://www.ibm.com/support/search.wss?rs=3099&tc=SS2MGB&q=RN7.1.1&rankprofile=8>
- Resolved issues, Rational Build Forge Version 7.1
<http://www.ibm.com/support/search.wss?rs=3099&tc=SS2MGB&q=RN7.1C&rankprofile=8>

At the IBM Rational Build Forge support site you can also run your own queries to request additional types of information:

- Downloads
- Flashes (Alerts)
- News
- Product information and publications
- Troubleshooting (with a broader scope than the URLs provide)

Related information

 [Contacting IBM Support](#)

Release notes - *IBM Rational Build Forge version 7.1.2.1*

Build Forge version 7.1.2.1 is available. Compatibility, installation, and other getting-started issues are addressed.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

- “Description”
- “New and changed features”
- “APARs fixed in this release” on page 24
- “Announcement” on page 24
- “Compatibility issues with earlier versions” on page 24
- “System requirements” on page 24
- “Installation” on page 24
- “Known problems” on page 24

Description

IBM® Rational® Build Forge® is an adaptive process execution framework that automates, orchestrates, manages, and tracks all the processes between each handoff within the assembly line of software development, creating an automated software factory. IBM Rational Build Forge integrates into your current environment and supports major development languages, scripts, tools, and platforms. With it, you can continue to use your existing investments while adding valuable capabilities around process automation, acceleration, notification, and scheduling.

New and changed features

The following list indicates the new and changed features in this release. For the latest list of features, see the New & Noteworthy page for the GA release on Jazz.net (account required).

- Ability to edit the tag number when the Tag Sync property is enabled
This ability had been removed, but the 7.1.2 iFix 1 release restored the feature. (This change announcement is repeated this release to inform a larger audience.)
- Support for Oracle 11g
- JDK 1.6, instead of JDK 1.5, ships with the product
- New system configuration setting: Set how often to check whether a server has become available (Server Wait Time)
Use the Server Wait Time setting to set the number of seconds between checks to determine whether a server has become available. (This setting was available before this release but had not been announced.)
- These issues were fixed in Rational Build Forge 7.1.2 iFix 1. The fixes are also included in this release.
 - A build with inline steps does not retain the BF_RESERVE setting on a server
 - Multiple server definitions for a single host can cause Failed status and jobs that cannot be canceled
 - Fields in Quick Report are missing
 - Reports in Quick Report cannot be edited
 - The management console does not work with the EN_GB locale setting

- Access Group names start with 'Levels' and Build Class names start with 'Classes' when the locale is EN_GB
- The only selector variable you can add to a selector is BF_NAME

APARs fixed in this release

For the latest list of APARs fixed in this release, see the Release Notes page for the GA release on Jazz.net (account required).

Announcement

The release announcement for this version is available at www.ibm.com/common/ssi/index.wss. See the announcement for the following information:

- Detailed product description, including a description of new features
- Hardware and software requirements
- Software services and support availability
- Packaging and ordering details
- International compatibility information

Compatibility issues with earlier versions

The database table architecture changed in version 7.1. Build Forge configuration data and historical data need to be migrated from the old database table architecture.

The requirements for upgrading to Version 7.1.2.1 depend on your current version.

- 7.1.0.x and 7.1.1.x: if you are running any of these versions, you can use the Update feature of Installation Manager to upgrade.
- 7.0.2.x: if you are running any of these versions, you must first upgrade to Version 7.1.1.3, and then upgrade to Version 7.1.2.1.

System requirements

For information about hardware and software compatibility, see Chapter 5, "Requirements," on page 29.

Installation

For step-by-step installation instructions, see the topics immediately following Chapter 5, "Requirements," on page 29.

Known problems

Known problems are documented in the form of individual TechNotes in the Support knowledge base at <http://www.ibm.com/software/rational/support/>. As problems are discovered and resolved, the IBM Support team updates the knowledge base. By searching the knowledge base, you can quickly find workarounds or solutions.

- Reported issues, Rational Build Forge Version 7.1.2.1
<http://www.ibm.com/support/search.wss?rs=3099&tc=SS2MGB&q=RN7.1.2.1&rankprofile=8>
- Reported issues, Rational Build Forge Version 7.1.2.0
<http://www.ibm.com/support/search.wss?rs=3099&tc=SS2MGB&q=RN7.1.2.0&rankprofile=8>
- Reported issues, Rational Build Forge Version 7.1.1

<http://www.ibm.com/support/search.wss?rs=3099&tc=SS2MGB&q=RN7.1.1&rankprofile=8>

- Resolved issues, Rational Build Forge Version 7.1

<http://www.ibm.com/support/search.wss?rs=3099&tc=SS2MGB&q=RN7.1C&rankprofile=8>

At the IBM Rational Build Forge support site you can also run your own queries to request additional types of information:

- Downloads
- Flashes (Alerts)
- News
- Product information and publications
- Troubleshooting (with a broader scope than the URLs provide)

Related information

 [Contacting IBM Support](#)

Release notes - *IBM Rational Build Forge version 7.1.2.2*

Build Forge version 7.1.2.2 is available. Compatibility, installation, and other getting-started issues are addressed.

© Copyright International Business Machines Corporation 2003, 2011. All rights reserved.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

- “Description”
- “New and changed features”
- “APARs fixed in this release” on page 26
- “Announcement” on page 26
- “Compatibility issues with earlier versions” on page 26
- “System requirements” on page 27
- “Installation” on page 27
- “Known problems” on page 27

Description

IBM® Rational® Build Forge® is an adaptive process execution framework that automates, orchestrates, manages, and tracks all the processes between each handoff within the assembly line of software development, creating an automated software factory. IBM Rational Build Forge integrates into your current environment and supports major development languages, scripts, tools, and platforms. With it, you can continue to use your existing investments while adding valuable capabilities around process automation, acceleration, notification, and scheduling.

New and changed features

The following list indicates the new and changed features in this release. For the latest list of features, see the New & Noteworthy page for the GA release on Jazz.net (account required).

- Windows 2008 R2 is now a supported platform for the console.
- SQL Server 2008: SQL Server 2008 is supported in SQL Server 2005 compatibility mode.
- Oracle Instant Client 11.2: Oracle Instant Client 11.2 is supported for use with Oracle 11g.
- Oracle 11g RAC is now a supported database
- Red Hat 6: Red Hat Enterprise Linux 6 Server is supported for the console and agent.
- MySQL 5.1: MySQL 5.1 is supported when used with the MySQL 5.0 client.
- Change in behavior:
 - Before Rational Build Forge 7.1.2.2, when a job with the **On Fail** property set to **Continue** failed, if notification was set, that notification would go to the access group specified in the **Pass Notify** property.
 - Starting with version 7.1.2.2, when a job with the **On Fail** property set to **Continue** fails, if notification is set, that notification goes to the access group specified in the **Fail Notify** property.
- System configuration settings - Removed
The Bump Active Users setting was defunct and was therefore removed from the product.
- Integrating Rational Build Forge and Rational Quality Manager
You can integrate Rational Quality Manager 2.0.1.1 and later with Rational Build Forge 7.1.2 and later. To implement this integration:
 1. Locate the rbf-services-client-war-java.jar file on the computer running your Rational Build Forge management console.
 2. Copy that rbf-services-client-war-java.jar file and use it to overwrite the file of the same name in your Rational Quality Manager installation.

APARs fixed in this release

For the latest list of APARs fixed in this release, see the Release Notes page for the GA release on Jazz.net (account required).

Announcement

The release announcement for this version is available at www.ibm.com/common/ssi/index.wss. See the announcement for the following information:

- Detailed product description, including a description of new features
- Hardware and software requirements
- Software services and support availability
- Packaging and ordering details
- International compatibility information

Compatibility issues with earlier versions

The database table architecture changed in version 7.1. Build Forge configuration data and historical data need to be migrated from the old database table architecture.

The requirements for upgrading to version 7.1.2.2 depend on your current version.

- 7.1.0.x and 7.1.1.x: if you are running any of these versions, you can use the Update feature of Installation Manager to upgrade.

- 7.0.2.x: if you are running any of these versions, you must first upgrade to version 7.1.1.3, and then upgrade to version 7.1.2.2.

System requirements

For information about hardware and software compatibility, see Chapter 5, “Requirements,” on page 29.

Installation

For step-by-step installation instructions, see the topics immediately following Chapter 5, “Requirements,” on page 29.

Known problems


Known problems are documented in the form of individual TechNotes in the Support knowledge base at <http://www.ibm.com/software/rational/support/>. As problems are discovered and resolved, the IBM Support team updates the knowledge base. By searching the knowledge base, you can quickly find workarounds or solutions.

- Reported issues, Rational Build Forge Version 7.1.2.2
<http://www.ibm.com/support/search.wss?rs=3099&tc=SS2MGB&q=RN7.1.2.2&rankprofile=8>
- Reported issues, Rational Build Forge Version 7.1.2.1
<http://www.ibm.com/support/search.wss?rs=3099&tc=SS2MGB&q=RN7.1.2.1&rankprofile=8>
- Reported issues, Rational Build Forge Version 7.1.2.0
<http://www.ibm.com/support/search.wss?rs=3099&tc=SS2MGB&q=RN7.1.2.0&rankprofile=8>
- Reported issues, Rational Build Forge Version 7.1.1
<http://www.ibm.com/support/search.wss?rs=3099&tc=SS2MGB&q=RN7.1.1&rankprofile=8>
- Resolved issues, Rational Build Forge Version 7.1
<http://www.ibm.com/support/search.wss?rs=3099&tc=SS2MGB&q=RN7.1C&rankprofile=8>

At the IBM Rational Build Forge support site you can also run your own queries to request additional types of information:

- Downloads
- Flashes (Alerts)
- News
- Product information and publications
- Troubleshooting (with a broader scope than the URLs provide)

Related information

 [Contacting IBM Support](#)

Chapter 5. Requirements

The Build Forge[®] product components have hardware and software requirements.

Each of the following components has specific requirements:

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

Management Console requirements

This section identifies operating system and licensing requirements.

Operating system requirements

Resource Requirements:

- Disk space: 1 GB of disk space for all product components except the database, which is typically installed on a separate host.
If the provided DB2 Express database is installed on Windows and on the same host as the core product components, an extra 1 GB of disk space is recommended.
- Memory: 1 GB of RAM, if you are using Quick Report, an extra 1 GB of RAM is recommended.

The Management Console runs on the following operating systems and hardware platforms. The Management Console is compiled as a 32-bit application that can be run on 64-bit operating systems; for example, Solaris 64-bit SPARC. Unless specified, all service packs, updates, and minor releases for the indicated version are supported. If a service pack or minor release is specified, later service packs and minor releases are also supported.

Windows operating systems:

- **Windows Server 2008**
- **Windows Server 2008 R2**
- **Windows Server 2003:** Service Pack 1
- **Windows XP Professional:** Service Pack 2

UNIX and Linux Systems:

- **AIX 5.3, 6.1:** Note: due to a chipset incompatibility, Power7 platforms are not supported.
- **Red Hat Enterprise Linux 4 Server:** Intel
- **Red Hat Enterprise Linux 5 Server:** Intel
- **Red Hat Enterprise Linux 6 Server:** Intel
- **SUSE Linux Enterprise Server 10:** Intel
- **Solaris 9, 10:** SPARC

z/Linux:

This platform is supported only with the Enterprise Plus Edition.

- **SUSE Linux Enterprise Service 10:** Service Pack 2 (IBM System z: S390x)
- **Red Hat Enterprise Linux 4.6 Server:** (IBM System z: S390x)
- **Red Hat Enterprise Linux 5 Server:** (IBM System z: S390x)
- **Red Hat Enterprise Linux 6 Server:** (IBM System z: S390x)

Licensing requirements

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

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

- Install Rational License Server V7.1; it supports IPv4 and IPv6 addresses. Obtain version 7.1 from IBM Rational Support or the IBM Rational Download and Licensing Center.
- For information about IPv6 support in the product, see “Networking requirements for IPv6 support” on page 36.

For server licenses, a FLEXlm client is automatically installed and configured. The provided FLEXlm client supports Rational License Server versions 7.0 and 7.1. Only configure a different version of the FLEXlm client licensing software if you must use IPv6 addresses. For details, see “Setting up the FLEXlm client” on page 100.

For more information about licensing, see “Licensing requirements for product editions” on page 33.

Installation Manager requirements

If you have Installation Manager installed, it must meet version requirements to install Build Forge.

The installer attempts to find an installed Installation Manager. It must meet version requirements.

The installer installs Installation Manager if one is not found.

- The minimum supported version is 1.3.0.

Database requirements

The following databases and versions are supported:

- DB2 9.1, 9.5, 9.7 (all editions)
- MySQL 5.0.45 and 5.0.15 a-community-nt
- MySQL 5.1 - requires use of a MySQL 5.0 client
- Microsoft® SQL Server 2005 Service Pack 3 - requires use of the 2005 JDBC driver. Supported **on Windows platforms only**. *Support for SQL Server is for English-only, because it does not support UTF-8.* The database must be created using a case-insensitive collation, such as the default:
SQL_Latin1_General_CP1_CI_AS

- Microsoft® SQL Server 2008, in 2005 compatibility mode only - requires use of the 3.0 JDBC driver. Supported **on Windows platforms only**. *Support for SQL Server is for English-only, because it does not support UTF-8.* The database must be created using a case-insensitive collation, such as the default: SQL_Latin1_General_CP1_CI_AS
- Oracle 10.2.0 - requires Oracle Instant Client 10.2
- Oracle 11g - requires Oracle Instant Client 11.2
- Oracle 11g RAC - requires Oracle Instant Client 11.2

Note: With Oracle, a separate installation of Oracle Instant Client on the IBM Rational Build Forge host is required for PHP oci8 drivers.

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

Databases included with the product

Two databases are available for you to use with the product.

- DB2 Express 9.1.1. This version is packaged with the product. You can choose to have it installed when you install the console. You can run it on Windows only. It is installed on the same host where the console is installed.

Note: If you are installing the console on Windows Server 2008, you must install DB2 Express 9.1.1 (or other database) separately, before installing the console.

- DB2 Workgroup 9.7. This database is available through a separate download from Passport Advantage. Your product license includes a license to this product. If you intend to use DB2 Workgroup 9.7 with the product, you must download it from Passport Advantage and install it before you install the console. There are no platform or host location restrictions for use with Build Forge.

Agent requirements

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

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

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

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

Unless specified, all service packs, updates, and minor releases for the indicated version are supported. If a service pack or minor release is specified, later service packs and minor releases are also supported.

Windows

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

- **Microsoft Windows 7**

- **Microsoft Windows Server 2003**
- **Microsoft Windows Server 2008:** Service Pack 2
- **Microsoft Windows Server 2008 R2**
- **Microsoft Windows XP Professional:** Service Pack 2
- **Microsoft Windows Vista Business, Windows Vista Enterprise, Windows Vista Ultimate**

UNIX, Linux, and z/Linux

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

- **AIX 5.3, 6.1:** System p, Power 5. The agent can run in a System WPAR on AIX.
- **HP-UX 11i (v1, v2, v3):** HP 700, 800: PA-RISC
- **Solaris 9, 10:** SPARC
- **Red Hat Enterprise Linux 4, 5, 6:** Intel, IBM System z: S390x
- **Macintosh OS X 10**
- **Macintosh OS X Server 10**

System i

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

- **IBM i 5.4:** AS/400

System z

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

- **z/OS 1.6 or later:** S390x

Agent version interoperability

Review this compatibility information regarding agents and consoles:

- Agents from version 7.0.2 iFix 2 or later work with consoles from version 7.1 or later.
- Agents from version 7.1 or later do not work with consoles earlier than version 7.1.

For the best performance, maintain the console and agent at the same version.

Web client requirements

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

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

- These browsers are supported:
 - **Internet Explorer version 7 and 8:** Enable ActiveX controls; otherwise, the console server host must be added to the Trusted Sites list. (In the browser, click **Tools > Internet Options > Security**). Note that Windows 2003 disables ActiveX controls by default.
 - **Mozilla Firefox:** version 3.x

Note: To access the management console, cookies must be enabled in your web browser. To enable cookies, follow the instructions provided by your web browser.

- Minimum display resolution:
 - 1024 x 768 minimum to display the Management Console correctly. A resolution of 1280 x 1024 or higher displays the Management Console more clearly.

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

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

Licensing requirements for product editions

The licensing mechanism that you use depends on the product edition that you have. See “Build Forge product editions” on page 1.

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

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

Specifying a license file during installation

This topic applies only to customers using Build Forge Enterprise Plus Edition, which requires file-based licensing.

To specify the license file during installation, complete the following steps:

1. Obtain the license key file, `irbf_license.properties`, from Passport Advantage. Place it in the root installation directory.

Windows	C:\Program Files\IBM\Build Forge
UNIX/Linux	/opt/buildforge

2. In the Installation Manager installer, on the License Server Configuration page, select **License File**.
3. Click **Browse** to locate the license key file in the root installation directory.
4. Double-click the `irbf_license.properties` file to select it and click **Next** to continue with the installation.

Configuring a Rational license server for Build Forge

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

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

- Build Forge Standard Edition
- Build Forge Enterprise Edition

Build Forge configuration requirements for the license server

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

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

Specifying a license server during installation

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

To configure a server-based license:

1. In the Installation Manager installer, on the License Server Configuration page, select **License Server**.
2. (UNIX and Linux) At **Which user should Build Forge run as**, accept the default user (root) or specify a different user. If you do specify a different user, you must still start Build Forge as root, but Build Forge starts the engine and Apache Tomcat as the user specified during this installation step. The user must have read and execute permissions for the database libraries and the JDBC jar files specified in the Database Configuration page in Installation Manager.
3. At **License Server**, provide a valid host name for the Rational license server. If you provide a host name, this information is automatically added to Build Forge system settings.
If you do not know the host name, enter any character or value in this field and update Build Forge system settings in the UI after installation is complete.

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

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

Changing the license server for the Management Console

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

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

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

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

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

port@hostname | @hostname

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

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

Use one of the following formats.

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

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

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

Obtaining license keys and setting up a Rational licensing server

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

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

License administrator task	Resource
Obtains license keys from the Rational License Key Center.	Rational License Management Guide Quick Start Guide for Rational License Key Center at http://www-306.ibm.com/software/rational/support/licensing
Verifies network connectivity for Management Console hosts and the Rational license server.	Rational License Management Guide
Obtains the required version of the Rational License Server software by either: <ul style="list-style-type: none">• Accessing the software included with the product distribution.• Downloading the software from the IBM Rational Download and Licensing Center at: https://www14.software.ibm.com/webapp/iwm/web/reg/signup.do?source=rational	Rational License Management Guide
Installs a Rational license server.	Rational License Management Guide

License administrator task	Resource
<p>Installs or imports license keys to the Rational license server, as follows:</p> <ul style="list-style-type: none"> • On Windows, use the IBM Rational License Key Administrator (LKAD), installed with many IBM Rational products and with the Rational License Server software. • On UNIX/Linux, use the license_setup script and licensing executables from the IBM Rational Download and Licensing Center at: https://www14.software.ibm.com/webapp/iwm/web/reg/signup.do?source=rational 	Rational License Management Guide
<p>Starts the Rational license server and sets up the Rational license server to start up automatically and run as a service, as follows:</p> <ul style="list-style-type: none"> • On Windows, the Rational License Server software is automatically set to start up as a service when the computer starts. If it does not start automatically, see the instructions for automatically starting the license server on Windows. • On UNIX/Linux, create a startup script using the template startup script provided and modify it for your installation. Obtain the template from the IBM Rational Download and Licensing Center at: https://www14.software.ibm.com/webapp/iwm/web/reg/signup.do?source=rational 	Rational License Management Guide

Networking requirements for IPv6 support

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

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

You must manually configure Build Forge for IPv6. To do this, see “Modifying httpd.conf” on page 100.

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

- During installation, administrators specify a host name or IP address that the Management Console uses to communicate with the license server (Standard and Enterprise Editions) and the database.
- In the agent configuration (bfagent.conf file), an optional setting restricts to a particular address or range of addresses connecting to the agent.
- Users enter a URL in a browser to view the Management Console user interface. The URL consists of the host name or IP address of the server where the Management Console is running. For example, to access the Management

Console installed on a server named BFMachine that has IPv4 and IPv6 addresses configured, a user can enter any of the following addresses in the Web browser:

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

The IPv4 and IPv6 addresses differ in format and length.

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

Components that do not support IPv6

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

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

To support IPv6 addresses, you must install Rational License Server 7.1. Obtain it from IBM Rational Support or the IBM Rational Download and Licensing Center. You must also configure the version of the FLEXlm client software that supports IPv6. See “Setting up the FLEXlm client” on page 100.

- DB2 database: the PHP database drivers do not yet support IPv6.
- MySQL database: the PHP database drivers do not yet support IPv6.
- Oracle database other than 11g—the PHP database drivers do not yet support IPv6.

Integration requirements for other products

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

Rational Automation Framework for WebSphere

Starting with Rational Build Forge 7.1.2, you can use IBM Installation Manager to install Rational Automation Framework for WebSphere as part of the Rational Build Forge installation.

For more information, see the Information Center for Rational Automation Framework for WebSphere: <http://publib.boulder.ibm.com/infocenter/rafwhelp/v7r1m2/index.jsp>.

Alternate configuration of required components

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

The following products and technologies are supported:

- Apache Tomcat server 5.5.9
- Apache HTTP Server 2.2.4
- PHP 5.2.4
- WebSphere Application Server 6.1, 7.0
- IBM HTTP Server 6.1

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

Java API requirements

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

Java SDK 1.5 or 1.6 is required for use with the Java API. To use the Java API, you must:

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

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

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

Documentation is provided in JavaDocs.

ClearQuest integration

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

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

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

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

SCM adaptor requirements

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

- ClearQuest: version 7.0 or later
- ClearCase: version 7.0 or later
- CVS 1.1 (CVSv1* adaptor templates)
- CVS 1.2 (CVSv2* adaptor templates)
- Junit
- Microsoft Visual Source Safe 6.x
- Perforce 2009.1
- Rational Team Concert 1.x, 2.x, 3.x. Note that each version has different requirements for integration. For version 3.x there is significant configuration needed with in RTC as well as in Build Forge.
- Star Team 2009
- Subversion 1.3.1 and later

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

IDE plug-in requirements

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

- Eclipse: version 3.02 or later running on Java 5. A plug-in is provided that users can install in their client.
- Rational Application Developer: version 7.0 or later. A plug-in is provided that users can install in their client.
- Rational Team Concert:
Version 1.x and 2.x. Integration requires installation of a server extension. After that installation, users can install a provided client plug-in.
Version 3.0 and later provides a pre-installed Build Forge plug-in that works with Build Forge 7.1.1.3 and later.

Tivoli IT Asset Manager requirements

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

National language support requirements

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

This section provides information about language support in Build Forge.

Language settings for the Management Console and Agent

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

Management Console language settings

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

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

Root user

The installation program creates a root user account that the administrator uses to log in to the console the first time.

On initial log in, the language setting for the root user is based on the operating system language for the Build Forge engine host.

Language default for other users

For administrator-created user accounts, the default console language is initially set using the language preference of the Web browser that the user logs in on. (This is a configuration setting of the Web browser, not Build Forge.)

If the administrator does not change the language preference for the user, it is inherited from the Web browser.

If the administrator wants to change the language for the user, complete this procedure. In the UI, click **Administration > Users > Language**, and then select a language.

LDAP-created user accounts always use the language preference that is configured for the Web browser.

Changing the default user language

After logging in to the console, the administrator can select a different language setting for individual Build Forge users by selecting **Administration > Users > Language** from the left navigation panel of Build Forge window.

If language preferences for users are configured in this way, the Management Console displays interface controls in the language selected for the user regardless of the language configured for the Web browser.

Agent and Build Forge engine language settings

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

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

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

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

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

Language setup

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

International data support for the database host

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

- Use the Unicode UTF-8 character set.
- Install the fonts that you intend to use to display data.

Changing user language preference in the Management Console

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

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

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

Determining the language/charset for UNIX/Linux hosts

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

- To determine the language/charset currently being used by the operating system:
`$ locale`
- To determine the language/charset combinations available to the operating system:
`$ locale -a`
- To set locale on login, use an rc or profile script.

Determining the language code page for Windows hosts

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

- To determine the active code page number, type:
`> chcp`
- To set the code page, enter the number for the language:
`> chcp code_page`

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

Language	Code Page
English	1252
French	
Spanish	
Italian	
German	
Portuguese	
Japanese Shift-JIS	932
Korean	949
Simplified Chinese GBK	936
Traditional Chinese Big5	950

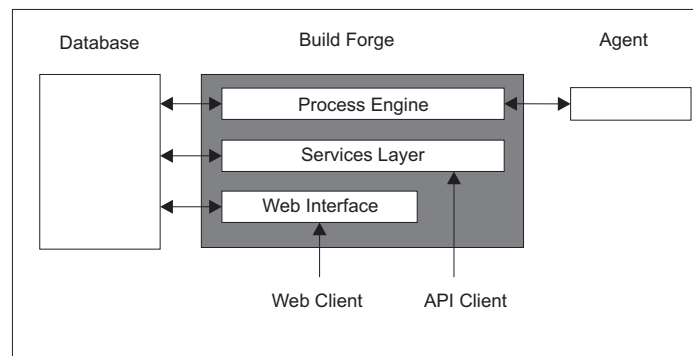
Chapter 6. Planning the installation

Installing Rational® Build Forge® requires you consider several product components.

This topic describes the planning needed for installing the Rational® Build Forge® product components. It serves as a roadmap for the choices you can make about how and what to install and configure.

Components

The Build Forge® system has the following components:



- **Web Client:** used by users and administrators who use Web browsers to access the system. Browser clients access the **Web Interface** component.
- **API Client:** any program using the Java API or Perl API to access Build Forge. API clients access the **Services Layer** directly.
- **Build Forge:** a collective term for the system. During installation the system is shown as made up of **Core Product Features**:
 - **Web Interface:** also referred to as the Management Console or console. This component is made up of a set of PHP modules.
 - **Process Engine:** also referred to as the engine. The engine manages job scheduling and execution.
 - **Services Layer:** a database abstraction layer through which API Clients, the Web Interface, and the Process Engine make requests.

In simple installations all three components are installed on the same host. They can be installed on separate hosts. In that case the Build Forge configuration file `buildforge.conf` must be modified so that the components can communicate.

- **Database:** Information storage for the system. The database stores project definitions, system configurations, and user configurations.
- **Agent:** A program installed on a host. An agent must be installed on every host that you want the Management Console to use as a server resource. The agent receives requests to perform work (steps) and runs them on the host where it is installed.

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

Types of deployments

This section describes the following deployment types:

- Standalone
- Normal production installation
- Tiered installation

Standalone

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

Normal

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

Tiered installation

In a tiered deployment, there are typically three normal deployments:

- Development (also called scratch or sandbox): used by development staff to create new projects. It is also used as the initial area for upgrading.
- Test: used by quality assurance and development staff to test the system.
- Production: used by all to run jobs in the course of daily business.

Build Forge data is exported from one tier and imported to the next.

Important: The three deployments should be symmetric. Use the same operating system, database, and components on each deployment.

Installation methods

This section describes the installation methods available for you to use.

- Installation Manager, interactive: Provide input as the installation occurs
- Installation Manager, silent: Save input to a file and then use that file to provide input during installation
- Installation to use your own components: Continue using your existing components that Build Forge needs
- Installation on virtual images: Read the guidelines for installation on VMware
- Installation on IBM System z: Understand the tools and information needed for installing the console on SUSE Linux on System z

Configuration options

This topic describes features that require you to alter the configuration of the product.

- Port assignments: Learn how to use a port other than the default port 80

- “Secure login” on page 101: you provide a keystore password during installation. The keystore is used both by the secure login mechanism (provided by default) and by SSL, which requires additional setup. You can disable the secure login if you want.
- Password encryption: Encrypt passwords used by the Build Forge engine, agents, services layer, and database
- SSL and HTTPS: Understand the steps required to use SSL throughout the Build Forge system
- Single sign-on: Simplify signing on while maintaining security
- Supporting higher throughput using console redundancy: Set up multiple consoles to use one database
- Accelerating make-based builds using Build Catalyst: Set up single-system parallel builds and multisystem distributed builds

Integrations with other products

- WebSphere: Use WebSphere Application Server rather than Apache Tomcat
- Rational Team Concert IDE: Set up Build Forge as an RTC build server and perform other tasks
- Eclipse and Rational Application Developer IDEs: Learn how to access the console from within Eclipse and Rational Application Developer IDEs
- Using adaptors to access source: Use provided adaptors and design your own to interact with other products

Chapter 7. Pre-installation setup

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

Note: If you already have Build Forge version 7.1 or earlier installed, please see Chapter 13, “Upgrading from a previous version,” on page 175.

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

International data setup

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

Procedure

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

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

```
> chcp
```

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

```
locale
```

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

```
LANG=en_US.UTF-8
LC_CTYPE="en_US.UTF-8"
```
3. Configure databases to use the UTF-8 character set and fonts that support international data.
 - **DB2:**
 - a. Set the codeset and territory. Example: CREATE DATABASE USING CODESET UTF-8 TERRITORY US (or select the appropriate codeset and territory in Control Center).

- b. Set the DB2CODEPAGE environment variable on the management console computer to 1208.

On Windows, use the command:

```
set DB2CODEPAGE=1208
```

On UNIX or Linux, use the command:

```
export DB2CODEPAGE=1208
```

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

- **Microsoft SQL Server** (for use with Management Console on Windows only): *No support for international data.* Microsoft SQL Server uses UCS-2 for storing Unicode data and does not support UTF-8.
 - **MySQL**: Set the server character set and collation. If your installation of MySQL does not currently support international data, you can recompile it from source and use `./configure --with-charset=utf8 --with-collation=utf8_bin`. The Build Forge engine will not start if this support is not configured.
 - **Oracle**: Set the character set to **UTF8 - Unicode 3.0** on the instance when you install it. In the Database Configuration Assistant, the setting is made on the Initialization Parameters step on the Character Sets tab. If you use the command line, set the character set to **AL32UTF8**.
4. (Optional) Configure messages for the Build Catalyst feature.

On Microsoft® Windows® platforms, Build Catalyst messages are displayed in the correct translation automatically.

To view these messages on UNIX and Linux platforms, set the NLSPATH environment variable to the following value:

```
/opt/rational/buildforge/buildcatalyst/lib/nls/%L/utf8/%N.cat:$NLSPATH
```

where:

- %L substitutes the value of the LANG environment variable
- %N substitutes the value of the name parameter passed to catopen(3C)

Alternatively, you can explicitly specify the translation. The following value is for the ja_JP translation:

```
/opt/rational/buildforge/buildcatalyst/lib/nls/ja_JP/utf8/%N.cat:$NLSPATH
```

Database setup

This section contains setup instructions for each supported database.

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

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

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

- Create database objects for Build Forge to use (database, database user).
- Install the necessary client software for Build Forge to use.
- Determine what additional information you will need during installation. Typically this is the location of JDBC drivers that will be used by the Apache Tomcat application server.

- Configure support for the UTF-8 character set and an appropriate collation. Typically, support for international data is specified when you create your database; international data support cannot be configured after database creation. You must install the fonts you intend to use to display data. **Build Forge requires the use of international data (UTF-8 character sets).**

DB2 Express setup

Two types of installation are supported for DB2 Express:

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

DB2 Express information needed when installing the provided database

Before starting the installation, be sure you are logged on as Administrator or a user with administrative privileges.

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

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

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

Database Configuration

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

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

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

DB2 Express Installation Specifics

- **DB2 Express Installation Directory** - Where to install the files. The default is C:\Program Files\IBM\SQLLIB\.
- **Drive to store DB2 Data on:** - The drive on which to create the DB2 directory. Choose a driver from the list of all drivers on your system. The default is C:.

- **Create User Account:** If you select this check box, Installation Manager creates the user and password in Windows. Clear the check box if you specify an existing account.

DB2 Express requirements when connecting to an existing database

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

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

DB2 Express system and database object requirements

The following objects must be defined before running the installer:

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

DB2 Express requirements when connecting to an existing database

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

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

Database Configuration

- **Database Type:** Choose **DB2**.
- **Do you wish to populate this database at install time?:** Choose Yes if you want Build Forge to automatically populate at installation, and No to manually populate it after install.
- **Database Host:** Enter the host name where the DB2 Express installation is located.
- **Database Name:** Enter the name of the database you created for Build Forge to use.
- **Database Schema Name:** Enter the name of the schema you created for Build Forge to use.
- **Database Port:** The default is 50000. Enter the port number you use if it is different.
- **Database Username:** User name for Build Forge to use when accessing the database. Enter the Windows account you created for Build Forge to use.
- **Password:** Password for the Database Username. Enter the password for the user account.
- **Confirm Password:** Enter the password for the user account.

Test the Database Configuration

- **Path to the DB2 Client Libraries:** by default these libraries are in C:/Program Files/IBM/SQLLIB/java on the drive where you installed DB2 Express. Build Forge needs to use these files: db2jcc.jar, db2jcc_license_cu.jar.

- JDBC Driver location: by default these libraries are in C:/Program Files/IBM/SQLLIB/java on the drive where you installed DB2 Express. The JDBC drivers are used by Apache Tomcat to access the database.

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

DB2 setup

Use this procedure to set up support for DB2.

Database objects for DB2

Procedure

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

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

Perform the remaining steps in DB2:

2. Create a user table space with a 16K page size.
3. Create an empty database named **build** with the necessary table spaces.
Database names are case-sensitive.
4. Grant user access to BFUSE_TEMP table space.

Results

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

Sample DB2 SQL command script

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

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

db2 "CREATE BUFFERPOOL 'BUFFP1' IMMEDIATE SIZE 1000 AUTOMATIC PAGESIZE 16384"
db2 "CONNECT RESET"
db2 "CONNECT TO BUILD"
db2 "CREATE SCHEMA schema name"

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

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

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

```
BUFFERPOOL BUFFP1"
```

```
// User must be granted use of table spaces
db2 "GRANT CREATETAB,CONNECT,IMPLICIT_SCHEMA ON DATABASE TO USER BUILD"
db2 "GRANT CREATEIN,DROPIN,ALTERIN ON SCHEMA schema name TO USER BUILD WITH GRANT OPTION"
db2 "GRANT USE OF TABLESPACE BFUSE_TEMP TO USER BUILD WITH GRANT OPTION"
db2 "GRANT USE OF TABLESPACE USERSPACE2 TO USER BUILD WITH GRANT OPTION"
db2 "commit work"
db2 "CONNECT RESET"
db2 "terminate"
```

Tuning parameters recommended for DB2

About this task

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

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

Procedure

1. Set the tuning parameters. Run the following DB2 commands:

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

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

2. Restart DB2. This step is required to put the parameters into effect. Be sure there are no sessions running on the database first.

```
db2stop force
db2start
```

DB2 client drivers

Before you begin

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

Install DB2 Connect Client to provide the drivers.

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

DB2 information needed during installation

About this task

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

Database Configuration

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

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

Test the Database Configuration

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

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

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

The following information is displayed:

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

Microsoft SQL Server setup

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

Before you begin

Note: MS SQL Server is supported only on Windows. Microsoft SQL Server is not supported in Build Forge Version 7.1. Microsoft SQL Server 2005 is supported in versions 7.1.1 and later.

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

Database objects for Microsoft SQL Server

About this task

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

Procedure

1. **Create a user to serve as the database owner.** Build Forge uses this username to log on to the database. Use **build** unless you must use a different name. The user must have full permissions.
 - a. Open SQL Server Management Studio.
 - b. Open the database server in the Object Explorer (left panel).
 - c. Right-click the **Security** folder and choose **New → Login**.
 - d. In the Login - New dialog, specify the login name and choose options as follows. Important: uncheck User must change password at next login.
 - Choose **SQL Server authentication** and provide a password.
 - Uncheck **Enforce password expiration**
 - Uncheck **User must change password at next login**

2. **Create the database.** You must use mixed-mode authentication.
 - a. Open SQL Server Management Studio.
 - b. Open the database server in the Object Explorer (left panel).
 - c. Right-click the **Databases** folder and chose **New Database**.
 - d. In the New Database dialog, specify parameters for the database:
 - Specify a Database name. Use **build** unless you must use another name. The name is case-sensitive. The name of the data and log files are updated automatically in the Database files box.
 - Specify the Database owner.
 - Click the [...] control to the right of the field.
 - In the Select Database Owner dialog, click **Browse**.
 - Check the name of the user you created, then click **OK**.
 - Click **OK** in the Select Database Owner dialog.
 - Specify the Database files parameters. In the Database files table, do the following:
 - For both files: set the Initial Size to 500 (in MB)
 - For both files: set Autogrowth. In the Autogrowth column, click the [...] control to open the dialog. Check the **Enable Autogrowth** box, set growth to 500 MB, and select **Unrestricted Growth**, then click **OK**.

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

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

```

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

```

Note: The READ_COMMITTED_SNAPSHOT attribute must be set to ON. To test for READ_COMMITTED_SNAPSHOT, run the following SQL statement. SELECT is_read_committed_snapshot_on FROM sys.databases WHERE name=<build> Substitute your database name if you did not use build. The statement must return "1".

3. **Set the default database for the user.**
 - a. Open SQL Server Management Studio.
 - b. Open the database server in the Object Explorer (left panel).
 - c. In Object Explorer, open **Security** → **Logins**.
 - d. Right-click the user you created and choose **Properties**.
 - e. On the General page, select a Default database. Select the database you created.
 - f. Click **OK**.

TCP/IP setup on Microsoft SQL Server

About this task

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

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

Procedure

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

Results

Microsoft SQL Server client and JDBC driver installation About this task

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

Procedure

1. Install SQL Native Client. Version 2005.90.4035.00 is required. It is included in Service Pack 3 for Microsoft SQL Server.
2. Install JDBC drivers.
 - For version 2005: JDBC Library version 1.2 is required. Download it from Microsoft at the following location:

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

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

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

- For version 2008: JDBC Library version 3.0 is required:

<http://www.microsoft.com/downloads/en/details.aspx?FamilyID=%20a737000d-68d0-4531-b65d-da0f2a735707&displaylang=en>

ODBC Data Source Setup for Microsoft SQL Server About this task

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

Procedure

1. From the Windows start menu, select **Settings** → **Control Panel** → **Administrative Tools** → **Data Sources**. The **ODBC Data Source Administrator** dialog appears.
2. On the **System DSN** tab, click **Add**. A list of drivers appears.
3. Select **MS SQL Server** from the list of drivers, then click **Finish**. In the **ODBC Setup** dialog box that appears, enter the following information:
 - **Data Source Name:** the name for this data source, *must be the same as Database Name and must not be the same as the schema name associated with the database.*
 - **Description:** description for this data source.
 - **Server Name:** host name of the host where the MS SQL Server database is installed.
 - **Database Name:** database name you created above.

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

Results

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

- **Data source name:** as you assigned when creating the ODBC data source
- **Data source type:** SQL Server

- **User name:** user name you created for the database (for example, **build**)
- **Password:** password for the user name (for example, **build**)

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

Microsoft SQL Server 2005 information needed during installation

About this task

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

Database Configuration

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

Test the Database Configuration

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

Important: Microsoft SQL Server is supported on Windows platforms only.

- **JDBC driver location** - the directory where the JDBC driver is located. This driver is used by Apache Tomcat to access the database.
 - 2005 versions: use the driver provided with the 2005 version.
 - 2008 version: use the version 3.0 driver.

The following information is displayed:

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

MySQL setup

Use this procedure to install and configure support for MySQL.

Before you begin

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

Red Hat Linux 4 Requirements for MySQL

Before you begin

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

Procedure

- Install compatibility shared libraries. You need to install the MySQL-shared-compat package or RPM (but not both).
- Rename startup script. You need to rename the startup script so that it causes MySQL to be loaded earlier in the startup process than it does by default. The following example should move it up enough.

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

Database Objects for MySQL

Procedure

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

Results

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

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

MySQL client drivers

Before you begin

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

For AIX systems only:

The installer attempts to repackage the MySQL client shared libraries. The packaging of the files as they are posted on mysql.com cannot be used by Build Forge. The user running the installer needs to have write access to the directory where the client driver files are installed, typically /opt/mysql. The installer replaces the files libmysqlclient.a and libmysqlclient.so.15. If the installer is not able to repackage the files, it notifies you during installation and continues the installation. In that case, you must repackage the files manually before running Build Forge.

Note: If you are running other applications that use this MySQL client, you may want to repackage the files manually in a separate directory.

To manually package the files do the following.

1. From the directory MySQL was installed in, create a new directory, libbf:
root@myaix:/opt/mysql/> mkdir libbf
2. Copy the lib/libmysqlclient.so.15 file into the new directory:
root@myaix:/opt/mysql/> cp lib/libmysqlclient.so.15 libbf
3. Change into the new directory and build the new archive file:

```
root@myaix:/opt/mysql/> cd libbf
```

```
root@myaix:/opt/mysql/libbf/> ar -q libmysqlclient.a libmysqlclient.so.15
```

This directory can now be used as the client library path for the installation process if it has not already run. If one of the silent or command line methods was used, update the LIBPATH in \$BFROOT/rc/buildforge to use this path.

MySQL Configuration Procedure

Increase maximum database connections to 200. Edit the [mysqld] section in `<mysql-installdir>/my.ini` (Windows) or `/etc/my.cnf` (UNIX/Linux) as follows:

```
max_connections=200
```

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

MySQL information needed during installation About this task

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

Database Configuration

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

Test the Database Configuration

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

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

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

The following information is displayed:

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

Oracle setup

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

Before you begin

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

Red Hat Linux 6 Requirements for Oracle 10

Before you begin

Oracle 10 on Red Hat Linux 6 requires additional setup before you install Rational Build Forge.

Procedure

Install compatibility shared libraries. You must install these packages:

- `compat-libs-5.2-1.i386.rpm`
- `compat-libstdc++-33-3.2.3-68.i686`

Database Objects for Oracle

About this task

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

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

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

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

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

Tuning parameters recommended for Oracle

About this task

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

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

Procedure

1. Set the tuning parameters. Run the following commands:

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

Note: If you get the message "SQL Error: ORA-02095: specified initialization parameter cannot be modified" when you run ALTER SYSTEM SET processes=500 SCOPE=BOTH, use ALTER SYSTEM SET processes=500 SCOPE=SPFile instead.

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

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

Oracle information needed during installation

About this task

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

Database Configuration

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

Test the Database Configuration

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

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

- **ORACLE_HOME Environment Variable** - the directory where Oracle is installed.
- **Path to tnsnames.ora file (TNS_ADMIN)** - the directory containing the tnsnames.ora file. Ensure that full access permissions have been set for the tnsnames.ora file.
- **JDBC driver location** - the directory where the JDBC driver is located. This driver is used by Apache Tomcat to access the database.

The following information is displayed:

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

Oracle Client Configuration

About this task

To install and configure the client for Oracle:

Procedure

1. Install Oracle Instant Client on the Management Console host. You must install the 32-bit version, regardless of whether you are running on a 32-bit or 64-bit platform. Use **Instant Client Package - Basic** for your operating system and Oracle version:
 - Use the version 10.2 client for Oracle 10.2.
 - Use the version 11.2 client for Oracle 11g.

Download the client from Oracle at <http://www.oracle.com/technology/software/tech/oci/instantclient/index.html>.

2. Set up the environment on the Management Console host. Several environment variables must be set.

- LD_LIBRARY_PATH: set to include the client installation directory.

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

- NLS_LANG: set to an appropriate value for international language support.
 - UNIX and Linux systems: the value must include AL32UTF8. Example: AMERICAN_AMERICA.AL32UTF8.
 - Windows systems: the value must include a character map specification that corresponds to the Active Code Page setting in the Windows registry. See “Oracle Client Example Configuration on Windows.”

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

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

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

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

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

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

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

What to do next

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

Oracle Client Example Configuration on Windows: About this task

Example environment:

- Instant Client - Basic at C:\instantclient_11_2, to use American English
- Oracle 11.2 on a Windows system at C:\oracle\product\11.2.0\db_1, installed to support international data

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

- LD_LIBRARY_PATH includes C:\instantclient_11_2\
- NLS_LANG=AMERICAN_AMERICA.WE8MSWIN1252
- ORACLE_HOME=C:\instantclient_11_2\

- ORA_NLS10=C:\oracle\ocommon\nls\admin\data
- PATH includes C:\instantclient_11_2\
- TNS_ADMIN=C:\oracle\product\11.2\db_1\network\admin

About NLS_LANG on Windows Systems

The Oracle client on windows uses a setting in windows to perform local character mapping.

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls\CodePage\ACP

When the client sends data to the server, the characters are sent in the server's character mapping.

The example NLS_LANG setting above corresponds to an ACP setting of 1252, the default for US Windows. For other languages, consult the Oracle InstantClient documentation to get the correct language, locale, and character map parts of NLX_LANG. Be sure to check the actual ACP setting on your Windows system and make the character mapping part of the setting match it.

The character maps for other languages are as follows. Use them in combination with the correct language and locale to create the NLS_LANG setting.

```
1250 EE8MSWIN1250
1251 CL8MSWIN1251
1252 WE8MSWIN1252
1253 EL8MSWIN1253
1254 TR8MSWIN1254
1255 IW8MSWIN1255
1256 AR8MSWIN1256
1257 BLT8MSWIN1257
1258 VN8MSWIN1258
874 TH8TISASCII
932 JA16SJIS
936 ZHS16GBK
949 KO16MSWIN949
950 ZHT16MSWIN950
```

Note: If the character mapping is not set correctly, you do not get a warning and at first there is no obvious difference in behavior or performance. The client attempts to make character conversions in memory. When it runs out of memory and needs to swap to perform the conversions, *performance in communication between the client and server degrades radically.*

Oracle Client Example Configuration on UNIX or Linux: About this task

Example environment:

- Instant Client - Basic at /usr/local/instantclient_11_2, to use American English
- Oracle 11.2 on a UNIX system at /usr/local/oracle/product/11.2.0/db_1, installed to support international data

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

- LD_LIBRARY_PATH includes /usr/local/instantclient_11_2
- NLS_LANG=AMERICAN_AMERICA.AL32UTF8
- ORACLE_HOME=/usr/local/instantclient_11_2
- ORA_NLS10=/usr/local/oracle/ocommon/nls/admin/data

- PATH includes /usr/local/instantclient_11_2
- TNS_ADMIN=/usr/local/oracle/product/11.2/db_1/network/admin

UNIX and Linux Systems using Oracle Instant Client 11.2

The Build Forge application has a dependency on library libclntsh.so.10.1, which is part of the version 10.2 client. During installation, the installer checks for this file in ORACLE_HOME. If the file does not exist, the installer assumes that you are using the version 11.2 client and attempts to create a symbolic link from libclntsh.so.10.1 to the corresponding version 11.2 library, libclntsh.so.11.1.

```
ln -s libclntsh.so.11.1 libclntsh.so.10.1
```

Important: The user running the Build Forge installer must have write access to the ORACLE_HOME directory. If this is an issue at your site, have an authorized user create the link manually before you install Build Forge. At some sites the client may be installed on a file system that is mounted read-only (common practice with Solaris systems).

Oracle information needed during installation

About this task

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

Database Configuration

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

Test the Database Configuration

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

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

- **ORACLE_HOME Environment Variable** - the directory where Oracle is installed.
- **Path to tnsnames.ora file (TNS_ADMIN)** - the directory containing the tnsnames.ora file. Ensure that full access permissions have been set for the tnsnames.ora file.
- **JDBC driver location** - the directory where the JDBC driver is located. This driver is used by Apache Tomcat to access the database.

The following information is displayed:

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

Security setup

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

- Keystore password: you must provide a password for the keystore. It is used both for enabling secure login (credentials encryption) and as a starting point for enabling HTTPS/SSL.
- Certificates: you are given the option of installing a personal certificate or importing a certificate that you already have.
- Secure HTTP: you are asked whether you want to install the Apache server enabled for HTTPS/SSL. The certificate you choose is used. If you need to use a port number other than the default of 443, you need to enter the port number at that time.

Using the provided personal certificate

The provided certificate has the following attributes set:

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

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

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

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

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

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

Using your own certificate

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

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

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

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

Chapter 8. Installing the Management Console

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

1. Perform pre-installation setup, as described in Chapter 7, “Pre-installation setup,” on page 47. Tasks include the following.
 - International data support setup (required)
 - Database setup (required). This step typically involves creating database objects, installing a database client, and collecting information that is needed during the installation steps in Installation Manager.
 - Security setup (optional, depending on your needs)
 - If you are installing the optional Rational Automation Framework for WebSphere along with Build Forge, consult its documentation for preinstallation setup requirements.
2. Start launchpad to run Installation Manager.

Note: Launchpad searches for Installation Manager on the host where you run it. If it does not exist, launchpad runs a packaged Installation Manager to install Installation Manager on your host. It then uses the installed Installation Manager to install Build Forge. If you prefer, you can install Installation Manager manually rather than from Launchpad.

3. Perform the installation steps in Installation Manager.
4. Perform required post-installation checks.

See Chapter 9, “Alternative installation methods,” on page 75 for installation procedures for the following scenarios:

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

Starting Installation Manager with launchpad

Use launchpad to start Installation Manager and install Build Forge.

You can start launchpad in the following ways:

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

Launchpad detects whether Installation Manager is installed on your host:

- If Installation Manager exists, launchpad starts it. In Installation Manager, you select the **Build Forge** package to install.

Important: Your version of Installation Manager must meet the minimum version requirements. See “Installation Manager requirements” on page 30.

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

Launchpad can also install an agent on the local host, if it is a Windows host. You cannot use the launchpad to install the agent on a non-Windows operating system. For agent installation instructions, see Chapter 11, “Installing agents,” on page 143.

Starting launchpad from the product DVDs

Use these instructions to start launchpad from the product DVDs.

Choose the instructions for your operating system.

- UNIX or Linux
 1. Insert the first DVD on the host where you are installing Build Forge.
 2. Mount the drive.
 3. In the root directory of the drive, run `launchpad.sh`.
 4. Select the package to install in Installation Manager.
 - If IBM Installation Manager is found on your host, launchpad starts it. On the first **Install Packages** page, select the **Build Forge** package, and then click **Next**.
 - If IBM Installation Manager is not found, a packaged Installation Manager starts in order to install Installation Manager, and then uses it to install Build Forge. On the first **Install Packages** page, select the **Installation Manager** and **Build Forge** packages, and then click **Next**.
- Windows
 1. Insert the first DVD on the host where you are installing Build Forge.
 2. If autorun is enabled, launchpad starts automatically. If it is not enabled: in the root directory of the drive, run `launchpad.exe`
 3. Select the package to install in Installation Manager.
 - If IBM Installation Manager is found on your host, launchpad starts it. On the first **Install Packages** page, select the **Build Forge** package, and then click **Next**.
 - If IBM Installation Manager is not found, a packaged Installation Manager starts in order to install Installation Manager, and then uses it to install Build Forge. On the first **Install Packages** page, select the **Installation Manager** and **Build Forge** packages, and then click **Next**.

Starting launchpad from a downloaded package

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

1. From IBM Passport Advantage, download the installation package for your operating system to a temporary directory on the host where you are installing Build Forge.
2. Extract the installation image from the downloaded file to a local directory. The contents of the file are extracted to the local directory.

3. Start the launchpad program from the directory where you extracted the files, as follows:
 - Windows: run launchpad.exe.
 - UNIX/Linux: run launchpad.sh.
4. Select the package to install in Installation Manager.
 - If IBM Installation Manager is found on your host, launchpad starts it.
On the first **Install Packages** page, select the **Build Forge** package, and then click **Next**.
 - If IBM Installation Manager is not found, a packaged Installation Manager starts in order to install Installation Manager, and then uses it to install Build Forge.
On the first **Install Packages** page, select the **Installation Manager** and **Build Forge** packages, and then click **Next**.

Installation steps in Installation Manager

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

Before you begin

You must have started Installation Manager and selected the **Build Forge** package to install to follow these instructions. You may also install Installation Manager and the Rational Automation Framework for WebSphere option.

About this task

Follow the prompts to install the desired packages:

Procedure

1. *Install Packages* – Select the **Build Forge** and **Version** check boxes. If Installation Manager is not already installed, select the **Installation Manager** check box. Select the **Rational Automation Framework for WebSphere** check box if you have licensed that option and want to install it along with Build Forge. Click **Next** after you make the selections.
2. *Install Packages: Location - Shared Resources* – Enter or choose the directory where you want shared resources installed, and then click **Next**.
3. *Install Packages: Location - Package Group* – Choose the directory where you want the installation packages installed, and then click **Next**. The default location is: C:\Program Files\IBM\Build Forge.
4. *Install Packages: Features* – By default all three core product modules are installed: Web Interface, Process Engine, and Services Layer. Click **Next**.
5. *Install Packages: License Server Configuration* – Select the "Run as" user and the type of license the console will use. Fill in the following information, then click **Next**.
 - (UNIX and Linux) At **Which user should Build Forge run as?**, accept the default user (root) or specify a different user. This user will start the Build Forge engine and the supplied Apache Tomcat. The user must have read and execute permissions for the database libraries and the JDBC jar files specified in the Database Configuration page in Installation Manager.
 - **Rational License Server Based:** Enter the host name of the Rational License Server.

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

After installation is complete, provided the correct host name. For instructions, see “Configuring a Rational license server for Build Forge” on page 33.

- If a license server is displayed but greyed out, your FLEXlm license client has already registered a license server for the host.

After installation is complete, provide the correct host name. For instructions, see “Changing the license server for the Management Console” on page 34.

- **File Based:** browse to the location where you downloaded the license file.

6. *Database Configuration*

Depending on the OS platform you are installing Build Forge on and the database you want to install, you must specify certain information. Refer to database setup instructions in “Database setup” on page 48 for the following:

- For DB2 Express, see “DB2 Express setup” on page 49.
- For DB2, see “DB2 setup” on page 51.
- For Microsoft SQL Server, see “Microsoft SQL Server setup” on page 53. Note that the JDBC driver to specify depends on the version you choose (SQL Server 2005 or SQL Server 2008).
- For MySQL, see “MySQL setup” on page 57.
- For Oracle, see “Oracle setup” on page 59. Note that you must choose the version that corresponds to the Oracle Instant Client version you installed (Oracle 10 or Oracle 11).

Note: On UNIX and Linux platforms, you must install and use 32-bit database client drivers if you are using an Oracle, DB2, or MySQL database. On the Database Configuration page in Installation Manager, for your specific database type, make sure that you specify the 32-bit version of driver libraries in the **Path to the [DB2|Oracle|MySQL] client libraries** field on the Database Configuration page. Also, for DB2, load the db2profile.

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

a. *Web Server/PHP Configuration*

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

b. *Application Server Configuration*

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

Note: You do this only if you are configuring Build Forge to use one or more components that you have already installed. Normally Build Forge installs these components during installation. See “Installing using your own components” on page 75.

- c. *Security Configuration* Enter and verify a password for the keystore that the installer creates for Build Forge in the following fields.

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

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

The password is required to enable the default secure login (credential encryption). It is also required if you intend to configure Build Forge to use HTTPS/SSL.

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

You have the following options:

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

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

8. *Services Configuration* Fill in the requested information, then click **Next**.

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

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

- **Listen on port:** 3966 is the default. One or both ports must be selected as the services port.
 - **Listen on secure port:** 49150 is the default. One or both ports must be selected as the services port.
9. *Install Packages: Console Start Options* – fill in the requested information, and then click **Next**.
- a.
- **Create a shortcut on the desktop?** The default is create a shortcut on the desktop (for Windows).
- Note:** On Linux, a desktop shortcut is not created.
- b. *Startup Options*
- **Do not start the Console:** The Console starts by default. Click the radio button if you do not want the Console to start automatically.
 - **Start the Console in Service Mode:** The Console starts in service mode by default.
 - **Start the Console in Foreground Mode:** The Console does not start in foreground mode, by default. If you want the Console to start in foreground mode, click the radio button.
10. **IBM Rational Automation Framework for WebSphere Entitlement.** If you chose to install Rational Automation Framework for WebSphere, you are reminded that it is a licensed option of Build Forge and that you should have the licenses before installing. Click **Next** to continue.
11. *Install Packages: Summary review* – Review the summary information on this page to confirm where the Build Forge components will be installed, and then click **Install**.
- a. *Target Location*
- **Package Group Name:** buildforge.console is the default package name.
 - **Installation Directory:** The default installation directory is C:\Program Files\IBM\Build Forge.
 - **Shared Resources Directory:** The default shared resources directory is C:\Program Files\IBM\SDP70Shared.
- b. *Features*
- **Build Forge Features** You can review what features or modules will be installed. For example, the core product modules are: Web Interface, Process Engine, and Services Layer.
- c. *Environment*
- English is the default environment.
- d. *Repository Information*
- **Files will be retrieved from the following locations:** Use this section to review and confirm the repository location.
12. If you elected to install DB2 Express:
- a. *Install DB2 Express.* The installer starts automatically and displays a progress bar.
- b. *Reboot Now* When you are asked when to reboot, reboot now. DB2 Express requires a reboot before you can access it through Build Forge.
13. *Access the Management Console.* Start a browser. Go to the URL for the Management Console:
- General form: http://<hostname>[:<portnumber>]. The port number is optional if you used the HTTP default, port 80.

- Local: if you are running the browser on the same host where the Management Console is running, you can use `http://localhost`.

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

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

14. *Log in.* Use user name **root**, password **root**. Change the root password immediately.

Post-Installation checklist

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

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

Check PATH variable

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

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

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

Identify proxy server

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

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

Add the following entries:

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

Set JVM memory (required for Quick Report)

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

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

```
JAVA_OPTS -Xmx1024M
```

Log message migration

Build Forge 7.1.2 stores job messages in a different manner from older releases. As such, if the Build Forge install is an upgrade from a previous release, the "Message" column in the job output page may be temporarily blank until the migration of that particular job is complete. As this happens in small, low-priority batches, the migration may take time; if any job has blank messages, it is recommended that the user simply wait until that job's message migration is complete.

Increasing the number of file handles for Linux

Before you begin

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

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

To increase the file descriptors:

Procedure

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

Important: Do not leave an empty initscript file on your computer. If you do so, your computer will not start the next time that you turn it on or restart it.

4. On the first line, set ulimit to a number significantly larger than 1024, the default on most Linux computers.

```
ulimit -n 4096
```

Caution: Setting ulimit too high can impact system-wide performance.

5. On the second line, type `eval exec "$@"`.
6. Save and close the shell script.

Results

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

Chapter 9. Alternative installation methods

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

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

Installing using your own components

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

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

- Apache HTTP server and PHP
- Apache Tomcat

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

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

Prerequisites

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

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

International data support

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

Before you begin

- **Web browsers:**

- must have the language set
- must have the fonts you use to display data installed

- **Agents**

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

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

```
locale
```

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

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

- **All databases:**

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

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

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

- **DB2:**

1. Set the codeset and territory. Example: CREATE DATABASE USING CODESET UTF-8 TERRITORY US (or select the appropriate codeset and territory in Control Center).
2. Set the DB2CODEPAGE environment variable on the management console computer to 1208.

On Windows, use the command:

```
set DB2CODEPAGE=1208
```

On UNIX or Linux, use the command:

```
export DB2CODEPAGE=1208
```

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

- **MySQL:** Set the server character set and collation. If your installation of MySQL does not currently support international data, you can recompile it from source and use `./configure --with-charset=utf8 --with-collation=utf8_bin`. The Build Forge engine will not start if this support is not configured.
- **Oracle:** Set the character set to **UTF8 - Unicode 3.0** on the instance when you install it. In the Database Configuration Assistant, the setting is made on the Initialization Parameters step on the Character Sets tab.

Database installation and configuration

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

Before you begin

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

In general you must complete the following tasks:

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

Specific instructions are provided for each database type.

Apache HTTP Server installation and configuration

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

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

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

1. Modify your Apache HTTP Server configuration file (`httpd-vhosts.conf`) to point to the Build Forge application.
2. Install PHP and configure the PHP modules required for the Apache HTTP Server, your Build Forge database, and password encryption if you want to use this security feature.
3. Configure Apache for your database.

Install Build Forge using Installation Manager

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

Prerequisite software

- Apache HTTP Server 2.2.4
- PHP 5.2.4

Edit the Apache server configuration file

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

```
cd <apache-dir>/conf/extras/  
vi httpd-vhosts.conf
```
2. Edit the Apache httpd-vhosts.conf file. To add information about Build Forge to httpd-vhosts.conf, add the following lines:

```
<VirtualHost *:80>  
    ServerAdmin build@yourdomain.com  
    DocumentRoot /opt/buildforge/webroot/public  
    ServerName ausbuild01.yourdomain.com  
    ServerAlias build.yourdomain.com mc.yourdomain.com  
    ErrorLog logs/ausbuild.error_log  
    CustomLog logs/ausbuild.access_log common  
</VirtualHost>
```
3. Modify the DocumentRoot setting to point to the Build Forge web application. In the example, the Build Forge installation directory is /opt/buildforge.
4. Leave the port as 80 or change it to the port you run the Apache HTTP Server on locally.

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

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

5. Modify any other settings in httpd-vhosts.conf as required for your Apache HTTP server:
 - ServerAdmin: email address of the Build Forge administrator
 - DocumentRoot: location of the entry page for the Build Forge application
 - ServerName: server where the Build Forge application is installed
 - ServerAlias: optional aliases for the Build Forge ServerName URL
 - ErrorLog: Apache error log for the Build Forge application
 - CustomLog: Apache error log for logging access to the Build Forge application

Install and configure PHP for the Apache HTTP Server

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

Install and configure PHP for your Build Forge database

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

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

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

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

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

Windows	C:\Program Files\IBM\Build Forge\misc
UNIX/Linux	/opt/buildforge/Platform/misc

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

Configure Apache for your database

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

Apache configuration for DB2

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

```
source /home/db2bf/sqlllib/db2profile
```
2. Add the following lines to `httpd.conf`:

```
PassEnv LD_LIBRARY_PATH
PassEnv CLASSPATH
PassEnv LIBPATH
PassEnv VMSPATH
```

Apache configuration for MySQL

No additional configuration is required.

Apache configuration for Oracle

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

```
PassEnv LD_LIBRARY_PATH
PassEnv NLS_LANG
PassEnv ORACLE_HOME
PassEnv ORA_NLS
PassEnv ORA_NLS32
PassEnv TNS_ADMIN
```
2. Add the following lines to the script that starts Apache at boot time (commonly `/etc/init.d/httpd` or `/etc/init.d/apache2`) and provides values for the following settings.

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

Start IBM HTTP Server

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

PHP installation and configuration

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

Before you begin

Requirements:

- Version: PHP must be 5.2.4 or later

- Database drivers: PHP modules for the Build Forge database installed

About this task

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

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

Install PHP

About this task

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

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

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

Procedure

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

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

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

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

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

2. Compile PHP.

```
$ make
```

This step compiles executables in your local directory.

3. Install PHP (do as root).

```
# make install
```

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

Configure PHP Procedure

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

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

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

- DB2: `ibm_db2.so`
 - MySQL: `mysql.so` and `mysql_i.so`
 - Oracle: `oci8.so`
2. Edit the PHP configuration file `php.ini`. The following example assumes that PHP is installed in `/usr/local/php-5.2.4`:

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

Add the following entries:

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

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

- DB2: `extension=ibm_db2.so`
- MySQL: two entries -
`extension=mysql.so`
`extension=mysql_i.so`
- Oracle: `extension=oci8.so`

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

Edit Apache Configuration File Procedure

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

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

Add the following lines:

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

Identify Proxy Server

About this task

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

Procedure

Edit the PHP configuration file `php.ini`. It is located in `<php-install>/lib`, for example `/usr/local/php-5.2.4`.

Add the following entries:

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

Apache Tomcat installation and configuration

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

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

Software prerequisites

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

Install the jar file for the JDBC driver

Download and unzip the JDBC driver for your database.

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

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

- DB2 - <http://www-306.ibm.com/software/data/db2/express/download.html>
Click the download link next to **DB2 Driver for JDBC and SQLJ**; IBM account registration is required. You also need to locate and install a licensing .jar, `db2jcc_license_cu.jar`.

- MySQL - <http://www.mysql.com/products/connector/j/>
Click the link for **MySQL Connector/J 5.0 or 5.1**. Select the JDBC driver version that corresponds to your MySQL version.
- Oracle - http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html
Click the download link next to your version of Oracle; account registration is required.
- Microsoft SQL Server- <http://msdn.microsoft.com/en-us/data/aa937724.aspx>
Click the **Download SQL Server JDBC Driver** link.

Configuring your Apache Tomcat server in Installation Manager

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

1. Shut down Apache Tomcat.

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

2. Start Installation Manager.
3. On the Start page, click **Install**.
4. Follow the instructions in the Installation Manager wizard to install the product.
5. On the Application Server Configuration page, click **Yes** to configure your own application server.

Check list: Application server configuration

✓	Field	Description
	Redirection URL	Enter the host name and port number of your application server. You must specify rbf-services as the context path. For example: <code>http: https://<app_server_host>:<app_server_port>/rbf-services</code> .
	Specify the directory where you want to install the BF Services Plug-ins	Specify a directory local to the application server host. Installation Manager installs the Build Forge services layer application plug-in extensions in this directory. The user who is running the application server must have read, write, and execute permission to this directory.

Note: In previous versions of Build Forge, it was necessary to specify the WAR deployment directory. In Build Forge 7.1.2, the WAR deployment directory is automatically set to *bfinstall/PrepForExternal*.

6. Complete installation through Installation Manager.

Post-installation configuration of Apache Tomcat

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

1. You must manually add the buildforge.conf file to the rbf-services.war file.
The location of the rbf-services.war is dependant upon your operating system. UNIX and Linux place this file in `/opt/buildforge/PrepForExternal`; Windows

places this file in C:\Program Files\IBM\Build Forge\PrepForExternal. See “Updating the buildforge.conf file” on page 95 and complete the relevant steps.

2. Increase JVM heap size for the Apache Tomcat server.
Set the JVM maximum heap size option -Xmx to 1024 M.
Use the CATALINA_OPTS or the JAVA_OPTS environment variable in catalina.bat or catalina.sh to set this JVM option.
3. Before you start Build Forge, start Apache Tomcat:
`$CATALINA_HOME/bin/catalina.sh start.`

Manually installing Installation Manager

IBM Installation Manager is installed automatically or updated if you use launchpad program to start the product installation. See “Starting Installation Manager with launchpad” on page 67.

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

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

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

Starting Installation Manager

Start Installation Manager on Windows or on UNIX/Linux.

Before you begin

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

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

Specifying the repository URL

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

Before you begin

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

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

To specify a repository, complete the following steps:

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

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

Note: To search for updated packages, make sure **Search service repositories during installation and updates** is selected. This option is selected by default.

6. Click **OK** to close the Preferences window.

Performing a silent installation of product components

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

The following tasks are required for silent installation:

1. Install Installation Manager.
2. Create the response file.
3. Run Installation Manager in silent installation mode.

Note: You cannot use silent installation in the following cases:

- You are installing on a Linux server that does not have X11 installed
- You are installing Rational Automation Framework for WebSphere

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

Creating a response file with Installation Manager

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

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

To create a response file for installation:

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

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

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

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

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

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

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

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

3. Follow the instructions in the Install Packages wizard to make your installation choices.
4. Click **Finish**, and then close Installation Manager.

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

Installing and running Installation Manager in silent mode

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

The following tasks are required for silent installation:

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

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

When the Installation Manager runs in silent installation mode, it reads the response file and writes a log file to the directory that you specified. A response

file is required; log files are optional. The result of this execution should be a status of 0 for success and a non-zero number for failure.

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

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

Performing a silent upgrade of product components

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

The following prerequisites are required for silent upgrade:

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

The following tasks are required for silent installation:

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

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

Creating an update response file with Installation Manager

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

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

To create a response file for an Update installation:

1. Run Installation Manager. In Preferences, add the URL of the product repository for the update installation to the list of IM repositories and be sure it is selected.
2. Exit Installation Manager.
3. At a command line, change to the eclipse subdirectory in the directory where you installed Installation Manager. For example:

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

4. Start recording the installation without actually installing the product.
Enter a full path, including file name, for *response_file* and *log_file*. Ensure that the file paths that you enter exist. Installation Manager does not create directories for the response file and the log file. The *agentDataLocation* must be a writable directory.
`IBMIM -record response_file -log log_file -skipInstall agentDataLocation`
5. Installation Manager starts. In Installation Manager, click **Update** and then respond to the prompts.
6. When Installation Manager finishes, click **Finish**.
7. Exit Installation Manager.

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

Running the update installation in silent mode

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

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

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

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

- Windows example

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

- UNIX or Linux example

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

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

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

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

Installing the Build Forge system on VMWare

Build Forge can be installed and run on VMWare.

Use these guidelines:

- Install the database that Build Forge uses on a separate host, preferably a physical host rather than a VMWare image.
- Set the memory that VMWare Workstation uses to run your virtual machine uses to at least 1GB. In VMWare Workstation, click **Edit** → **Preferences** → **Memory** to adjust this value.
- You may need to address other system resource parameters to maximize performance on VMWare.

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

Installing the Management Console on Linux on System z

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

The console for z/Linux is packaged with IBM HTTP Server, rather than Apache web server.

After you install the Management Console, install the agent rpm package (`zlinux-bfagent-<version>.rpm`) on z/Linux to set up a z/Linux server for Build Forge. For installation instructions, see “Installing the agent on UNIX and Linux systems” on page 145.

Information needed during installation

During installation you are asked for the following information.

1. Installation directory
 - Provide an absolute path to the location where you want to install Build Forge.
2. Database information
 - Database type that Build Forge will use (DB2, Oracle, or MySQL)
 - Database server host name
 - Database port number
 - Database name to use
 - Database user name for Build Forge to use to connect to the database
 - Password for the database user name
 - Location of the client libraries used to access the database
 - Location of the JDBC driver jar file
3. Application server information
 - Application server to use (the provided Tomcat or a WebSphere Application Server installation that you have set up)

If you choose WebSphere Application Server, you provide more information:

- Location of the Build Forge services component as it will be installed on WAS. A domain, port, and path to rbf-services are required. Example:
`http://mydomain.com:9080/rbf-services`
 - Directory to use for plug-ins, a readable and writeable directory. Currently it is used only when integrating with Rational Team Concert.
 - WAR deployment directory: temporary location for the Build Forge services WAR. You deploy it to the application server after installation.
 - Path to the Java executable (java.jar)
 - Temporary storage directory for the services layer. The directory must be readable and writeable. It is used by the services component to store temporary information.
 - HTTP port for the Build Forge services (default 3966)
 - SSL port for the Build Forge services (default 49150)
4. Web server information
- Web server to use (the provided IBM HTTP Server or a web server that you have set up)
- If you choose the provided IHS server, you provide more information about SSL, including whether to use SSL, ports and memory to use and whether to use an existing secure certificate or create one.

Running the installer

1. Go to the directory where you extracted the package for the console from the tar file.
2. Run the following command:
`./cmdline-install.sh`

Example

The following annotated listing shows how the installer steps proceed. This is an annotated run and does not reflect an actual installation. Where defaults are available, they are shown in brackets, for example [y]. Press Enter to accept a default.

```
Install directory [/opt/buildforge]
```

```
What database will you be using?
```

```
Enter the # of the database you will use
```

- ```
1) DB2
2) Oracle
3) MySQL
```

Note: The rest of the listing assumes that Oracle was chosen.

```
What is your database hostname? 127.0.0.1
```

```
What is your database port number? [1521]
```

```
What is your database name? build
```

```
What is your database user name? build
```

```
What is your database user password?
```

```
Confirm your database user password?
```

```
Would you like this installation to create the Build Forge database schema? (y|n) [y]
```

Specify client libraries and information at this point. Depending on your choice of database, you are prompted as follows. Use absolute paths.

- DB2

Where are your 32-bit DB2 client libraries (libdb2.so.1)?

Note: the libraries must be 32-bit libraries. Enter an absolute path.

Where is your DB2 (DB2\_HOME) installed?

- MySQL

Where are your MySQL client libraries (libmysqlclient.so)?

Note: the libraries must be 32-bit libraries. Enter an absolute path.

- Oracle

Where are your Oracle client libraries (libclntsh.so)?

Note: the libraries must be 64-bit libraries. Enter an absolute path.

Where is your Oracle instant client (ORACLE\_HOME) installed?

Note: this is the absolute path to the root of the instant client libraries.

Where is your tns.names file located (TNS\_ADMIN)?

Note: this is the directory that contains the tns.names file. Enter an absolute path.

Please enter the directory with your database JDBC jar file?

Note: Enter the absolute path to ojdbc14.jar.

Will you be using the supplied Tomcat app server? (y|n) n

Note: The rest of the listing assumes you are using WAS as your application server.

Enter the full URL used to contact the services layer on your application server:

`http://mydomain.com:9080/rbf-services`

Enter the directory to install the Build Forge Services plugins to:

Note: This should be readable and writable directory on the application server host. The services component uses it when Rational Team Concert is integrated with Build Forge.

Specify the war deployment directory:

Note: Specify a directory on the local host. The installer places the services .war file here when installation is complete. You then deploy it into your application server.

Enter the path to a jar executable (which should be included in any JDK):

Note: enter the path to the .jar file for your database driver.

Enter the temporary storage path for the Services Layer:

Note: Enter the path to a directory that the services component can use. It must be readable and writeable.

What http port will the Build Forge services layer use? [3966]

What ssl port will the Build Forge services layer use? [49150]

Will you be using the supplied Apache web server? (y|n)[y]

Note: The rest of the listing assumes you entered 'y'

Would you like Apache to use SSL? (y|n)[n] Enter 'y' to configure Apache for SSL

Note: The rest of the listing assumes you entered 'y'.

What ssl port will Apache use? [443]  
Please enter a memory limit for PHP (in MB): [256]

Would you like to modify or specify a custom SSL certificate? (y|n) [n]

Note: The rest of the listing assumes you entered 'y' and intend to create a custom certificate.

Do you have an existing secure certificate? (y|n)[n]

A validity period is required for this cert please enter in [number][period] format

Examples: 10Y = 10 years, 6M = 6 months, 350D = 350 daysEnter the validity period for this cert:

Enter the common name for the certificate (usually the name of the server) [linux142.rtp.raleigh.ibm.com]:

Please enter your Locality/City:

Please enter your State/Province:

Please enter your Organization Name:

Please enter your Organization Unit:

Please enter your Country from the list below:

France  
Taiwan  
Italy  
Germany  
Korea  
United States  
China  
Brazil  
Spain  
Japan

Please enter your Street Address:

A keystore password is required, and must be at least 6 characters long

Please enter a keystore password

If you use IBM HTTP Server rather than Apache as a web server, for additional information about setting it up and enabling SSL, see "Using IBM HTTP Server instead of Apache HTTP Server" on page 481.

## Starting the console

1. Start the Management Console:

`<bfinstall>/rc/buildforge start`

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

`<bfinstall>/server/tomcat/logs/catalina.out`

3. Start a Web browser and enter the fully qualified z/Linux host name. For example: `http://myhost.mycompany.com`.

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

## Installing the license file

The license file for z/Linux is located in the `<bfinstall>` directory. The license file name is `IRBF_license`. After installation it contains a text message instructing you to download your actual license file from Passport Advantage.

After you have downloaded the license file and placed it in the `<bfinstall>` directory, configure Build Forge to use it:

1. Start the Management Console.
2. Log in as **root/root**.



3. Select **Administration > System**.
4. Locate the License Server setting and set its value to fully qualified path to the license file.  
For example: `<bfinstall>/IRBF_license`.

## Enabling SSL for the Management Console

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

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

The installation program does some of the work needed to enable SSL, if you answered yes when prompted. Complete the tasks described in this section to enable SSL:

1. Review Personal Certificates and Keystores
2. Configure IBM HTTP Server for SSL
3. Enable SSL in the Management Console UI
4. Enable debugging for SSL

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

### Review Personal Certificates and Keystores

The following keystores are created by the installation program:

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

See “Managing certificates” on page 124 for more information about converting an existing PEM certificate and managing certificates.

### Configure IBM HTTP Server for SSL

See "Configuring SSL for IHS" in “Using IBM HTTP Server instead of Apache HTTP Server” on page 481. You must change keystore formats and add entries to httpd.conf.

### Enable SSL in the Management Console UI

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

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

### Enable Debugging for SSL

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

1. Enable debugging in the engine. Set the following environment variable before you start the Build Forge engine:  
`export BFDEBUG_SECURITY=1`
  - a. Restart the Build Forge engine.
  - b. Restart IHS. Restarting IHS enables PHP to use this debug parameter.
2. Enable debugging in Tomcat. Make the following changes in `<bfinstall>/server/tomcat/common/classes/logging.properties`:
  - a. Add the following line:  
`com.buildforge.level = ALL`
  - b. In the handlers section, change all other levels from FINE to ALL.Restart Tomcat to make the changes take effect.

---

## Chapter 10. Configuring additional features in Management Console

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

---

### Build Forge configuration file (buildforge.conf)

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

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

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

|            |                                  |
|------------|----------------------------------|
| Windows    | C:\Program Files\IBM\Build Forge |
| UNIX/Linux | /opt/buildforge/Platform         |

### Updating the buildforge.conf file

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

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

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

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

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

**Note:** You need root or Administrator privileges to edit this file.

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

|                      |                                                              |
|----------------------|--------------------------------------------------------------|
| Apache Tomcat server | <bfinstall>/Apache/tomcat/webapps<br>\$CATALINA_HOME/webapps |
|----------------------|--------------------------------------------------------------|

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

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

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

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

7. On Windows, complete the following commands:

- Re-create the classes directory, for example:  
mkdir WEB-INF\classes
- Copy the updated buildforge.conf file into the new classes directory, for example:  
copy <path\_to\_bf\_conf>\buildforge.conf WEB-INF\classes
- Update the rbf-services.war file, as follows:  
jar -uvf rbf-services.war WEB-INF

8. On UNIX and Linux, complete the following commands:

- Use the unzip utility to create and populate the rbf-services directory:  
unzip -d rbf-services rbf-services.war
- Copy the updated buildforge.conf file into the new classes directory, for example:  
cp <path\_to\_bf\_conf>/buildforge.conf rbf-services/WEB-INF/classes

9. Restart the Build Forge engine.

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

## Buildforge.conf reference

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

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

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

| Keyword         | Value                                                                                                                                                                            |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bf_file_storage | Directory where temporary Build Forge files reside. Example:<br>C:\Program Files\IBM\BuildForge\temp                                                                             |
| bf_plugin_dir   | Directory where IDE plug-ins connecting to a Management Console reside.                                                                                                          |
| birt_home       | File location of Eclipse reporting tool (BIRT).                                                                                                                                  |
| db_database     | The name of the database that you created for console use.                                                                                                                       |
| db_hostname     | The host name/IP address of the computer running the database. When entering a value for db_hostname, use the actual name or an IP address. Do not use the default of localhost. |
| db_password     | The password that you created for the database user name.                                                                                                                        |

| Keyword           | Value                                                                                                                                                                                                                                                                                                                       |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| db_provider       | The database you elected to install Build Forge to. Do not edit this value.                                                                                                                                                                                                                                                 |
| db_schema         | The name of the schema of your database (usually the same as db_username, but you may have chosen another schema name).                                                                                                                                                                                                     |
| db_tcp_port       | The database connection port you are using.                                                                                                                                                                                                                                                                                 |
| db_type           | The type of database Build Forge was installed to. Default is odbc. Do not edit this value.                                                                                                                                                                                                                                 |
| db_username       | <p>The user name of the database. This is set up before running the installer.</p> <p>For DB2 and DB2 Express, create a user in Windows, not in DB2.</p> <p>For all other database types, you create a database user name.</p> <p>See “Database setup” on page 48.</p> <p>This value is required for all console types.</p> |
| services_hostname | The host name/IP address of the computer running the Build Forge services layer. It is the fully qualified domain name (FQDN) in your services configuration.                                                                                                                                                               |
| services_ssl_port | The SSL port for connecting to Build Forge services securely.                                                                                                                                                                                                                                                               |
| services_tcp_port | The TCP port for connecting to Build Forge services when SSL is not specified.                                                                                                                                                                                                                                              |
| services_url      | URL that specifies the port used for the services layer. Example:<br>services_url http://mybfhost.com:8080                                                                                                                                                                                                                  |

---

## Configuring the Management Console to use an alternate port

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

### About this task

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

- During installation, set the port to the value that you want.
- If the Management Console is already installed, complete these steps:
  1. Change two settings in httpd.conf (located at *<bfinstall>/Apache/conf/* for Windows installs, and *<bfinstall>/server/apache/conf/* for \*nix installs). For example, if myHost is your local computer and you want to use port 81, specify these settings:
 

```
Listen 81
ServerName myHost:81
```
  2. Start the console and log in as root or a user name with administrative privileges.
  3. Select **Administration** → **System**, and then change the "Console URL" system configuration setting to the URL and port that the Management Console runs on. For example: http://myHost:81 (Do not use a trailing slash).
  4. Stop and then restart the engine.
    - Windows: Click **Start** → **Programs** → **IBM Rational Build Forge Management Console** → **Stop Engine Service** and then click **Start Engine Service**.

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

- UNIX or Linux: Use the script provided for the rc file.

```
$ /opt/buildforge/rc/buildforge start
```

```
$ /opt/buildforge/rc/buildforge stop
```

You may also use manual commands.

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

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

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

```
<bfinstall>/Platform/buildforge
```

---

## Configuring redundancy

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

### About redundancy

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

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

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

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

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

## Installing redundant systems

### About this task

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

### Procedure

1. Perform pre-installation setup on the host according to instructions for your database. Depending on your database, the setup may require installing a database client on the host and performing other configuration. See Chapter 7, “Pre-installation setup,” on page 47.

2. Make sure the database server is configured to accept external connections (TCP).
3. Follow the installation instructions. Starting at Chapter 8, “Installing the Management Console,” on page 67, do the following:

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

Only panels that require specific entries are described in the following steps.

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

**CAUTION:**

**Risk of data loss: If you select Yes, the Build Forge database schema is overwritten in the database. All data from any Build Forge operations on the first installation is lost.**

- Click **Perform Test**. When the test passes, you can click **Next** to proceed.

**Note:** When **No** is selected, the only test performed is a check for the correct path to the JDBC driver.

6. At the **Console Start Options** panel, the **Do Not Start Build Forge** option is selected. It is also greyed out so it cannot be changed. You need to start the console manually after installation.
7. Perform any necessary post-installation configuration. See Chapter 12, “Post-installation tasks,” on page 167. If necessary, catalog the database so that the database client can connect to it.

## Working with redundancy

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

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

---

## Enabling IPv6 network support

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

### About this task

Configuring a Management Console for IPv6 requires the following::

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

Review the requirements and configuration steps for using Build Forge with IPv6. See “Networking requirements for IPv6 support” on page 36.

## Modifying httpd.conf

### About this task

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

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

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

## Setting up the FLEXlm client

### About this task

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

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

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

To configure the IPv6 version of the FLEXlm client:

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

|            |                                                                                                                                                                                                    |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Windows    | C:\<bfinstall>\Apache\tomcat\webapps\rbf-services\bin, where <bfinstall> is the Build Forge installation directory.<br><br>The default installation directory is C:\Program Files\IBM\Build Forge. |
| UNIX/Linux | <bfinstall>/server/tomcat/webapps/rbf-services/bin, where <bfinstall> is the Build Forge installation directory.<br><br>On UNIX/Linux, the default installation directory is /opt/buildforge.      |

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

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



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

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

5. Start the Build Forge engine.

---

## Security features

This section describes ways to enable security features in Build Forge® :

- Secure login, which is enabled by default during installation
- Single sign-on (SSO)
- Enabling HTTPS and SSL
- Enabling password encryption
- `bfclient.conf` file for security configurations

These features are enabled through a combination of selections in the management console at **Administration** → **Security** and manual setup of configuration files. This section includes a reference section on `bfclient.conf`, a configuration file used to enable security features.

**Note:** This section is not for users who run Build Forge on z/Linux. See “Installing the Management Console on Linux on System z” on page 89 for information about security features available in Build Forge on z/Linux.

### Secure login

Build Forge includes login security by default. When a user logs in, the request is redirected to an authentication servlet. The user name and password entered is encrypted for use by the servlet. If the login is successful, the console user interface home is shown. The subsequent session communications between the client and the console may be over http (the default) or over https. Using https requires additional configuration of the system. See “Enabling SSL and HTTPS” on page 115.

During installation, you provide a password for the keystore that is used for encryption. You also have the option of installing a self-signed certificate.

#### Certificate messages about self-signed certificate

If you have Build Forge install a self-signed certificate, users accessing the system through a security-enabled browser get warning messages about the certificate.

To prevent those warnings, distribute the certificate to users for installation in their browser. The specifics of installing the certificate vary by browser. Consult the browser documentation.

The certificate is located in `<bfinstall>/keystore`.

#### Disabling login security

If login security is disabled, then during login user credentials are communicated to the console in unencrypted cleartext. Disabling login security does not effect the use of HTTPS/SSL by the console, if the console is configured to use it.

To disable the authentication servlet, do the following:

1. Stop Build Forge if it is running.
2. Edit `<bfinstall>/buildforge.conf` to specify HTTP and port 8080 in communications with the services layer.  
Change this line:  
`services_url https://hostname:8443/rbf-services`  
  
Change the line to the following:  
`services_url http://hostname:8080/rbf-services`
3. Edit the services layer configuration file to turn off forced SSL. Edit `<bfinstall>/Apache/tomcat/webapps/rbf-services/WEB-INF/web.xml`. Change the `ForceHttps` setting to `false`.  
`forceHttps=false`
4. Start Build Forge.

**Note:** If the authentication servlet is disabled, user credentials are communicated in clear text over the network. This is a security risk.

## Implementing single sign-on

A single sign-on framework is provided with Build Forge.

Single sign-on is an authentication scheme that allows users to access an application without entering a user name and password each time. Build Forge ships a framework that can be used in combination with a third-party HTTP interceptor to implement single sign-on.

### About the single sign-on framework

The Build Forge SSO framework provides the capability to integrate with many SSO solutions on the market. The SSO framework is interceptor-based, meaning that it intercepts an HTTP request and provides methods for handling it. You can write custom interceptors to receive and validate security artifacts in the HTTP request. In particular, the interceptor can set tokens in the HTTP response and then look for those tokens in a successive request.

Two SSO solutions are provided with Build Forge:

- Interceptor for SPNEGO (Simple and Protected Negotiation Protocol). See “Implementing single-signon using SPNEGO in an Active Directory domain” on page 106.
- Interceptor for an integration with WebSphere SSO. See “Integrating with WAS security using a custom interceptor” on page 111.

**SSO framework methods:** An SSO interceptor is a Java class that implements an interface used by the Build Forge SSO framework:

```
com.buildforge.services.server.sso.ISSOIntercaptor
```

It is located in the services layer component:

```
<bfinstall>/Apache/tomcat/webapps/rbf-services/WEB-INF/classes
```

The interface provides the following methods.

#### **initInterceptor**

Called when the interceptor is loaded. A map of configuration properties is passed to the `initInterceptor()` method. Configuration properties are created in the Build Forge console at **Administration** → **Security** → **SSO**.

**isTargetInterceptor**

Reviews attributes in the inbound request to determine if this interceptor needs to act on them. If so, the interceptor is responsible for authenticating the request with the `authenticateRequest()` method. Otherwise this interceptor is skipped. Interceptor selection assumes that multiple interceptors are configured in running. They are addressed in order.

**authenticateRequest**

Authenticates the request using data in the request. It uses a response attribute to send data back to the client.

**logoutRequest**

Cleans up any user-related security information after the request is handled.

**Interceptor configurations and ordering:** Interceptor configurations are defined in **Administration** → **Security** → **SSO**. The following configurations are shipped with Build Forge:

- Form SSO Interceptor - active by default, implements a simple login form.
- SPNEGO SSO Interceptor - inactive by default, implements SPNEGO to perform authentication.

After you implement an interceptor class and place it in the Build Forge Apache Tomcat application server, you configure a new SSO configuration here. The class is one property of the SSO configuration.

The order of this list determines the order in which interceptors are consulted to handle requests. You can configure multiple interceptors to handle requests. During a login, each interceptor is consulted in order. The interceptor that handles the request is the first active interceptor whose attributes are appropriate for the attributes in the request. Only one interceptor handles the request. It is always the first one that responds true for `isTargetInterceptor`.

**Note:** The Form SSO Interceptor should be left active to provide a fallback in the case of error. Custom interceptors should be placed in the list before it.

**Adding a custom SSO interceptor:** To create a custom interceptor in Build Forge, do the following:

1. Create a custom Java class.  
The class must implement the `ISSOInterceptor` interface.
2. Deploy the custom class to the services layer component WAR.
  - a. Create a JAR file containing the compiled custom SSO interceptor class.
  - b. Copy the JAR file to the Build Forge services layer component in the following location:  
`<bfinstall>/Apache/tomcat/webapps/rbf-services/WEB-INF/classes`
  - c. Unzip the JAR file in this directory. The `SSOManager` looks for the class to load here.
  - d. Restart Build Forge.
3. Optional: define an environment. This environment can be passed to the `initInterceptor()` method as a `Properties` object.
  - a. In the Management Console, go to **Environments**.
  - b. Click **Add Environment**.
  - c. Define all of the properties needed by the SSO interceptor to initialize.

4. Add the SSO interceptor to Build Forge:
  - a. In the Management Console, go to **Administration** → **Security** → **SSO**.
  - b. Click **Add SSO Configuration**. Fill in its properties:
    - **Name** - enter a name for the SSO configuration.
    - **Active** - set to Yes. Active configurations are all accessed during an authentication request. They are accessed in the order in which they appear in this panel.
    - **Java Class** - enter the full package name of the class. A given class can be assigned to only one SSO interceptor.
    - **Environment** - if you defined an environment for use with this SSO interceptor, select it.
  - c. Click **Save**.

Your SSO interceptor now appears in the list.

5. Order the SSO configurations. Click the icon to the left of your SSO Interceptor, and then select **Move to Top**.

During a request, active SSO configurations are accessed in the order in which they appear in this panel. Your configuration must be placed before the **Form SSO** configuration, because it is active by default and always returns true when accessed. The **SPNEGO SSO** configuration is inactive by default.

**Example authenticateRequest implementation:** The following example is taken from the WebSphere SSO Interceptor, which is used to integrate WebSphere security with Build Forge.

The interceptor uses reflection to find the WebSphere class WSSubject. The class has a `getCallerPrincipal` method to return the principal used to log in to the `AuthServlet`. The `AuthServlet` needs to be protected before WAS will authenticate with it.

Other methods are available that can return even more information. Similar methods are available to work with any application server.

```
public Result authenticateRequest
 (Request requestAttributes, Response responseAttributes)
 throws SSOException {

 Result result = null;

 try {
 Class<?> c1 =
 Class.forName("com.ibm.websphere.security.auth.WSSubject");
 Method theMethod = c1.getMethod("getCallerPrincipal",
 (Class[])null);
 String principal = (String)theMethod.invoke((Object[])null,
 (Object[])null);

 if (principal != null
 && principal.length() > 0
 && !principal.equals("UNAUTHENTICATED")) {
 result = new Result(Result.UserIdOnlyOID, domain, principal);
 responseAttributes.setStatus(HttpServletResponse.SC_OK);
 } catch (Exception e) {
 throw new SSOException(e);
 }

 return result;
 }
}
```

During the implementation of `authenticateRequest`, you must set a response status before returning:

- If you do not need to perform any redirection and the information found is good, you can return the following:  
`responseAttributes.setStatus(HttpServletResponse.SC_OK);`
- If there is not enough information in the request to continue a valid login, return the following:  
`responseAttributes.setStatus(HttpServletResponse.SC_FORBIDDEN);`
- If you need to perform a redirection to gather additional information, return the following:  
`responseAttributes.setStatus(HttpServletResponse.SC_MOVED_TEMPORARILY);`  
`responseAttributes.sendRedirect(url);`

There are additional status values that can be used. See the JavaDoc for `HttpServletResponse`.

**Recovering from a login error:** If your custom interceptor does not work correctly when tested, the most likely issue is authentication. You see an error page with the following information:

Build Forge Error

Access is denied to the Build Forge console

Error authenticating:  
com.buildforge.services.common.api.APIException - API:  
Authentication Error."

Please click [here](#) to try the same type of login again  
or click [here](#) to force a form login (user ID/password).

You have two options for recovery:

- Retry the login. It goes through the list of configured interceptors again in the same way.
- Force a form login. This choice bypasses the custom interceptor and uses the form login page.

**Method source listing:** The following comments and source listings provide more information about the methods in the `ISSOInterceptor` interface.

#### **initInterceptor**

```
/**
 * This method is called when the interceptor is loaded. A map of the
 * configuration properties is passed into the init method. You can create
 * the configuration properties from a BuildForge Environment and associate
 * it with the SSO configuration.
 *
 * @param initializationProps used to configure the implementation
 * @return true if successful, false if an error should be reported.
 * @throws SSOException if the initialization fails
 */
public boolean initInterceptor (Properties initializationProps) throws SSOException;
```

#### **isTargetInterceptor**

```
/**
 * This methods will review the attributes in the requestAttributes Map
 * to determine if there is something that this interceptor should
 * act on. If the interceptor return is "true", then the interceptor will
 * be responsible for authenticating the request and the authenticateRequest
 * method is invoked. If the interceptor return is "false", then this
 * interceptor is skipped and the next isTargetInterceptor in the list will
```

```

 be called. Ordering of the interceptors during the configuration will
 return which interceptor has the first shot at authenticating a request.
 *
 * @param requestAttributes attributes found in the inbound request
 * @return true if this interceptor will authenticate the request,
 * false if it will not.
 * @throws SSOException
 */
public boolean isTargetInterceptor(Request requestAttributes) throws SSOException;

authenticateRequest

/**
 * This method is called on an interceptor that returns true for the
 * isTargetInterceptor method. The Request will contain data used
 * to perform the authentication. The Response is for the interceptor
 * to send information back to the client. The Result returned will contain
 * the following information if the status code is 200:
 *
 * *
 * * OID: an object identifier of the SecurityContext that can process token
 * information stored in this map when going to an Agent.
 * * Domain: a valid BF domain name or <default> if not known
 * (the username must be valid in the configured realm).
 * * Username: a valid BF username. This will be used to lookup BFUser attributes
 * that are used in checking authorization policy.
 * * @see com.buildforge.services.common.security.context.Result
 * *
 * * @param requestAttributes attributes found in the inbound request
 * * @param responseAttributes sent back in the outbound response
 * * @return com.buildforge.services.common.security.context.Result - result
 * information that tells BF how to handle the authentication request.
 * * @throws com.buildforge.services.server.sso.SSOException
 */
public Result authenticateRequest(
 Request requestAttributes,
 Response responseAttributes)
 throws SSOException;

```

#### **LogoutRequest**

```

/**
 * This method is called to logout a request. The first interceptor that
 * returns true for the isTargetInterceptor method will perform the logout.
 * The main point is to clean up any user-related security information that
 * should not be kept. The interceptor can inspect the request and response
 * objects to determine what needs to be removed.
 *
 * *
 * * @param requestAttributes attributes found in the inbound request
 * * @param responseAttributes sent back in the outbound response
 * * @return boolean - true if request redirect to exit page,
 * false if redirect to login page.
 * * @throws com.buildforge.services.server.sso.SSOException
 */
public boolean logoutRequest(
 Request requestAttributes,
 Response responseAttributes)
 throws SSOException;

```

## **Implementing single-signon using SPNEGO in an Active Directory domain**

A Simple and Protected GSS-API Negotiation (SPNEGO) mechanism is provided to implement single sign-on in Active Directory domains

## Before you begin

This task requires the following elements in your network:

- Active Directory domain
- Directory server host name
- Kerberos Key Distribution Center (KDC) host name
- Build Forge installation on a host in the Active Directory domain
- Client host in the Active Directory domain
- Kerberos configuration files on each client
- Windows Server 2003 SP2 resource toolkit installed on the directory server host
- Supported web browser

**Note:** Internet Explorer 6 is not supported for use with SPNEGO. Use a supported browser. See “Web client requirements” on page 32.

The following procedures include examples based on the following setup:

- mycompany.com is the name of the TCP/IP domain used by all hosts in the domain.
- ITDEV.COM is the name of the Active Directory domain.
- it\_directory.mycompany.com is the host where the directory server runs. It also runs the Kerberos KDC.
- it\_domain.mycompany.com is the host where the Active Directory domain controller runs.
- it\_buildforge.mycompany.com is the host where Build Forge is installed.
- bfuser is the domain user name for the Build Forge system.
- happy\_user is the domain user name for an example user who will use SSO in a web browser to access Build Forge.

## About this task

Perform the following tasks to implement the SPNEGO SSO in an Active Directory domain and KDC. Each is given a section with detailed procedures.

**Note:** The SPNEGO interceptor can be used with KDCs other than Active Directory.

## Procedure

1. Set up Active Directory users and service principals.
2. Set up Kerberos files.
3. Configure Build Forge to use Active Directory and SPNEGO.
4. Configure browser clients for secure access.
5. Access Build Forge through SSO.

### Setting up Active Directory users and service principals:

The Build Forge server and Build Forge clients must be set up in an Active Directory domain.

## Before you begin

Support Tools for Windows 2003 SP2 are required for the following procedure. They contain the `setspn` command, which is required to set a service principal in Active Directory. Install Support Tools from the Windows Server 2003 product CD or the Microsoft Download Center.

## About this task

When the Build Forge client and server are in an Active Directory domain, a user generates a Kerberos credentials token when logging into a Windows host. When the user then attempts to access the Build Forge server, the SPNEGO interceptor receives the user token and validates it. The validated identity is passed to the Build Forge to perform a login through the configured Microsoft Active Directory LDAP server.

## Procedure

1. Log on to the domain controller host. In the example, the host is `it_example.mycompany.com`.
2. Add the Build Forge host to the Active Directory domain if it is not already a member. In this example, add host `it_buildforge` to the `ITDEV.COM` domain. The host now has a fully qualified name in the domain:  
`it_buildforge.ITDEV.COM`
3. Add a Build Forge user to the Active Directory domain. In this example, create user `bfuser`.

### Important:

- Select **Password never expires**. You may select other password management. However, you will need to enter a new password for the Build Forge server each time it expires.
  - In the **Accounts** tab, select **Account is trusted for delegation**
4. If they do not exist, create user accounts in Microsoft Active Directory for all clients. In this example, there is one user to create, `happy_user`.
  5. Create a service principal name (SPN) for Build Forge. In the example, the Active Directory user `bfuser` is associated with service name `HTTP/it_buildforge.mycompany.com` to create the SPN for the Build Forge server, `it_buildforge`.  

```
setspn -A HTTP/it_buildforge.mycompany.com bfuser
```

`HTTP` is the service name for the Build Forge service.

## Setting up files for Kerberos authentication:

A startup file (Kerberos client configuration file) and a keytab file need to be set up on the Build Forge host.

## Procedure

1. Set up the startup file on the host where Build Forge runs.
  - Windows systems:
    - Name the file `krb.ini` and place it in `C:\winnt`. Create `C:\winnt` if it does not exist.
    - Set `default_keytab_name` to `FILE:C:\winnt\krb5.keytab`.
  - UNIX and Linux systems:



- Name the file `krb.conf` and place it in `C:\winnt`.

Set `default_keytab_name` to `FILE:/etc/krb5.keytab`.

The following example file is set up for Windows using domain and realm settings from the example systems.

```
[libdefaults]
default_realm = ITDEV.COM
default_keytab_name = FILE:C:\winnt\krb5.keytab
default_tkt_enctypes = rc4_hmac
default_tgs_enctypes = rc4_hmac
kdc_default_options = 0x40800000
forwardable = true
renewable = true
noaddresses = true
clockskew = 300
[realms]
ITDEV.COM = {
 kdc = it_directory.itdev.com:88
 default_domain = mycompany.com
}
[domain_realm]
.mycompany.com = ITDEV.COM
```

**Note:** Tokens will not work if the clock skew between client hosts and the Build Forge server host is more than 300 seconds. Set the time, date, and time zone to within skew limits on the client and server hosts.

2. Set up a Kerberos keytab file. The keytab file is used by the Build Forge server to validate Kerberos tokens when a client attempts to access the Build Forge server URL. Use the `ktpass` command on the Domain Controller host to create the file. The `ktpass` command is included in the prerequisite Windows resource toolkit. The following example uses the principal name of the Build Forge service and the Active Directory user name set up for Build Forge in the example scenario. Substitute your own password for `-pass Rational`. Line breaks are shown in the example for clarity. Do not use them in your `ktpass` command.

```
ktpass -out C:\it_buildforge.keytab
-princ HTTP/it_buildforge.mycompany.com@ITDEV.COM
-mapuser bfuser -mapop set
-pass Rational /crypto RC4-HMAC-NT /rndpass /ptype KRB5_NT_SRV_HST
```

Rename `it_buildforge.keytab` to `krb5.keytab` and put it on the Build Forge host in the directory that contains the Kerberos startup file.

- Windows: `C:\winnt\`
- UNIX and Linux: `/etc`

## Configuring Build Forge to use Active Directory and SPNEGO:

### Procedure

1. In Build Forge, set up LDAP to point to the Active Directory domain controller.
  - a. In Build Forge, click **Administration** → **LDAP**.
  - b. Set up access to your domain controller by creating a new LDAP configuration and setting properties as follows.
    - Name: set to the name of your Active Directory domain. In the example environment this would be `itdev`.
    - Admin DN: set to an administrator user in the domain.
    - Map Access Groups: No
    - Host: set to the IP address of the domain controller host.

- Bind User Account: Yes
  - Protocol: LDAP
  - Display Name: displayname
  - Distinguished Name: distinguishedname
  - Group Name: memberof
  - Mail Name: displayname
  - Search Base: on=users,do=domainname,do=domainextension. In the example environment this would be on=users,do=itdev,do=.com
  - Unique Identifier: sAMAccountNames=%
- c. Click **Make Default**. This configuration needs to be the default LDAP configuration.
2. Set Build Forge environment variables for SPNEGO.
- a. In Build Forge, go to **Environments** → **Environment for SPNEGO SSO**.
  - b. Set bf\_spnego\_service\_name to HTTP This matches the service principal name.
  - c. Set bf\_spnego\_server\_name to it\_buildforge.mycompany.com, the fully qualified host name for the Build Forge server host. If this variable is not set, INetAddress APIs attempt to locate the host name.
  - d. Set bf\_spnego\_realm to ITDEV.COM, the Kerberos realm name. If this variable is not set, the value in the Kerberos startup file is used.
3. Enable the SPNEGO interceptor.
- a. In Build Forge, go to **Administration** → **Security** → **SSO** → **SPNEGO SSO Interceptor**.
  - b. Set the Active property to Yes, and then click **Save**.
  - c. In **Administration** → **Security** → **SSO**, move **SPNEGO SSO Interceptor** to the top of the list. Use the **Move to Top** selection in the SSO Options menu for SPNEGO SSO Interceptor, then click **Save**.

### Configuring client browsers for SSO:

Client browsers must have security settings set up to use SPNEGO.

#### About this task

Use the client setup instructions for the browser used to access Build Forge, either Microsoft Internet Explorer or Mozilla Firefox.

#### Procedure

- For Internet Explorer, do the following.
  1. Log in to the Active Directory domain. In the example configuration, you would log in to itdev.com.
  2. Open Internet Explorer.
  3. Click **Tools** → **Internet Options** → **Security**.
  4. Select **Local intranet**, and then click **Sites**.
  5. In the **Local intranet** dialog, check **Include all local (intranet) sites not listed in other zones**, then click **Advanced**.
  6. Add the host where Build Forge runs to the **Web sites** list, and then click **OK**.
  7. Click **OK** to close the **Local intranet** dialog.
  8. In the **Internet Options** window, click the **Advanced** tab.

9. Scroll to the Security group and select **Enable Integrated Windows Authentication (requires restart)** if it is not already selected.
10. Click **OK**.
11. Restart Internet Explorer.
- For Mozilla Firefox, do the following.
  1. Log in to the Active Directory domain. In the example configuration, you would log in to itdev.com.
  2. Open Firefox.
  3. Type **about:config** in the address field.
  4. In the Filter box, type **network.n**. The list updates itself.
  5. Double-click **network.negotiate-auth.trusted-uris**. Enter the list of trusted domains. It should include the directory server host and the Build Forge server host (it\_directory.mycompany.com and it\_buildforge.mycompany.com, in the example). Click **OK**.
  6. Set up delegation. Double-click **network.negotiate-auth.delegation-uris** and enter the list of sites to which the browser can delegate user authentication.

### Access Build Forge through SSO:

Enter the server URL to test login through SSO.

#### Procedure

1. Log on to a host that is in the Active Directory domain, using a user name that is in the Active Directory list of users.
2. Open your browser.
3. Enter the URL for the Build Forge server host. Using the example configuration, this would be http://it\_buildforge.mycompany.com. If SSO is configured correctly, you see the Build Forge management console.
4. Verify that the user name shown on the upper right of the Build Forge console matches the client's Windows login name.

### Integrating with WAS security using a custom interceptor

This section describes how to create an SSO interceptor to integrate with WebSphere Application Server (WAS) security.

#### Before you begin

The provided Form SSO Interceptor authenticates users with form-based login page. The following is an example of how to create a custom SSO interceptor. The custom interceptor uses a custom interceptor class.

The interceptor class accesses WAS to obtain authenticated user credentials. After those credentials are obtained, they are cached. Subsequent logins use the cached credentials.

*Prerequisite:* A user must be configured in WAS with LDAP user credentials.

**Note:** You must make the Build Forge LDAP domain containing the WAS users the "default" LDAP server. To do this, go to this Build Forge LDAP domain, and select "Make Default."

#### Protecting your authorization service (AuthServlet):

Build Forge normally runs its services as an application in the provided Apache Tomcat application server.

### About this task

The following instructions configure Build Forge to use WAS 6.1 instead of Tomcat. Follow the instructions in the section “Using WebSphere Application Server instead of Apache Tomcat” on page 479, with one exception: the `rbf-services.war` file contains a file named `web.xml` that you need to extract and modify to add a security constraint. Before installing this application under WAS, the war file will need to be regenerated after modifying it to use the protected version of this file.

To do this, follow these instructions:

### Procedure

1. Navigate to the directory containing your `rbf-services.war` file (the webapps directory in your Tomcat server root). Copy this file to a temporary location, such as `C:\rbf`.
2. Expand the WAR file. From command line, run the command:  
`%IBM_JAVA_HOME%\jar -xvf rbf-services.war` to expand the contents of the war file.

**Note:** Java must be available and the `IBM_JAVA_HOME` environment variable must already have been created.

3. Save the `rbf-services.war` file to retrieve later:
  - a. Windows: copy `rbf-services.war` `rbf-services.war.bak`
  - b. UNIX or Linux: `cp rbf-services.war rbf-services.war.bak`
4. Find the `web.xml` file in the `WEB-INF` directory (from the files expanded from the war file). Edit this file to add a security context. For example, add the following lines at the end of the file just before the `</web-app>` tag:

```
<security-constraint id="SecurityConstraint_1">
 <web-resource-collection id="WebResourceCollection_1">
 <web-resource-name>*/</web-resource-name>
 <url-pattern>/AuthServlet/*</url-pattern>
 <http-method>GET</http-method>
 <http-method>POST</http-method>
 <http-method>PUT</http-method>
 <http-method>DELETE</http-method>
 </web-resource-collection>
 <auth-constraint id="AuthConstraint_1">
 <description>myconstraint:+:</description>
 <role-name>User</role-name>
 </auth-constraint>
 <user-data-constraint id="UserDataConstraint_1">
 <transport-guarantee>NONE</transport-guarantee>
 </user-data-constraint>
</security-constraint>
<login-config id="LoginConfig_1">
 <auth-method>BASIC</auth-method>
 <realm-name>full-qualified-domain</realm-name>
</login-config>
<security-role id="SecurityRole_1">
 <role-name>User</role-name>
</security-role>
```

**Note:** The `<auth-method>` can be any J2EE auth-method supported by WAS. The most common auth-method is FORM, which requires additional

configuration parameters. Refer to your WebSphere documentation for instructions on configuring FORM in your application.

5. From command line, regenerate the war file (called from the same directory that it was extracted to) by running the following command:

```
%IBM_JAVA_HOME%\jar -cvf rbf-services.war
```

You should now have a new version of rbf-services that has been modified to protect the AuthServlet with J2EE constraints. Complete the installation instructions for running with WAS and install this version of the rbf-services.war via **Applications->Install New Application**. Make sure that Build Forge is not running while configuring WAS.

After this is installed, go to **Applications->Enterprise Applications**. Click the application name to configure it. Under **Detail Properties**, click the link entitled **Security role to user/group mapping**. Select the **All Authenticated** checkbox for User. After making this change, be sure to save to the master configuration.

Application Security also needs to be enabled under WAS. To do this, go to **Security->Secure administration, applications, and infrastructure**. Make sure that **Enable application security** is checked.

At this point, restart the WAS server, and then restart the Build Forge server.

**Note:** The rbf-services should now start as part of the WAS startup process, so WAS needs to be started before Build Forge.

### Creating a new SSO configuration:

Create a new SSO configuration to use the interceptor.

#### Procedure

1. In the Build Forge console, go to **Administration->Security->SSO**.
2. Click **Add SSO Configuration**.
3. Set properties for the configuration.
  - **Name** - Enter a name for this configuration.
  - **Java Class** - Enter  
com.buildforge.services.server.sso.was.WebSphereSSOInterceptor
  - **Active** - Select Yes.
4. Click **Save**.
5. Move this configuration to appear first in the list. In the menu to the left of the configuration name, select **Move to Top**.

### Mapping LDAP users or an LDAP group to local users:

Map LDAP users or an LDAP group to local users, such as the local root user.

#### About this task

After you create a WebSphere SSO interceptor, you can configure the interceptor to:

- Map LDAP users to local users, including root
- Map an LDAP group to a local user, such as root

To set up a mapping, define an environment and then reference that environment in the interceptor.

*Defining an environment:*

### Procedure

1. In the Build Forge console, click **Environments**.
2. Specify a name for the environment and click **Save Environment**.
3. To map LDAP users to local users, set and save the following variables:
  - a. Set `ldap_user_list_mapping` to `LDAP_user1|local_user1;LDAP_user2|local_user2;LDAP_user3|local_user3`
  - b. Set `ldap_realm_name` to `LDAP_domain_name`

For example, assume the following environment is set:

```
ldap_user_list_mapping = user1|root;user2|root;user3|root
ldap_realm_name = bluepages.ibm.com:389
```

In this case, LDAP user1, user2, and user3 are all mapped to the local root user.

4. To map members of a specific LDAP group to a local user, set and save the following variables:
  - a. Set `ldap_group_name_mapping` to `LDAP_group|local_user`
  - b. Set `ldap_realm_name` to `LDAP_domain_name`

**Note:** If you use both types of mappings, the user mapping takes precedence over the group mapping.

For an example of a group mapping, assume the following environment is set:

```
ldap_group_name_mapping = cn=bf_admin,ou=memberlist,ou=ibmggroups,o=ibm.com|root
ldap_realm_name = bluepages.ibm.com:389
```

In this case, users who are members of the LDAP group `cn=bf_admin,ou=memberlist,ou=ibmggroups,o=ibm.com` are mapped to the local root user.

*Referencing the environment:*

### Procedure

1. In the Build Forge console, click **Administration->Security->SSO**.
2. Click the name of the WebSphere SSO interceptor.
3. Set the **Environment** property for the configuration to the environment you just defined.
4. Click **Save**.
5. Move this configuration to appear first in the list. In the menu to the left of the configuration name, select **Move to Top**.

### Running the SSO custom interceptor:

You can now log in using your new configurations.

### About this task

This customized SSO interceptor will now allow WAS security techniques to authenticate the user, via an `AuthServlet` request to be passed to Build Forge as a user.

### Procedure

1. Open your web browser. Enter the address `http://localhost`.
2. Instead of the Build Forge login form, you will now see an authentication page. Enter your user credentials and hit **Enter**.
3. After authentication, note that login should occur automatically.

4. After logging out, it will display the default jsp page instead of the login form. Subsequent logins will be automatic if the user is still authenticated.

### Reverting to form-based SSO:

You can revert to using the SSO login form.

### About this task

To reconfigure the systems to use the Form Login, under WAS you must uninstall rbf-services and reinstall the original rbf-services.war file. Under Build Forge, ensure that the form-based SSO interceptor is enabled and listed as the top item. Disable the Custom WAS interceptor. WAS and Build Forge need to be restarted for these changes to take place.

### Procedure

1. Uninstall rbf-services and reinstall the original rbf-services.war file.
2. Under Build Forge, ensure that the form SSO interceptor is enabled and listed as the first item (see "Build Forge SSO security configurations," above).
3. Disable the Custom WAS interceptor.
4. Restart WAS.
5. Restart Build Forge.

## Enabling SSL and HTTPS

Configuring the Build Forge system to use SSL and HTTPS increases the security of the system. SSL includes endpoint authentication and data encryption.

By default the Build Forge system uses SSL only for the login form, which uses the authentication servlet on Apache Tomcat. Enabling additional SSL protection throughout the Build Forge system requires the following setup:

1. Enabling SSL on the Apache server. This step is needed only if you did not specify that SSL be enabled during installation.
2. Enabling SSL for client and internal communications
3. Enabling SSL for agents

**Note:** If you are integrating with WebSphere components, be sure that the prerequisites for SSL support are met.

- For integration with WebSphere Application Server: see "Using WebSphere Application Server instead of Apache Tomcat" on page 479.

### About SSL and Build Forge components

Build Forge components are set up by default to use certain ports and security settings when SSL is enabled.

**Default SSL Setup:** Enabling SSL is relatively simple when you use the default certificates. Procedures in this section are based on that scenario.

However, it is generally not wise to use the same certificate (private key) on each system. If the private key on one system is compromised, then the entire infrastructure could be compromised. The chances of compromise can be reduced by enforcing physical security.



A more secure system uses a certificate for each process. In Build Forge that means you do the following:

- Create a certificate for every agent.
- Create a certificate for every engine. This is applicable when redundancy is set up. See “Configuring redundancy” on page 98.

This setup requires additional certificate management. You have choices:

- You can use a CA (Certificate Authority) for generating certificates. Doing this reduces the number of signer exchanges.
- You can ensure that every trust store or CA store has the signers needed for making connections.

The following sections identify the interfaces in the Build Forge system where SSL security is enforced.

**Client interfaces:** Users access the Build Forge system through client interfaces.

#### **Web Client to Build Forge**

Web clients access Build Forge through its Apache web server. When SSL is enabled and a security-enabled web browser is used, the following interfaces are used.

- **Apache web server port 443**

Web clients access Build Forge through its URL. When SSL is enabled, the URL is the following:

`https://host/`

The *host* is the host where Build Forge runs. If you set up a port other than 443 for secure access to Apache, users must also specify the port:

`https://host:port/`

Web clients are redirected to an authentication servlet running on Apache Tomcat server.

- **Apache Tomcat application server port 8443**

An authentication servlet accepts login credentials and authenticates the user. The servlet encrypts the credentials so they never appear in clear text over the wire.

The configuration for the listener port used by the Apache Tomcat servlet is managed through a configuration file. It is located in `<bfinstall>/Apache/tomcat/conf/server.xml`. Locate the following connector configuration.

```
<Connector port="8443" maxHttpHeaderSize="8192" algorithm="IbmX509"
 maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
 enableLookups="false" disableUploadTimeout="true"
 acceptCount="100" scheme="https" secure="true"
 clientAuth="false" sslProtocol="SSL_TLS"
 keystoreFile="C:\BuildForge71.536\keystore\buildForgeKeyStore.p12"
 keystorePass="password"
 keystoreType="PKCS12"
 truststoreFile="C:\BuildForge71.536\keystore\buildForgeTrustStore.p12"
 truststorePass="password"
 truststoreType="PKCS12"/>
```

#### **API Program Client to Build Forge**

- **Apache Tomcat application server port 49150**



API clients access Build Forge through its services layer component, an application running on the Apache Tomcat application server. API clients need to have a valid `bfclient.conf` file.

The services layer component uses an SSL configuration for inbound communications. It is defined on the Build Forge console at **Administration → Security → SSL**. The default used is **Default JSSE Inbound SSL**.

**Internal interfaces:** Build Forge is made up of a web interface component (Apache web server and PHP), a services layer component, and an engine component. The web interface and engine components are clients of the services layer component. API program clients are also clients of the services layer component.

#### Services Layer inbound

##### Apache Tomcat application server port 49150

The services layer component uses an SSL configuration for inbound communications. It is defined on the Build Forge console at **Administration → Security → SSL**. The default used is **Default JSSE Inbound SSL**.

#### Services Layer client outbound

**Apache Tomcat application server port 49150** The web interface component (through PHP) and the engine component both use an SSL configuration dedicated for outbound communications to the services layer component. It is defined on the Build Forge console at **Administration → Security → SSL**. The default used is **Default JSSE Outbound SSL**.

The SSL properties for the client outbound configuration and services layer inbound configuration need to be compatible for an SSL handshake to be successful. **Administration → Security → SSL**, the **Type** and **Handshake Protocol** properties must match.

Each SSL configuration has a reference keystore configurations:

- **Keystore Configuration:** defines the properties of a keystore that contains private certificates.
- **Truststore Configuration:** defines the properties of a keystore that contains trusted signers.

The configurations are specified by name. You define them in **Administration → Security → SSL**. Several defaults are provided.

**External interfaces:** External interfaces are those used by Build Forge to talk to external systems.

- The Build Forge engine communicates with agents.
- The Build Forge services layer component communicates with the database.

#### Build Forge engine to agent communications

Enabling SSL for this interface requires the following:

- Configuration of the agent. It requires a change in the agent configuration file and placing certificates on the agent host.
- Enabling SSL communications for each Server resource that uses the agent. You do this on the console in the **Servers** panel.

See “Enabling SSL for agent communications” on page 120.

### Build Forge services layer component to database communications

The SSL configuration for this interface is defined in the device driver for your database.

### Enabling SSL on the Apache server

Build Forge components are set up by default to use certain ports and security settings when SSL is enabled.

During installation you are asked the following question in the Web and Application Server panel: **Do you wish to use secure HTTP?**

If you answered yes, then SSL on the Apache server is enabled. Skip this section.

If you answered no, but you do want to enable SSL, then log on to the Build Forge host and configure the Apache server as follows:

- `httpd.conf` file  
Edit `<bfinstall>/Apache/conf/httpd.conf` to use these settings:  
`Listen 0.0.0.0:443`  
`ServerName localhost:443`
- `ssl.conf` file  
Edit `<bfinstall>/Apache/conf/ssl/ssl.conf` to use these settings:  
`<VirtualHost *:443>`  
`ServerName localhost:443`  
`SSLEngine on`

You may specify the fully qualified domain name for the host if you choose.

### Enabling SSL for client and internal connections

Build Forge components are set up by default to use certain ports and security settings when SSL is enabled.

To enable SSL connections from clients to Build Forge and among Build Forge internal components, do the following in exactly the order specified:

- In the console, go to **Administration** → **Security**.
- Set **SSL Enabled** to Yes. Several additional properties are shown. *Leave them set to the default.* They can be customized later if required.
- Click **Save**. The SSL configuration is saved in the Build Forge database.
- Click **Update Master BFClient.conf**. The SSL configuration is used to update the `BFClient.conf` file. The settings must be in this file for Build Forge to use them.
- Stop Build Forge and restart. *This is required.*

Once the system is proven to work with the default settings, you may edit the properties that affect SSL.

**Note:** If you want to have clients use SSL but do not want to use SSL between internal clients (web interface component and engine component) and the services layer component, do the following:

- Edit the `BFClient.conf` file manually. It is in `<bfinstall>`.
- Change the `bf_services_preferred_protocol` setting to `tcp`.  
`bf_services_preferred_protocol` to `tcp`

This configuration improves performance at minimum security risk if the Build Forge host is physically secured.

## Re-enabling TCP communications on a locked system

If there is a misconfiguration in SSL, the system locks you out.

To get access to a locked system, do the following:

- Stop Build Forge.
- In the *<bfinstall>* directory, open the *bfclient.conf* file with a text editor.
- Change the protocol property as follows:  
    `bf_services_preferred_protocol=tcp`
- Start Build Forge.

You should be able to log in.

**Note:** Changing the protocol does not disable secure login authentication, which is enabled by default. Users are redirected to a secure connection that allows secure communication of login credentials to Build Forge.

## Enabling SSL for an API client (Perl or Java)

Use *bfclient.conf* to configure SSL connections for an API client written in Perl or Java.

### About this task

To make an SSL connection with an API client program, set up *bfclient.conf*. This file contains the SSL configuration properties. To simplify this procedure, the steps below assume you run your client in the client directory mentioned in the first step.

### Procedure

1. Create a client directory where you are going to run the script.  
    Windows: `mkdir c:\client`  
    UNIX or Linux: `mkdir /temp/client`
2. Copy *bfclient.conf* to the client directory.  
    Windows: `copy c:\BuildForge\bfclient.conf c:\client`  
    UNIX or Linux: `cp /opt/BuildForge/Platform/bfclient.conf /temp/client`
3. Create a keystore subdirectory in the client directory.  
    Windows: `mkdir c:\client\keystore`  
    UNIX or Linux: `mkdir /temp/client/keystore`
4. Copy the keystores from the BuildForge installation into the client keystore directory.  
    Windows: `copy \BuildForge\keystore\*.pem \client\keystore`  
    UNIX or Linux: `cp /opt/BuildForge/Platform/keystore/*.pem /temp/client/keystore`
5. For Perl, set the following OS environment variable so that the *bfclient.conf* location is found.  
    Windows: `set BFCLIENT_CONF=c:\client\bfclient.conf`  
    UNIX or Linux: `export BFCLIENT_CONF="/temp/client/bfclient.conf"`
6. For Java, complete these steps:
  - a. Use *SecureAPIClientConnection* instead of *APIClientConnection* in your code to make an SSL connection using *bfclient.conf*.

- b. Set the following System property on the Java command line when running your script.  
Windows: `-Dcom.buildforge.client.config=c:\client\bfclient.conf`  
UNIX or Linux: `-Dcom.buildforge.client.config=/temp/client/bfclient.conf`
7. Edit `bfclient.conf` and check the following properties:
  - `bf_services_preferred_protocol=ssl`  
Make sure the value is `ssl` instead of `tcp`.
  - `bf_services_ssl_port=49150`  
Make sure 49150 is your Services Layer SSL port.
  - `bf_keystore_location=./keystore/buildForgeKey.pem`  
There are several keystore locations. If you run your script in a directory other than the client directory, change each keystore location to use a fully qualified path.
8. If you want to be sure that your script is using `bfclient.conf` correctly, set the following debug property in your environment. When you run your script, you should see more output about the SSL connection properties.  
Windows: `set BFDEBUG_SECURITY=1`  
UNIX or Linux: `export BFDEBUG_SECURITY=1`

## Enabling SSL for agent communications

Build Forge components are set up by default to use certain ports and security settings when SSL is enabled.

To enable SSL communication between Build Forge and agents, you need to do the following:

- On UNIX computers, ensure that the GCC library is installed on the computer hosting the agent.
- Prerequisite: enable SSL for client and internal communications. See “Enabling SSL for client and internal connections” on page 118.
- Configure each agent. This task includes:
  - Adding certificates to the agent host
  - Editing the `bfagent.conf` file for the agent.
- In the console, enable SSL in each Server definition that connects to the agent.

## Configuring agents for SSL

1. If the agent is running, stop it.
2. Place .PEM files for the certificates in the root installation directory for the agent.

To implement and test SSL quickly you can copy the .PEM files from the Build Forge installation. The files are in `<bfinstall>/keystore`.

The best practice for SSL is to use a separate certificate for each agent:

- a. Create separate keystore files (.PEM) for each of the following:
  - private key (key)
  - public certificate for the private key (cert)
  - trusted signers (ca or certificate authority)
- b. If you are using an unique certificate for the agent (instead of a copy of the Build Forge engine's certificate), then add the certificate for the agent to the certificate authority keystore for Build Forge, `<bfinstall>/keystore/`

buildForgeCA.pem. If you are running multiple engines (redundant engines), add the certificate to each engine's certificate authority keystore.

3. Edit BFAgent.conf. The following lines are commented out in the file. Remove the comment prefix.

```
ssl_key_location buildForgeKey.pem
ssl_key_password password
ssl_cert_location buildForgeCert.pem
ssl_ca_location buildForgeCA.pem
ssl_protocol TLSv1
ssl_cipher_group ALL
```

The *password* is for the buildForgeKey.pem keystore. If you want to encrypt it, see “Encrypting passwords in buildforge.conf and bfaagent.conf” on page 135.

If you want to require client authentication when a connection is made to the agent, uncomment the following line:

```
ssl_client_authentication true
```

This setting requires that the engine certificate be added in the agent's certificate authority keystore, buildForgeCA.pem.

If you want to use specific ciphers, uncomment this line and add your cipher list:

```
ssl_cipher_override cipher_list
```

4. Start the agent. The agent must be running to test the connection from the console.

## Enabling SSL in Server definitions

The console uses Server definitions to connect to agents.

For each Server definition that is connected to an SSL-enabled agent, do the following:

- In the console, go to the **Servers** panel.
- Click the server definition name.
- In the **Details** tab for the server definition:
  - Set **SSL Enabled** to Yes.
- Click **Save**.
- Click **Test Connection**.

## Troubleshooting SSL communication with agents

The following checklist describes common issues when enabling SSL:

- Agent
  - Agent SSL is not configured, though SSL is enabled in Security and in the Server definition.
  - The certificate for the agent is not trusted by the Build Forge engine. The agent certificate needs to be added to the engine's CA keystore:  
`<bfinstall>/keystore/buildForgeCA.pem`
  - Password for the keystore specified in BFAgent.conf is incorrect.
  - Client authentication is specified in BFAgent.conf but the engine's certificate has not been added to the agent's certificate authority, buildForgeCA.pem.
- Build Forge console

- SSL was enabled on the console. It was not saved, or **Update Master Bfclient.conf** was clicked before saving, or **Update Master Bfclient.conf** was never clicked after saving.
- Server definition
  - SSL was not successfully enabled. You must click **Save** before **Test Connection**.
- Engine and Agent settings match
  - Handshake protocol mismatch. The handshake protocol must be set to the same value for both the engine SSL configuration and agent SSL configuration, either TLSv1 or SSLv3. The default value is TLSv1.
  - Cipher suite mismatch. The cipher suites specified in the engine SSL configuration and agent SSL configuration must have ciphers in common. The default cipher suite group is ALL.

## Enabling debugging messages

You can enable debugging in the engine and the agent. When debugging is enabled, additional detailed output is produced that can help identify problems in the configuration.

- Enabling engine debugging:
  1. In an environment used by an SSL-enabled server definition, add the following variable:
 

```
BFDEBUG_SECURITY=1
```
  2. Stop the engine and restart it.
    - On Windows, start the engine in the foreground. Output appears in a command window.
    - On UNIX or Linux, start the engine with debugging turned on:
 

```
cd <bfinstall>/rc
./buildforge start
```

Output from the engine goes to the engine log file in *<bfinstall>/log*.

- Enabling agent debugging:
  1. Stop the agent.
  2. Add the following line to BFAgent.conf:
 

```
activity_log bfaagent.log
```

In this example, the agent writes output to *bfaagent.log*. You can specify a different filename.

**Note:** If the agent runs as a service, specify an absolute path.

3. Start the agent.

## Example engine debug output for a successful SSL connection

An engine produces the following output when it connects successfully to an agent.

```
SSL_ca_file: ./keystore/buildForgeCA.pem
SSL_cert_file: ./keystore/buildForgeCert.pem
SSL_key_file: ./keystore/buildForgeKey.pem
SSL_verify_mode: 0x01
SSL_version: TLSv1
SSL_cipher_list: ALL
SSL_use_cert: 1
```

```

Making as SSL connection using socket IO::Socket::INET=GLOB(0x1e8f0f4).
SSL connection to agent.
DEBUG: .../IO/Socket/SSL.pm:1387: new ctx 80662848
DEBUG: .../IO/Socket/SSL.pm:880: dont start handshake: IO::Socket::SSL=GLOB(0x1e8f0f4)
DEBUG: .../IO/Socket/SSL.pm:284: ssl handshake not started
DEBUG: .../IO/Socket/SSL.pm:327: Net::SSL::connect -> 1
DEBUG: .../IO/Socket/SSL.pm:382: ssl handshake done
Socket is of type: ref(IO::Socket::SSL=GLOB(0x1e8f0f4))
ReadyLine: 202 HELLO TLS - BuildForge Agent v_VERSION_
.
Storing Agent Version [999.999.999.999-999-9999] for [08974C8E-6C3B-1014-972D-D9B2901D9F42]
cmd ping
username pbirk
encpass c1713f4a31af3f1300f7b2414a24559c4d6097e07310cf9c412e
go
Sending agent request...

```

## Example agent debug output for a successful SSL connection

An agent running normally produces the following output when it establishes an SSL connection.

```

[2256] main.c : 409: === NEW AGENT ===
[2256] io.c : 264: In start SSL
[2256] io.c : 89: Key location: buildForgeKey.pem
[2256] bfpwdloader.c: 134: Looking for password locator: ssl_key_password_locator
[2256] bfpwdloader.c: 244: Looking for password for prop
 ssl_key_password from bfaagent.conf.
[2256] bfcryptloader.c: 202: Loading password encryption module.
[2256] bfcryptloader.c: 276: Password encryption property
 password_encrypt_module is not configured.
[2256] bfcryptloader.c: 539: Password decoded.
[2256] io.c : 98: Cert location: buildForgeCert.pem
[2256] bfpwdloader.c: 134: Looking for password locator:
 ssl_cert_password_locator
[2256] bfpwdloader.c: 244: Looking for password for prop
 ssl_cert_password from bfaagent.conf.
[2256] io.c : 153: Setting key password in default userdata.
[2256] io.c : 160: Getting private key from PEM.
[2256] io.c : 166: Checking private key from PEM.
[2256] io.c : 172: Getting CA store information.
[2256] bfpwdloader.c: 134: Looking for password locator:
 ssl_ca_password_locator
[2256] bfpwdloader.c: 244: Looking for password for prop
 ssl_ca_password from bfaagent.conf.
[2256] io.c : 178: CA location: buildForgeCert.pem
[2256] io.c : 184: Checking the CA store.
[2256] io.c : 230: Returning from init_CTX.
[2256] io.c : 281: Calling SSL_new
[2256] io.c : 294: Calling SSL_accept.
[2256] io.c : 346: Cipher chosen: AES256-SHA
[2256] io.c : 367: ssl_state = SS_CERTIFIED

```

## Example output for bad keystore password on the agent

If the keystore password configured on the agent side is wrong, it shows up in both engine and agent output.

Engine output (excerpt):

```

SSL_use_cert: 1
Making as SSL connection using socket IO::Socket::INET=GLOB(0x1e8f0f4).
SSL connection to agent.
DEBUG: .../IO/Socket/SSL.pm:1387: new ctx 80662848
DEBUG: .../IO/Socket/SSL.pm:880: dont start handshake: IO::Socket::SSL=GLOB(0x1e8f0f4)
DEBUG: .../IO/Socket/SSL.pm:284: ssl handshake not started

```



```

DEBUG: .../IO/Socket/SSL.pm:327: Net::SSLeay::connect -> -1
DEBUG: .../IO/Socket/SSL.pm:1135: SSL connect attempt failed with unknown error
error:1408F10B:SSL routines:SSL3_GET_RECORD:wrong version number

DEBUG: .../IO/Socket/SSL.pm:333: fatal SSL error: SSL connect attempt failed with unknown error
error:1408F10B:SSL routines:SSL3_GET_RECORD:wrong version number
DEBUG: .../IO/Socket/SSL.pm:1422: free ctx 80662848 open=80662848 80566656
DEBUG: .../IO/Socket/SSL.pm:1425: OK free ctx 80662848

```

#### Agent output (excerpt):

```

[5272] io.c : 98: Cert location: buildForgeCert.pem
[5272] bfpwdlocloader.c: 134: Looking for password locator: ssl_cert_passwor
d_locator
[5272] bfpwdlocloader.c: 244: Looking for password for prop ssl_cert_password from bfa
gent.conf.
[5272] io.c : 153: Setting key password in default userdata.
[5272] io.c : 160: Getting private key from PEM.
[5272] io.c : 218: Failure reason: SSLERRORBADKEYFILE
[5272] io.c : 221: OpenSSL error string: error:00000000:lib(0):func(0):reason(0)
[5272] io.c : 281: Calling SSL_new
[5272] platform.c :2693: platform_release_credentials
[5272] main.c : 412: --- EXITING ---

```

#### Error codes in agent output

This list includes some of the other error codes you may encounter and their causes:

- **SSLERRORBADCA**: Problem loading the signers in buildForgeCA.pem. It could be a problem in the format of the file resulting from how the signers were added.
- **SSLERRORBADCERT**: Problem loading the certificate in the buildForgeCert.pem. Either the certificate does not match the private key or it is corrupted in the PEM.
- **SSLERRORBADKEYFILE**: Problem with the buildForgeKey.pem password specified for the ssl\_key\_password property in BFAgent.conf.
- **SSLERRORBADKEY**: SSL\_CTX\_check\_private\_key returned a value other than 1. The private key format is not valid or it does not match the certificate.
- **SSLERRORFIPSEnablement**: An error occurred during FIPS enablement. This typically is due to an issue encountered during the FIPS self-check. This is likely an internal error.
- **SSLERRORINVALIDCIPHER**: A cipher specification does not match what OpenSSL allows. Check the ciphers specified on properties ssl\_cipher\_group or ssl\_cipher\_override in BFAgent.conf.
- **SSLERRORNOCTX**: Problem creating a new SSL CTX object. This is likely an internal error.

#### Managing certificates

Certificates and keystores used by Build Forge can be modified after installation.

During installation you are given the opportunity to specify a certificate to use (either your own or one generated by Build forge) and a keystore password. This section describes procedures for the following:

- Converting PEM keystores to OpenSSL and JSSE keystores needed by Build Forge
- Changing the keystore password and modifying Build Forge to use the new password
- Creating a new self-signed certificate
- Logging in with a client certificate



## Converting PEM keystores to Build Forge keystores:

PEM keystores received from a Certificate Authority can be converted into keystores for use with Build Forge.

### Before you begin

Download the unrestricted policy files for your SDK. This prerequisite applies only if your keysize is too large for the restricted policy files. Download the files from [https://www14.software.ibm.com/webapp/iwm/web/reg/signup.do?source=jcesdk&lang=en\\_US&S\\_PKG=142ww](https://www14.software.ibm.com/webapp/iwm/web/reg/signup.do?source=jcesdk&lang=en_US&S_PKG=142ww)

**Note:** You must use the keytool utility provided by IBM.

### About this task

If you have a set of PEM files from a Certificate Authority, you must use them to create a set of OpenSSL and JSSE keystores for Build Forge.

### Procedure

1. Include Build Forge tool directories in your PATH.

- `<bfinstall>/openssl`
- `<bfinstall>/ibmjdk/bin` for Windows
- `<bfinstall>server/ibmjdk/bin` for UNIX or Linux

For UNIX and Linux, include the following directory in LD\_LIBRARY\_PATH:

`<bfinstall>/openssl`

2. Convert the PEM files into a PKCS12 keystore.

Use the following command:

```
openssl pkcs12
 -export
 -name "buildforge"
 -out buildForgeKeyStore.p12
 -inkey <key.pem>
 -passin pass:<pempassword>
 -in <cert.pem>
 -password pass:<bfpasword>
```

3. Verify that the certificate has been added and can be read.

```
keytool -v
 -list
 -keystore buildForgeKeyStore.p12
 -storepass <bfpasword>
 -storetype pkcs12
```

If you get an error about an invalid key size, download unrestricted policy files. Use the directions at the beginning of this section.

4. Export the public certificate.

In a command window, go to `<bfinstall>/keystore`, and then run this command:

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

- The certificate is stored in file `cert.der`.

- Use the same *<bfpword>* that was specified for keystores during installation. Otherwise you need to change the configuration.
5. Create the truststore and import the public certificate.  
In a command window, go to *<bfinstall>/keystore*, then run this command:

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

6. Put the public client certificate in buildForgeCert.pem.  
In a command window, go to *<bfinstall>/keystore*, and then run this command:

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

7. Put the certificate and keys in buildForgeKey.pem  
In a command window, go to *<bfinstall>/keystore*, and then run this command:

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

8. Create the PEM Certificate Authority buildForgeCA.pem.
  - a. Download the CA root certificate to *<bfinstall>/keystore*. It is named CARootCert.crt. It needs to be added to your PEM keystores and can be imported into buildForgeTrustStore.p12.
  - b. In a command window, go to *<bfinstall>/keystore*, and then run these commands:

```
cat CARootCert.crt > buildForgeCA.pem
keytool -import -noprompt -v -trustcacerts
 -alias "CA Root"
 -file CARootCert.crt
 -keystore buildForgeTrustStore.p12
 -storepass <bfpword>
 -storetype pkcs12
```

## Results

Build Forge uses a password-protected PEM keystore, buildForgeKey.pem. The Apache server prompts for the password during startup.

If you do not want to be prompted for a password during startup, then generate a PEM keystore that is not password-protected and have the Apache server use it. The following command is an example.

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

Be sure the unprotected PEM keystore is readable by any user who needs access to the ID of the process that runs Build Forge.

**Changing keystore passwords:** During installation you are given the opportunity to specify a keystore password. If you want to change that password, you need to do the following:

1. Modify the Build Forge keystores to use a new password.
2. Modify the Build Forge configuration to use the new password.

*Modifying the keystore passwords:*

#### **About this task**

Build Forge has three default keystores that are password-protected, all installed on the host where you installed the Build Forge engine, in *<bfinstall>/keystore*:

- *buildForgeKey.pem* - used by OpenSSL, requires the *openssl* tool to change the password.
- *buildForgeKeyStore.p12* - used by JSSE, requires the *ibmjdk* tool to change the password.
- *buildForgeTrustStore.p12* - used by JSSE, requires the *ibmjdk* tool to change the password.

The tools are included with Build Forge software.

**Note:** Line breaks are used for clarity in the example commands. Do not use them in the command. Enter it as one string or use the line-continuation character (^ for Windows, \ for UNIX or Linux).

**Important:** The same password is used for all keystores. It is shown as *newpassword* in the examples.

#### **Procedure**

1. Log on to the host where the Build Forge engine is installed.
2. Put the tool directories on your PATH.
  - *<bfinstall>/openssl*
  - *<bfinstall>/ibmjdk/bin*
3. Disable SSL. In the console, go to **Administration** → **Security**. Set **SSL Enabled** to No.
4. Click **Save**.
5. Click **Update Master BFClient.conf**.
6. Stop the Build Forge engine.
7. Back up the existing key stores. Copy the existing Build Forge keystores to a temporary directory. If the modified files get corrupted, you can use the backed up keystores.
8. Modify *buildForgeKey.pem*. In the directory *<bfinstall>/keystore*, run this command:

```
openssl rsa
-in buildForgeKey.pem
-passin pass:oldpassword
-out buildForgeKey.pem
-passout pass:newpassword -aes128
```
9. Modify *buildForgeKeyStore.p12*. In the directory *<bfinstall>/keystore*, run this command:

```
keytool -storepasswd -all
-new newpassword
-keystore buildForgeKeyStore.p12
-storepass oldpassword
-storetype pkcs12
```

10. Modify buildForgeTrustStore.p12. In the directory *<bfinstall>/keystore*, run this command:

```
keytool -storepasswd -all
-new newpassword
-keystore buildForgeTrustStore.p12
-storepass oldpassword
-storetype pkcs12
```

## Results

After the passwords are changed, you need to modify the Build Forge configuration to use the new passwords.

*Modifying the Build Forge configuration to use the new password:*

Build Forge configurations must be changed to use a changed keystore password.

## Before you begin

Prerequisite:

- The Build Forge engine has not been started since you turned off SSL, stopped the Build Forge engine, and modified the keystore passwords.

## About this task

The Apache Tomcat application server contains keystore passwords in the *server.xml* configuration file. They are stored as clear text. Apache Tomcat does not support encoded or encrypted passwords in this setting. In this procedure you modify *server.xml* and security properties in the Build Forge console.

## Procedure

1. Enter the new password in the Tomcat configuration. Edit *<bfinstall>/Apache/tomcat/conf/server.xml*. The Connector statement for SSL is located just under the comment *<!-- Define a SSL HTTP/1.1 Connector on port 8443 -->*.

```
<Connector port="8443" maxHttpHeaderSize="8192" algorithm="IbmX509"
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" disableUploadTimeout="true"
acceptCount="100" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS"
keystoreFile="C:\Program Files\IBM\Build Forge\keystore\buildForgeTrustStore.p12"
keystorePass="newpassword"
keystoreType="PKCS12"
truststoreFile="C:\Program Files\IBM\Build Forge\keystore\buildForgeTrustStore.p12"
truststorePass="newpassword"
truststoreType="PKCS12"/>
```

2. Start Build Forge.
3. Log in to the console. Use root or a login that has the Security access role.
4. Enter the new password for the keystores. In **Administration** → **Security** → **Keystore**, edit these entries to use *newpassword* in the Password property.
  - Default JSSE Key Store
  - Default JSSE Trust Store
  - Default OpenSSL Key Store

5. Enable SSL.
  - a. In the console, go to **Administration** → **Security**.
  - b. Set **SSL Enabled** to Yes.
  - c. Click **Save**.
6. Export the change to `BFClient.conf`. Click **Update Master BFClient.conf**.
7. Start the Build Forge engine.

### Creating a new self-signed certificate:

Use the tools provided to create a new self-signed certificate.

### Before you begin

You need the password that was specified for the keystore during installation. If you do not know what it is, look in `bfinstall/Apache/tomcat/conf/server.xml`. The `keystorePass` attribute on the `SSL/HTTP` connector contains the password.

### About this task

This procedure describes how to replace a certificate that was created automatically during a Build Forge installation. It creates a certificate with the following properties:

- Keystore: `buildForgeKeyStore.p12`
- Expiration: 15 years (set as 5475 days)
- Subject DN: `CN=hostname`, where *hostname* is the fully qualified hostname.

Use the `openssl` and `ibmjdk` tools to create the certificate. The tools are included with Build Forge software.

Five keystores are needed:

- `buildForgeKeyStore.p12` - keystore, container for certificates and keys
- `buildForgeTrustStore.p12` - truststore, container for certificates and keys
- `buildForgeKey.pem` - PEM keystore
- `buildForgeCert.pem` - public certificate
- `buildForgeCA.pem` - PEM Certificate Authority (CA)

**Note:** Line breaks are used for clarity in the example commands. Do not use them in the command. Enter it as one string or use the line-continuation character (^ for Windows, \ for UNIX or Linux).

**Important:** The same password is used for all keystores. It is shown as *password* in the examples.

### Procedure

1. Log on to the host where the Build Forge engine is installed.
2. Put the tool directories on your `PATH`.
  - `<bfinstall>/openssl`
  - `<bfinstall>/ibmjdk/bin`
3. Put the `openssl` directory on `LD_LIBRARY_PATH`.
  - `<bfinstall>/openssl`

4. Create the keystore buildForgeKeyStore.p12, certificate, and public-private key pair.
  - a. *In temporary directory*, run keytool to create the keystore:
 

```
keytool -genkey -alias buildforge
-keyalg RSA -keysize 1024 -validity 5475 -dname "CN=hostname"
-keystore buildForgeKeyStore.p12
-storepass password
-storetype pkcs12
```
  - b. Copy the keystore file (buildForgeKeyStore.p12) to <bfinstall>/keystore. It overwrites the existing file.
5. Export the public certificate. In the directory <bfinstall>/keystore, run this command:
 

```
keytool -export -alias buildforge
-file cert.der -keystore buildForgeKeyStore.p12
-storepass password
-storetype pkcs12
```
6. Create the truststore.
  - a. *In temporary directory*, run keytool to create the truststore:
 

```
cd /temp
keytool -import -noprompt -trustcacerts -alias buildforge
-file cert.der -keystore buildForgeTrustStore.p12
-storepass password
-storetype pkcs12
```
  - b. Copy the truststore file (buildForgeTrustStore.p12) to <bfinstall>/keystore. It overwrites the existing file.
7. Copy the trust
8. Put the public client certificate in buildForgeCert.pem In the directory <bfinstall>/keystore, run this command:
 

```
openssl pkcs12 -clcerts -nokeys
-in buildForgeKeyStore.p12 -passin pass:password
-out buildForgeCert.pem
```
9. Put the certificate and keys in buildForgeKey.pem In the directory <bfinstall>/keystore, run this command:
 

```
openssl pkcs12
-in buildForgeKeyStore.p12 -passin pass:password
-passout pass:password -out buildForgeKey.pem
```
10. Create the PEM Certificate Authority buildForgeCA.pem. It is a copy of buildForgeKey.pem. In the directory <bfinstall>/keystore, run this command:
 

```
cat buildForgeCert.pem > buildForgeCA.pem
```

### What to do next

The buildForgeKey.pem is a password protected PEM keystore. The Apache server prompts for the password during startup. If you do not want to be prompted for this password during startup, generate a PEM keystore that is not password-protected for the Apache server's use.

To remove the password from the private key, you can run the following step. Make sure the buildForgeKeyForApache.pem file is readable by those who need access to the ID of the process running Build Forge.

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

### Logging in with a client certificate:

You have two options when configuring the ability to log in with a client certificate.

*Logging in when using WebSphere Application Server to host Build Forge:*

#### **About this task**

When using WebSphere Application Server to host Build Forge, you have an option you can use with some modifications. This option requires WebSphereSSOInterceptor, which obtains the authenticated principal after WebSphere has performed the authentication.

Make the modifications in the following procedure to use this option.

#### **Procedure**

1. Protect the Build Forge WAR file so that the WAS container-managed authentication will authenticate requests to rbf-services. For information about how to set up this protection, see “Integrating with WAS security using a custom interceptor” on page 111.
2. Configure WebSphere Application Server to support client certificate authentication mapping for the rbf-services web application. Set up this support by configuring the following items:
  - SSL for a client certificate
  - Your rbf-services webapp for a client certificate
  - Your web server for a client certificate
  - The LDAP server mapping for client certificates

For information about configuring client certificate authentication for WebSphere Portal that can guide you in configuring WebSphere Application Server, see <http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/index.jsp?topic=/com.ibm.wp.ent.doc/wpf/certauth.html>. Follow steps 1 to 3. That procedure refers to the web.xml file that you modify in “Integrating with WAS security using a custom interceptor” on page 111. You modify the file located at `${WAS_INSTALL_ROOT}/profiles/${PROFILE_NAME}/installedApps/${CELL_NAME}/rbf-services_war.ear/rbf-services.war/WEB-INF/web.xml`.

*Logging in based on a custom Build Forge SSO interceptor:*

#### **About this task**

This option requires you to add a custom SSO interceptor, as explained in “About the single sign-on framework” on page 102.

#### **Procedure**

1. Ensure the custom SSO interceptor receives the X509 certificate from the request attribute as follows:

```
X509Certificate[] certs =
(X509Certificate[])request.getAttribute("javax.net.ssl.peer_certificates");
```
2. Ensure the custom SSO interceptor maps portions of the Subject DN to LDAP attributes. Typically, you map the Common Name (CN) from the certificate to the userid in the LDAP directory. Return the result in the `authenticateRequest` method of the Build Forge SSO interceptor.

```

responseAttributes.setStatus(HttpServletResponse.SC_OK);
principal = mapCert(certs); /* Custom method to map from certificate to
 Build Forge LDAP principal. */
result = new Result(Result.UseridOnlyOID, domain, principal); /* Specify
 the domain name of the LDAP server the principal exists in. */
return result;

```

## Enabling password encryption

Configuring the Build Forge system to use password encryption increases the security of the system.

It is also critical to enforce physical security to prevent unauthorized access to the system.

**Note:** If you are integrating with WebSphere components, be sure that the prerequisites for password encryption are met before configuring password encryption.

- For integration with WebSphere Application Server: see “Using WebSphere Application Server instead of Apache Tomcat” on page 479.
- For integration with IBM HTTP Server (IHS), see “Using IBM HTTP Server instead of Apache HTTP Server” on page 481.

### About password security in Build Forge

The Build Forge system uses encoded passwords by default but can use encrypted passwords for additional security.

When password encryption is enabled, it is enabled as a symmetric key password scheme. The same key must be used by both the client using a password and the service that is accessed.

- Build Forge engine and Build Forge agents
- Build Forge services layer and the database used by Build Forge

In the Build Forge system, keys are kept in a `bfpwcrypt.conf` file. The file is located in the installation directory of Build Forge (for the engine) and the agent.

Password encryption uses symmetric keys. All systems that need to decrypt a common database password need the same key. Also, all agents that receive encrypted passwords from an engine need the engine's key. If multiple engines are running (redundant configuration), the agent needs each engine's key.

For a simple installation of one Build Forge Management Console on one host and one agent on another host, enabling password encryption requires the following procedure:

- Enable password encryption on the Build Forge console (**Administration** → **Security**).
- Export the current key to a file. (This key is used by the agent and would also be used by other engines.)
- Generate a new key for the agent. Export it to a file. (This key is used by the agent to encrypt its keystore password.)
- Update the agent's `bfpwcrypt.conf` with both keys. Put the new key last.
- In the Build Forge console, enable password encryption for all server definitions that use the agent. (**Servers** panel)



- On the Build Forge host, use the `bfwencrypt` utility to encrypt the password that Build Forge uses to access the database. Replace the current password (encoded) with the encrypted password in `buildforge.conf`.
- Update the service layer's copy of `buildforge.conf`. See “Build Forge configuration file (`buildforge.conf`)” on page 95.

## Enabling password encryption for the Management Console

Enabling password encryption in the Management Console is a prerequisite for enabling password encryption in all other components:

- Redundant Management Consoles require export of the password key from the first Management Console to be included in their `bfwencrypt.conf` files. All Management Consoles must use the same key. Typically they must also use the key to encrypt the database password.
- Agents require the export of a password key if the engine is sending encrypted passwords to it. Both the engine and the agent should use the same key. The key exported from the Management Console allows them to decrypt encrypted Server Auth passwords.

If the agent is not receiving encrypted passwords, but needs to generate encrypted passwords for use in its `BFAgent.conf` file, then each agent should use a different password key. Generate each key individually from the Management Console.

To enable password encryption in the Management Console, do the following:

1. In the console, go to **Administration** → **Security**.
2. Set **Password Encryption Enabled** to Yes.
3. Click **Save**. This step saves the configuration in the Build Forge database.
4. Click **Update Master BFClient.conf**. This step saves the configuration in the Build Forge `bfclient.conf` file.
5. Restart Build Forge. This step is required so that the running Build Forge process uses the new settings in the `bfclient.conf` file.

When **Password Encryption Enabled** is Yes and in the configuration, you are able to do the following:

- Export the password key to a file
- Generate new password keys
- Run the `bfwencrypt` and `bfagent` commands to create encrypted passwords for inclusion in the console and agent configuration files

## Managing password keys

The Build Forge system uses encoded passwords by default but can use encrypted passwords for additional security.

### Password encryption key file:

The `bfwencrypt.conf` file contains password encryption configuration properties.

When Build Forge is started for the first time, it automatically generates a `bfwencrypt.conf` file in the same location as the `bfclient.conf` file.

- Windows: `<bfinstall>`
- UNIX or Linux: `<bfinstall>/Platform`

**Important:** Do not rename this file. It must always be named `bfwencrypt.conf`.

The file contains these properties:

**bfpwcrypt\_key\_alias**

Alias of an encryption key. The alias is part of any password encrypted with this key. The system uses it to determine which key to use. There can be multiple definitions of this property, one for each key. The last definition is used to encrypt passwords. All others are used to decrypt an encrypted password when it is read. An encrypted password can be encountered in a configuration file, a database, or in a communication between an agent and the engine.

**bfpwcrypt\_key**

The encrypted master key, encrypted using 128-bit AES encryption. This key is used to encrypt passwords.

**bfpwcrypt\_key\_password**

Password needed to decrypt bfpwcrypt\_key. This password is encoded.

Example bfpwcrypt.conf file:

```
***** Password Encryption Configuration Properties *****
bf_pwcrypt_key=MKuoiwD+MsWBFg1/2xeG0TEtpY+hAzXQu21fBcofM0M=
bf_pwcrypt_key_alias=8a679d430c401000b55e00007d1a7d1a
bf_pwcrypt_key_password=Tq0eDXc4G/bdaWeatKTYUx6Sw4S3i6wX
Creation date=Thu Nov 20 03:44:48 CST 2008
Origination host=myhost.mycompany.com
```

**Exporting the password key:**

Export the password key from the Build Forge console so that you can place it in other locations.

Export the password key used on the Build Forge engine to place it in the following locations:

- Other Build Forge consoles that access the same database
- Computers hosting agents that are identified in Server definitions that have password encryption enabled

To export the password key, do the following:

1. In the console, go to **Administration** → **Security**.

**Note:** **Password Encryption Enabled** must already be set to Yes and the setting saved and updated in the master BFClient.conf file.

2. Click **Export Key File**. You are asked where to save the file.
3. Specify a location, and then click **Save**.

Take the file to the hosts that need keys to decrypt passwords (other redundant management consoles, agents) or encrypt passwords (agents). Add the contents of the file (all three property settings) to the bfpwcrypt.conf file on that host. This table describes how the placement of the contents determines when the keys are used.

Placement in bfpwcrypt.conf file	Password key used in these cases
End	<ul style="list-style-type: none"><li>• Decrypt passwords that contain its alias</li><li>• Encrypt new passwords</li></ul>
Anywhere else	<ul style="list-style-type: none"><li>• Decrypt passwords that contain its alias</li></ul>

**Note:** When you generate a new key, an additional key is placed at the end of the `bfwpcrypt.conf` file. When you export a key, only the latest key is exported.

**Generating a new password key:** You can generate new keys from the Management Console to change the system-wide key. This should be done periodically to maintain good system security. The new key must be updated on all engines and agents.

Generating a new password key *adds* a new key to the `bfwpcrypt.conf` file:

- If user or console passwords have been generated or saved using the old key, they continue to work.
- **Important:** If a previous key has been exported and incorporated into an agent's `bfwpcrypt.conf` file, communications with the agent now fail until the newly generated key is added.
- Any new passwords saved in Build Forge or generated with `bfwpcrypt` use the new key.
- Exports of the key file export only the newest key.

To generate a new password key, do the following:

1. In the console, go to **Administration** → **Security**.

**Note:** **Password Encryption Enabled** must already be set to Yes and the setting saved and updated in the master `BFCClient.conf` file.

2. Click **Generate New Key**. You are asked to confirm.
3. Click **Yes**.

## Enabling password encryption for agents

Password encryption for agents is enabled in their configuration files.

To enable password encryption for an agent, do the following:

1. Stop the agent if it is running.
2. Go to the directory where the agent is installed.
3. Edit `bfagent.conf` and turn on the encryption setting:  

```
password_encrypt_module ./bfcrypt.dll;./bfwpcrypt.conf
```

If the agent is launched from a directory other than where it is installed, change this path to refer to the files directly.
4. Get the Management Console encryption key. Export the key to a file. This key is required for the agent to decrypt an encrypted Server Auth password. It will also be used to encrypt local keystore passwords.
5. Place the key in the agent's `bfwpcrypt.conf` file. Put the generated key for the agent last in the file. The last entry in the file is used when you encrypt passwords manually.
6. Start the agent.
7. In the Management Console, go to **Servers**. For every server definition that uses this agent, set the **Password Encryption Configuration** property to **Enabled**.
8. Click **Test Connection** to make sure that the connection works with the encrypted password.

## Encrypting passwords in `buildforge.conf` and `bfagent.conf`

Use an exported password key to build encrypted passwords for use in `buildforge.conf` and `bfagent.conf`.

The `buildforge.conf` file contains a username and password (`db_password`) that Build Forge uses to access the database. That password is normally encoded but can be encrypted. To encrypt a password for the Management Console, do the following:

1. Go to the Management Console root directory.
  - Windows: `<bfinstall>`
  - UNIX or Linux: `<bfinstall>/Platform`
2. Run the following command:  
`bfpwencrypt -e password`  
Use the plain-text password you want to encrypt for `password`.  
The encrypted password is sent to stdout.

The `bfagent.conf` file contains the password key (`ssl_key_password`) that the agent uses to access the keystore. That password is normally clear text but can be encrypted. To encrypt a password for the agent, do the following:

1. Go to the agent root directory.
2. Run the following command:  
`bfagent -e password`  
Use the plain-text password you want to encrypt for `password`. The encrypted password is sent to stdout.

**Note:** If you are using AIX and your GCC library is not in `/lib` or `/usr/lib`, you might get an error indicating "Cannot load module `/usr/local/bin/bfcrypt.dll`." You can address this issue by updating `inetd.conf`.

To correct the error, do the following:

- a. Find the following line:  
`bfagent stream tcp nowait root /usr/local/bin/bfagent bfagent`
- b. Change the line to the following line:  
`bfagent stream tcp nowait root /usr/bin/env env LIBPATH=path /usr/local/bin/bfagent`

The *path* is the location of your GCC library.

- c. Reload `inetd.conf` with the command `refresh -s inetd`.

An encrypted password starts with the string `bfcrypt:` and the password key alias enclosed in braces, followed by the password, which is encrypted (AES 128 bit) and then encoded (Base63). Examples of encoded and encrypted passwords:

Encoded:  
`dd8b42eed5cc051500f5bffe2b82b1aa6a67baee028a85d0cefa`

Encrypted:  
`{bfcrypt:7427ab360c4010008f9d000049664966}drAIT1zLDGX/xRcvw65+B8aFpTqvmAdbmnh6FpwkHjU=`

## Debugging problems with password encryption

If there are problems with encrypted passwords, these sections describe approaches to debugging them.

### Debugging password encryption problems in the console:

Services layer, the web interface, and engine share the same key file.

When all three components are installed on the same host, they use the same keyfile:

- Windows:  
`<bfinstall>/bfpwcrypt.conf`
- UNIX or Linux:  
`<bfinstall>/Platform/bfpwcrypt.conf`

Check the following issues if there are problems after you enable password encryption:

- Make sure you restarted Build Forge after enabling password encryption. Ensure that all processes stopped and restarted properly (Apache, Apache Tomcat, engine).
- Redundant consoles: if you have multiple installations of the management console using the same database, they must all use the same `bfpwcrypt.conf` file. The most secure method is to distribute it manually rather than over the network.
- Server definitions: if **Test Connection** fails in the console, be sure the key was exported and put in the `bfagent.conf` file correctly. To be sure it is a password problem, disable password encryption and try **Test Connection**.
- Login: if you cannot log in after enabling password encryption, make sure that Build Forge is using the correct `bfpwcrypt.conf` keys in both `bfclient.conf` and `buildforge.conf`. The `buildforge.conf` must be updated in the `<bfinstall>` directory and in the service layer's copy of it. See "Build Forge configuration file (`buildforge.conf`)" on page 95.

If all of those checks are done but the problem persists, try enabling trace and examining the output logs.

- Web interface (UI): set environment variable `BFDEBUG_SECURITY=1`.  
Web interface: output appears in files.
  - Windows:  
`<bfinstall>/Apache/logs/php_error.log`
  - UNIX or Linux:  
`<bfinstall>/server/apache/logs/php_error.log`
- Engine: start the engine in the foreground. In the installation directory, run `bfengine -d`. On UNIX or Linux you can pipe this to a file using `bfengine -d 2>&1 | tee out.txt`. On Windows, you can do the same if you obtain the tee utility.
- Services: do the following:
  1. Stop Build Forge.
  2. Open the log file in an editor.
    - Windows:  
`<bfinstall>/Apache/tomcat/common/classes/logging.properties`
    - UNIX or Linux:  
`<bfinstall>/server/apache/tomcat/common/classes/logging.properties`
  3. Add the following line to the end of the file.  
`com.buildforge.services.common.security.level=ALL`
  4. Start Build Forge.
  5. Inspect the output.
    - Windows:  
`<bfinstall>/Apache/tomcat/logs/catalina.out`
    - UNIX or Linux:  
`<bfinstall>/server/apache/tomcat/logs/catalina.out`

## Debugging password encryption problems in the agent:

Debugging agent communications involves agent and engine components.

Check the following issues if there are problems after you enable password encryption:

- Check `bfagent.conf`. The following line should be uncommented:  
`password_encrypt_module ./bfcrypt.dll;./bfpcrypt.conf`
- Check that `bfpcrypt.conf` is present in the directory where the agent is launched. It must contain at least one key entry.
- Check that the final entry in the engine's `bfpcrypt.conf` is present somewhere in the agent's `bfpcrypt.conf`.

If all of those checks are done but the problem persists, try enabling trace and examining the output logs. To enable trace, do the following:

1. Open `bfagent.conf`.
2. Uncomment the following line:  
`activity_log bfagent.log`  
You can specify another path instead of `bfagent.log`.

### Path issues with `bfcrypt.dll`

In `bfagent.conf`, the `password_encrypt_module` property must point to the correct path to `bfcrypt.dll`. Example:

```
password_encrypt_module /opt/buildforge/bfcrypt.dll
```

With trace turned on, a problem with this path is indicated by output like the following:

```
[8928] bfcryptloader.c : 208: Loading password encryption module.
[8928] bfcryptloader.c : 223: module: bfcrypt
[8928] bfcryptloader.c : 232: Loading module: C:/BuildForge71.181.Agent/bfcr
ypt.dll
[8928] bfcryptloader.c : 262: Failed loading DLL, error code = 0
```

A successful load produces output like the following:

```
[12248] bfpwdlocloader.c: 134: Looking for password locator: ssl_key_password_locator
[12248] bfpwdlocloader.c: 244: Looking for password for prop ssl_key_password from bfagent.conf.
[12248] bfcryptloader.c : 208: Loading password encryption module.
[12248] bfcryptloader.c : 223: module: bfcrypt
[12248] bfcryptloader.c : 232: Loading module: ./bfcrypt.dll
[12248] bfcryptloader.c : 269: Loading procedure bfcrypt_init.
```

### Failed password decryption

When a password fails to decrypt because of the wrong key or some other reason, the log contains a line like the following:

```
[4912] agent.c : 237: AUTH failed
```

If you are sure that the password is correct, you can further diagnose the problem. Enable debugging for the `bfcrypt.dll` module. To enable debugging, set the following environment variable:

```
BFDEBUG_SECURITY=1
```

It needs to be set globally if the agent is running as a service.

Debug output is placed in bfcrypt.txt in the directory where the agent is launched.

The following output indicates that the correct key is not in bfpwcrypt.conf on the agent:

```
load_keys_from_file: Parsed 1 key configurations.decrypt:
 Looking for key matching info: 922492fe0c4010008304c3670e1e0e1e, length=32
decrypt: Comparing against: 4d553f110c401000ac08000051f651f6, length=32
decrypt: Warning! No matching key found.
```

The following output indicates a correct key match:

```
load_keys_from_file: Parsed 2 key configurations.decrypt:
 Looking for key matching info: 922492fe0c4010008304c3670e1e0e1e, length=32
decrypt: Comparing against: 7427ab360c4010008f9d000049664966, length=32
decrypt: Comparing against: 922492fe0c4010008304c3670e1e0e1e, length=32
decrypt: Found match: 922492fe0c4010008304c3670e1e0e1e, length=32
```

## bfclient.conf Reference

The bfclient.conf file stores settings about Build Forge security. The file is in the Build Forge installation root directory.

The bfclient.conf file contains settings used to enable secure communications (SSL) and password encryption. It contains the following sections:

- Connection properties
- Login properties
- SSL properties used by both OpenSSL and JSSE
- SSL properties used only with OpenSSL
- SSL properties used only with JSSE
- Keystore properties (bf\_keystore\_\*)
- Cryptographic properties

Table 1. Connection Properties

Attribute name	Default	Possible Values	Required	Description
bf_services_hostname	Specified during installation	n/a	Yes	Hostname where the Build Forge services layer is located
bf_services_tcp_port	Specified during installation	n/a	Yes	TCP port for connecting to Build Forge services. It is used when SSL is not specified.
bf_services_ssl_port	Specified during installation	n/a	Yes	SSL port for connecting to BuildForge services securely
bf_services_preferred_protocol	tcp	ssl, tcp	Yes	For Perl or PHP clients, specifies SSL or TCP for making connections. For Java clients, the SecureAPIClientConnection object specifies SSL and APIClientConnection specifies TCP.

Table 2. Login Properties

Attribute name	Default	Possible Values	Required	Description
bf_login_user	None	UserID in Build Forge users list	No	Used as the login ID. The login ID can also be specified within a client program.



Table 2. Login Properties (continued)

Attribute name	Default	Possible Values	Required	Description
bf_login_password	None	Password for bf_login_user	Yes, if bf_login_user is used	Password for bf_login_user
bf_login_realm	None	LDAP domain name	No	LDAP domain to query if the user is not already in the User table.

Table 3. SSL properties used by both openSSL and JSSE

Attribute name	Default	Possible Values	Required	Description
bf_ssl_usage	None	jsse, openssl	Yes	Selects the SSL implementation. Depending on the selection, different properties are available.
bf_ssl_cipher_group	ALL	ALL, HIGH, MEDIUM, LOW	No	Specifies the group of ciphers to be provided during the SSL handshake. HIGH is the most secure, LOW gives the best performance, ALL is the most interoperable.
bf_ssl_cipher_override	None	Cypher suites that you provide	No	Overrides bf_ssl_cipher_group. Can be used to choose a smaller set of ciphers to use during the SSL handshake.
bf_ssl_protocol	TLSv1	TLSv1, SSLv3. Can vary with implementation.	No	Handshake protocol used by SSL. TLSv1 is the preferred protocol.
bf_ssl_cert_alias	None	Valid certificate alias in the configured keystore	No	Specifies the certificate to use. This is possible when multiple certificates are in the same keystore.

Table 4. SSL properties used only with openSSL

Attribute name	Default	Possible Values	Required	Description
bf_ssl_key_ref	openssl_key	Any valid PEM keystore reference which contains a private key	No	Reference to a keystore configuration that contains a private key for the client to use when connecting to a server. When used, you must also specify a valid certificate for this private key in bf_ssl_cert_ref. Used only when the server is set up to request personal certificates.
bf_ssl_cert_ref	openssl_cert	Any valid PEM keystore reference that contains a certificate for the specified private key	No	Reference to a keystore configuration that contains a certificate for the private key above. Used only when the server is set up to request personal certificates.
bf_ssl_ca_ref	openssl_ca	Any valid PEM keystore reference that contains one or more certificates used to validate the server certificates which this client is connecting to	Yes	Reference to a keystore configuration that contains one or more signer certificates used to validate server certificates during an SSL handshake. The certificates can be CA root, intermediate, or self-signed.



Table 5. SSL properties used only with JSSE

Attribute name	Default	Possible Values	Required	Description
bf_ssl_keystore_ref	jsse_keystore	Any valid PKCS12, JKS, or JCEKS keystore reference that contains a keyEntry (private key and certificate)	No	Reference to a keystore configuration that contains a personal certificate (private key and associated certificate) for the client to use when connecting to a server. This is necessary only when the server requests a personal certificate for client authentication.
bf_ssl_truststore_ref	jsse_truststore	Any valid PKCS12, JKS, or JCEKS keystore reference that contains a keyEntry (private key and certificate)	Yes	Reference to a keystore configuration that contains signer certificates used to validate server certificates during an SSL handshake. The keystore contains one or more trustedCertEntries, which are certificates used to validate other certificate signatures.

Table 6. Keystore properties

Attribute name	Default	Possible Values	Required	Description
bf_keystore_alias	Various	String	Yes	This is the name an SSL configuration uses to refer to the keystore configuration.
bf_keystore_location	Various	Relative or fully qualified path to a keystore of the type specified	Yes	This is the path and location to the keystore of the type specified. The path can be a relative path, but it must be correct with respect to the starting directory.
bf_keystore_type	PEM for openssl, PKCS12 for jsse	PEM for openssl, PKCS12, JCEKS, or JKS for "jsse"	Yes	The type of the keystore. Must match the actual keystore type referenced by the bf_keystore_location property.
bf_keystore_password	Specified during installation	A string supported by the keystore type. Some keystores do not support non-ASCII strings.	No	The password for accessing the keystore. For OpenSSL, the password is usually not required for Cert and CA keystores containing just public keys.

Table 7. Cryptographic Properties

Attribute name	Default	Possible Values	Required	Description
bf_pw_crypt_enabled	false	true, false	No	Specifies whether passwords are encoded (false) or encrypted (true). When enabled, the password encryption implementation uses a file called bfpwcrypt.conf located in the same directory as bfclient.conf.



---

## Chapter 11. Installing agents

This section describes how to install, run, configure and troubleshoot agents.

Install an agent on each host that you want to use as a server resource from the Management Console. An agent is a service that receives requests from the Management Console to run projects and steps.

---

### Installing the agent on Windows platforms

To install the agent on Windows platforms:

1. Locate and start the agent installation program in the installation media. The filename for the installation program is `win-bfagent-version.exe`.  
  
**Tip:** The Launch Pad starts this installation process when you choose **Rational Build Forge Agent Installation**.
2. If the installer detects an existing version of the agent, it prompts you to confirm to overwrite it. Click **OK**. Default: OK.
3. After the Welcome message opens, click **Next**.
4. If you agree to the terms of license agreement, click **I Agree**.
5. In the Choose Install Location window, set the **Destination Folder**, and then click **Next**. Use the default location `C:\Program Files\IBM\Build Forge\Agent` so that the file can be easily located.
6. In the Configuration window, choose the **Agent Options** that you want, and then click **Install**.
7. Select one of the following installation methods:
  - **Install as a service**
  - **Install User Mode Agent**  
Select a user mode agent only if the agent must be able to run a GUI application.
8. Optional: Click **Enable Cygwin Support**

**Tip:** If you use the Cygwin Linux<sup>®</sup> emulation environment, you can choose to install Cygwin support when you install the agent. If you install Cygwin support, do the following steps.

- a. During Cygwin installation, choose **DOS/text** line endings.
- b. In projects, use UNIX<sup>®</sup>-style syntax for commands.

**Important:**

Cygwin works only with US ASCII. It does not support UTF-8, so it cannot be used with any other systems.

9. Specify the **Port** that the agent uses to communicate with the Management Console. The default port is 5555.
10. At the Completing Setup panel, click **Finish**.

**Note:** Do not close any pop-up windows during installation. Allow them to appear and disappear while the installation runs.

## User mode agent

An agent installed as User Mode allows a user to interact with applications launched by a project.

User mode is an option offered for Windows agents only. The option is set during installation. It cannot be configured after installation. User mode has the following applications:

- Collect input manually through a GUI application as the job runs. This makes the job dependent on human input.
- Troubleshoot projects and steps. Output that is hidden during service mode operation is visible in user mode. Each step produces a console window while the step is running.

Keep in mind the following differences when setting up projects to use a computer in user mode:

- The user mode agent operates as the currently logged-in user on the system. The agent is only active during the time that this user is logged into the computer. A server running a user-mode agent cannot be used if the user is logged out.
- Steps run on a computer in user mode are visible to any users of the computer.
  - Each step opens a console window on the Windows computer where the agent is running. It displays command activity from the Management Console.
  - If you launch a GUI application from a step, the application window appears on the Windows computer where the agent is running. The Management Console waits until the application exits before continuing the step.
- Alternative: Use the start command if you want the job to continue without waiting.
- Do not use the `_USE_BFCREDS` variable. Any steps that use this variable will fail.
- The user must have the following privileges. They are typically not available by default. They must be added explicitly.

```
SeInteractiveLogonRight
SeAssignPrimaryTokenPrivilege
SeImpersonatePrivilege
SeIncreaseQuotaPrivilege
SeTcbPrivilege
```

## Performing a silent agent installation on Windows operating systems

To perform an automatic and silent installation of an agent on Windows, use the `/S` (uppercase S) option. For example, at a command prompt enter the following command. The option is case-sensitive:

```
win-bfagent-7.0.1.2305.exe /S
```

The silent installation uses the following settings. They cannot be modified.

- Overwrite existing install: yes
- Install location: `C:\Program Files\IBM\Build Forge\Agent`
- Install as a Service: yes
- Cygwin support: no
- Port: 5555

---

## Installing the agent on UNIX and Linux systems

Follow the installation instructions for your platform:

- **AIX**

1. Use the `aix5-bfagent-<version>.tar.gz` file or `aix5np-bfagent-<version>.tar.gz` file.

The `aix5np` file does not contain support for PAM authentication.

**Important:**

If you install the `aixnp` agent to run as root, at runtime the agent will use an AIX authentication call to authenticate, using the Server Authentication credentials that you specify. If you do not install the agent to run as root, then you must also use the `magic_login` setting in the `bfagent.conf` file to restrict access to it.

2. Extract the file by entering this command:

```
gzip -d gzipfilename.gz
```

3. Extract the file by entering this command:

```
tar xvf tarfilename.tar
```

4. Install the agent by entering this command:

```
cd extracted-agent-directory
./install.sh
```

**Important:** If an agent is compiled for AIX with the configure option `--without-pam`, authentication for the agent is turned off.

If it is then installed with root privilege, it allows people to connect as any valid user regardless of the password that they specify.

If you must compile an agent to run on an AIX system that does not use PAM, be sure to use a dedicated account for the agent, install it to run as that user, and use the `magic_login` setting in the `bfagent.conf` file to restrict access to it.

- **HP-UX**

1. Ensure the Rational® Build Forge® agent can locate all the operating system commands you specify in your projects. You can accomplish this task using either of the following options:

- Edit the system's `/etc/PATH` file to include all the required directories.
- Specify the `nologonshell` setting in the `bfagent.conf` file on the HP-UX system. This setting prevents Build Forge® from processing `/etc/profile`. You must then specify a value for `PATH` and any other required settings from `/etc/profile` in a Build Forge® environment that you apply to each project.

2. Get the `hpux11-bfagent-<version>.tar.gz` file from the installation media. Place it where you want the agent installed.

3. Extract the file.

```
gzip -d gzipfilename.gz
```

4. Extract from the tar file.

```
tar xvf tarfilename.tar
```

5. Install the agent.

```
cd extracted-agent-directory
./install.sh
```

6. Modify the following line in `/etc/profile` to enable the agent to run commands in a non-interactive login shell.

```
if [! $VUE]
```

Change the line to the following:

```
if [-z "$VUE" -a -n "$PS1"]
```

- **Mac OS**

1. Get the mac-bfagent-*<version>*.dmg file from the installation media and place it where you want it.
2. Double-click the file to extract its components.

- **Red Hat Linux**

1. Get the rh9-bfagent-*<version>*.rpm (Red Hat Enterprise Linux 4) or rhel5-bfagent-*<version>*.rpm file (Red Hat Enterprise Linux 5) from the installation media

2. Enter this command:

```
rpm -iUvh rh9-bfagent-<version>.rpm
```

- **Solaris**

1. Use the sol9-bfagent-*<version>*-sparc-opt.gz file for Solaris 9 or Solaris 10 on SPARC.

2. Extract the package:

```
gzip -d solN-bfagent-<version>-platform-opt.gz
```

3. At a command prompt, enter this command:

```
pkgadd -d ./unzipped-package
```

- **Other Platforms - Compiling from Source**

If you need an agent for another platform, use the src-bfagent-*<version>*.tar.gz file to compile the agent from the source:

1. Extract the downloaded tar file.
2. Run the configure script located in the src directory.

To configure for SSL, the option is:

```
./configure --with-ssl=/usr/include/openssl
```

To configure for both SSL and password encryption, the option is:

```
./configure --with-ssl=/usr/include/openssl --enable-bfcrypt-dll
```

3. Run the **make** command in the src directory.

The source pack requires the GNU C compiler or the C compiler for your system. The source pack and prebuilt agents that do not include an installer for the local computer include an installer to install the agent in the system's inetd/xinetd configuration files of the computer.

---

## Installing the agent on System i platforms

Use the following instructions to manually install the agent on System i.

The command script in step 7 creates a job description that runs at start up and starts the agent as the BFAAGENT user with \*ALLOBJ special authority.

- Any user with \*ALLOBJ special authority or the QSECOFR user can be authenticated by using the server authentication credentials that you specify in the Management Console.
- To authenticate a user without these privileges, you must configure the magic\_login setting in the bfagent.conf file. See the "bfagent.conf reference" on page 155 for details.

To install the agent on System i platforms:

1. Using the product installation media or the download image, locate the iseries-bfagent-<version>.tar.gz file.
2. Extract the tar file from the archive by entering this command:  
gzip -d iseries-bfagent-<version>.tar.gz
3. Extract the files from the tar file.  
tar xvf iseries-bfagent-<version>.tar
4. On the iSeries server, place the bfagent executable file in the agent installation directory, for example: /bin.
5. On the iSeries server, place the bfagent.conf file in /etc.
6. In the bfagent.conf file, uncomment the shell option and specify the default shell for PASE, as shown in the following example, or specify your preferred shell.  
shell /bin/sh
7. Configure System i to run as the BFAAGENT user at startup.  
Enter the following commands to create the BFAAGENT user with \*ALLOBJ special authority and create a job description that runs at startup as the BFAAGENT user. In the following example, the bfagent executable is installed in /bin.

CRTLIB BFAAGENT

CRTSBSD SBSD(BFAAGENT/BFAAGENT) POOLS((1 \*BASE)) TEXT('Build Forge Agent subsystem')

CRTJOBQ JOBQ(BFAAGENT/BFAJOBQ) TEXT('Build Forge Agent job queue')

CRTUSRPRF USRPRF(BFAAGENT) PASSWORD(\*NONE) INLMNU(\*SIGNOFF) LMTCPB(\*YES)  
SPCAUT(\*ALLOBJ) TEXT('Build Forge Agent user profile')

CRTJOBQ JOBQ(BFAAGENT/BFAJOBQ) JOBQ(BFAAGENT/BFAJOBQ)  
TEXT('Build Forge Agent autostart')USER(BFAAGENT) RQSDTA('CALL PGM(QP2SHELL)  
PARM('/bin/bfagent' ' ' -s'))

CRTCLS CLS(BFAAGENT/BFACLS) TEXT('Build Forge Agent job class')

ADDRTGE SBSD(BFAAGENT/BFAAGENT) SEQNBR(1) CMPVAL(\*ANY) PGM(QCMD) CLS(BFAAGENT/BFACLS)

ADDJOBQE SBSD(BFAAGENT/BFAAGENT) JOBQ(BFAAGENT/BFAJOBQ) MAXACT(\*NOMAX) SEQNBR(10)

ADDAJE SBSD(BFAAGENT/BFAAGENT) JOB(BFAAGENT) JOBQ(BFAAGENT/BFAJOBQ)

---

## Installing and running the agent on System z platforms

Follow these instructions to manually extract and compile the Build Forge agent source code on System z. The agent source code for z/OS is provided as uncompiled source only. A binary distribution is not available.

The following software and programs are required:

- The c89 compiler and Unix header files. On the z/OS system, the agent runs in the Unix System Services (USS) environment.
- The z/OS UNIX shell interface. During installation, you run all commands on z/OS in the z/OS UNIX shell.
- The gzip utility.

**Note:** If gzip is available on the z/OS system, you can extract the tar file on the z/OS system after transferring the source pack to z/OS. If it is not, you must first extract the files on a non-z/OS computer, and then transfer them to the z/OS system.

- The Build Forge agent source pack for z/OS: src-bfagent-*<version>*.tar.gz.

To install the agent on System z platforms:

1. Using either the product installation media or the download product image, locate the file for the agent source pack: src-bfagent-*<version>*.tar.gz.  
Copy or download the source pack to a directory on the non-z/OS computer.
2. At a shell prompt on the non-z/OS computer, extract the tar file from the agent source pack by entering the command:

```
gzip -d src-bfagent-<version>.tar.gz
```

3. Using ftp or another transfer method, transfer the tar file to the z/OS system as a binary image and place it in a dedicated HFS subdirectory, usually the USS home directory for a user account.
4. On the z/OS system, run the following commands to build the agent source code:

```
pax -rf src-bfagent-<version>.tar -ofrom=ISO8859-1,to=IBM-1047
cd bfagent-<version>/src
./configure-zos
```

After the ./configure-zos script completes, run the following command:

```
./build-zos
```

**Note:** If you receive errors after the ./build-zos script runs, see “Troubleshooting the agent installation on z/OS.”

5. On the z/OS system, place the bfagent.conf file in /etc.  
If bfagent.conf is not in /etc, the agent must be started with the -f option. See “bfagent reference” on page 152.
6. On the z/OS system, place the bfagent executable file in an appropriate location, for example, /usr/bin or /usr/local/bin.
7. On the z/OS system, run the following command as root:

```
extattr +p -s bfagent
```

8. On the z/OS system, log in as root and start the agent manually by using the -s option:

```
bfagent -s
```

If security policy does not allow you to log in as root, see “bfagent.conf reference” on page 155 and see the instructions for the magic\_login setting in bfagent.conf.

The agent runs as a standalone daemon and uses the default agent port 5555. To change default port, use the port setting in bfagent.conf. See “bfagent reference” on page 152.

**Note:** If the Unix TCP/IP daemon (inetd or xinetd) is installed and active on the z/OS system, you can set up the Build Forge agent to run as a service and start automatically. See “Running an agent on UNIX, Linux, and MacOS” on page 150.

9. On the z/OS system, use the telnet command to test the connection. See “Testing the connection” on page 163.

## Troubleshooting the agent installation on z/OS

You might receive error messages after building the agent source code on z/OS. This topic describes fixes for some common errors.



The configure-zos script sets some common values and performs some basic checks to identify the headers and functions available on your system.

Because of variations in z/OS system configurations, the ./configure-zos script might run without errors but you might see the following errors when you run the ./build-zos script.

**CEE3501S The module CCNDVR was not found.**

FSUM3066 The COMPILE step ended with the following return code:

-1: EDC5083I An error occurred attempting to load a module into storage.

This error indicates that a required dynamic library cannot be loaded by the compiler.

Run the command: % export STEPLIB="SYS1.SCCNCMP"

Rerun the ./build-zos command. If the command fails again, contact your system administrator for assistance in locating the required library.

**IKJ56228I DATA SET CEE.SCEE0BJ NOT IN CATALOG OR CATALOG CAN NOT BE ACCESSED**

FSUM3066 The COMPILE step ended with the following return code:

FSUM3052 The data definition name C8961 cannot be resolved. The data set was not found. Ensure that data set name CEE.SCEE0BJ is specified correctly.

This error indicates that the linker was unable to locate a system library that it needs to complete the compilation. Run the commands:

% export \_C89\_LSYSLIB=SYS1.SCEELKED:SYS1.SCEELKEX

% export \_C89\_PSYSLIB=SYS1.SCEE0BJ

Rerun the ./build-zos command. If the command fails again, contact your system administrator for assistance in locating the required libraries.

**IEW2456E 9207 SYMBOL xxx UNRESOLVED**

The unresolved symbol errors indicate that the build expected a symbol to be defined by your system C library that is not actually there. In most cases, this is a symbol that is often missing from other systems too, and there will be a setting in config.h to work around the problem.

For example, your system might not define the unsetenv function. The configure-zos script should normally detect this; if it does not, edit the config.h file that is provided with the agent source pack, as follows:

Change #define HAVE\_UNSETENV 1 to #undef HAVE\_UNSETENV.

Rerun the ./build-zos command to correct the problem.

**Note:** Similar #define statements exist for other functions.

---

## Running an agent

This section describes how to set up an agent to run. It is normally run as an auto-starting service or daemon.

## Running an agent on Windows

The agent is typically installed as a service and is set to Auto so that it starts when you turn on the computer. You must be logged on to the computer where the agent is installed to start and stop it.

To start and stop the agent, you can use the **Start** menu:

- To start the agent, click **Start** → **Programs** → **IBM Rational Build Forge** → **Start Agent Service**.
- To stop the agent, click **Start** → **Programs** → **IBM Rational Build Forge** → **Stop Agent Service**.

You can also use the following commands at a command prompt:

- `net start bfagent`
- `net stop bfagent`

## Running an agent on UNIX, Linux, and MacOS

The agent is intended to be run as a service and needs to be restarted automatically at system restart.

Add a bfagent entry to your configuration of inetd or xinetd as appropriate. The following example is the bfagent entry in xinetd.d on a Linux system, where the agent is installed in /usr/local/bin:

```
description: The IBM Rational Build Forge Agent serves build requests
from the IBM Rational Build Forge Management Console.
service bfagent
{
 disable = no
 flags = REUSE
 socket_type = stream
 wait = no
 user = root
 server = /usr/local/bin/bfagent
 log_on_failure += USERID
}
```

The agent can be run outside of the inetd/xinetd environment if necessary. To run it as a standalone daemon, use the `-s` option.

```
bfagent -s
```

When you use this option, the agent moves into the background and begins listening for connections. Place this command in a startup script so that the agent starts automatically when the computer is started.

## Running the agent on System i

Review the information in this topic if you plan to run the agent on a System i platform.

### Verifying that the agent port number is unique

Port 5555, which is the standard Build Forge agent port, might be preassigned to other agents on System i servers. In this case, change the Build Forge agent port to an unassigned port before starting the agent. To do this, edit the bfagent.conf file directly. For details, see “Changing the agent port” on page 153.

## Starting the agent manually

If you completed step 7 in the installation instructions, “Installing the agent on System i platforms” on page 146, the agent starts as the BAGENT user when System i starts.

Alternatively, you can start the agent on System i manually, by using the following command.

```
bfagent -s
```

**Note:** If the bfagent.conf file is not installed in /etc (the default location), use the -f option to specify the bfagent.conf location.

When you issue the bfagent command and start the agent manually, the agent starts as the user who starts the agent.

- If the QSECOFR user or a user with the \*ALLOBJ special authority starts the agent, the user is authenticated by using the server authentication that you specify in the Management Console.
- If another user starts the agent, authenticate the user by configuring the magic\_login setting in the bfagent.conf file. See the “bfagent.conf reference” on page 155.

## Verifying that the i5/OS PASE program is installed

The agent runs as an i5/OS Portable Application Solution Environment (PASE) program. PASE is included in i5/OS and enables AIX binaries and commands to be run. PASE is typically installed by default.

To determine whether the PASE program is installed, run DSPSFWRSC at a command line.

If the PASE program is not installed, load it from the installation CD.

## Using the agent in PASE

Most tasks necessary for building applications on i5/OS are accessible from the PASE environment. It is important to keep this fact in mind when planning and defining automation of processes targeted for the iSeries platform.

Commands in a step are interpreted by the PASE shell. You can also run native commands using the following syntax:

```
system -biOE "<native commands>"
```

**Important:** Each system command in a step runs its own process. This has implications for commands that work only within their own process.

For example, if you want to set library lists for a set of steps:

- You cannot use CHGSYSLIBL or ADDLIBL as step commands because they are native commands (not recognized by PASE).
- You cannot use the supported native command syntax (for example, system -biOE "ADDLIBL FLGHT400") in a step, because it changes the library list only for the command's own process. Subsequent commands and steps are not affected by the change.

Although you cannot set library lists for just a step, a set of steps, or a project, you can set them in the startup command script for the BFAGENT user. See the example startup script in “Installing the agent on System i platforms” on page 146. Setting library lists in the startup command script sets library lists for all projects and steps that are run in the example as the BFAGENT user. The user who runs the projects and steps must have access to the required libraries.

To set library lists, add a job description for the agent that lists the required libraries. The following example job description includes libraries FLGHT400 and FLGHT400M.

```
10 UTLIB
20 QGPL
30 QTEMP
40 FLGHT400
50 FLGHT400M
```

The agent specifies this job description in its startup routine. For example, if the job description is BFAJOB, the line in the system startup routine would be as follows:

```
ADDAJE SBSDB(BFAGENT/BFAGENT) JOB(BFAGENT) JOBD(BFAGENT/BFAJOB)
```

This solution affects all commands (from any step and project) that are run on the System i server associated with this agent.

## bfagent reference

The bfagent executable file starts the Build Forge agent. It reads its configuration from a BFAgent.conf file in the same directory.

The syntax for the command is:

```
bfagent [-f configfile | -s]
```

### Options

#### **-f configfile**

Run using the configuration file in configfile, rather than BFAgent.conf. This is a runtime option on UNIX or Linux. It is a debugging option when running the agent manually on Windows. It cannot be used for starting the service on Windows.

**-s** Start as a standalone service. You can use this option only on UNIX or Linux. Running this way is an alternative to starting bfagent with either the **inetd** or **xinetd**.

---

## Configuring the agent

This section describes how to configure the agent after installation.

### Locating the agent configuration file

The agent configuration file, BFAgent.conf, provides runtime configuration of the agent's operation. It contains comments that explain all of the possible options. The file is located in the agent installation directory:

- Windows default: C:\Program Files\IBM\Build Forge\Agent\BFAgent.conf
- UNIX and Linux default: /etc/bfagent.conf

**Important:** If you make changes to the BFAgent.conf file in the installation directory, you must repeat the changes after you subsequently reinstall or upgrade the agent. The configuration file is overwritten during every installation.

You can specify an alternate configuration file:

- On UNIX or Linux systems, you can preserve agent configurations by using a configuration file outside of the agent installation directory. When you do this, use the `-f` command line option on the command that starts the agent. Example:  
`bfagent -f /opt/bfagent.conf`
- On Windows systems, you cannot start the service with this option. The service can be used only when you run the agent manually. It is a tool for debugging.

## Changing the agent port

If the agent is being installed on a server where port 5555 is already occupied, the agent port can be changed after it is installed.

To change the port on Windows operating systems:

1. Open the registry editor: Click **Start** → **Run**, and then type `regedit`.
2. Go to `HKEY_LOCAL_MACHINE\SOFTWARE\BuildForge Agent`.
3. Change the value of `AgentPort` to the port number you want.

On UNIX, Linux, and Macintosh operating systems:

1. Open the `/etc/services` file.
2. Change the value of `BuildForge Agent Port` to the port number you want.

## Configuring a different shell

You can configure an agent to use a shell other than the default shell by editing parameters in the BFAgent.conf file.

For example, to change a Windows system to use the Korn shell provided by MKSTools, you can change the shell parameter with this command:

```
shell C:\MKSTools\mksnt\ksh.exe -L -c \"%s\"
```

The `%` in this command is replaced by the step command when the system sends a command to the server. In this case, use the backslash escape character to include quotation marks as literals in the command.

## Running agent commands on a Network Share File System (Windows)

The Build Forge agent initially starts with Windows system account credentials. To run commands, the agent later authenticates with Windows using Build Forge server authentication credentials.

The server authentication credentials are accepted for local commands but might fail for some commands that the agent must run on external, networked shared drives. For example, to modify files in a ClearCase dynamic view, the agent must access ClearCase files on a network shared drive.

The commands fail because the external file system ignores the agent server authentication credentials; it recognizes only the agent's initial system account credentials.

If you experience problems running commands on a network shared drive, try the following actions:

#### **Run commands using server authentication credentials**

To run commands using a Build Forge server authentication credentials with access to network shares, add the `win_reexec_after_auth` setting to the `BFagent.conf` file.

If you want to use Build Forge server authentication credentials to establish access to a network share, adding this setting is prerequisite.

The `win_reexec_after_auth` setting causes the agent to start a new process after authenticating with Windows. The new process forces the shared file system to recognize that the agent changed the user credentials.

When `win_reexec_after_auth` is set, the agent runs as a service and does not distinguish between commands that access network shares and those that do not, so you might notice a performance impact.

#### **Run the agent in single user mode**

During agent installation, set up the agent to run commands in single user mode without Build Forge server authentication credentials. Select the **Install User Mode Agent** option.

If the specified user is a member of the Administrator group, then the user's credentials must be specified using server authentication credentials.

If the user is not an administrator, then use the `magic_login` setting in `BFagent.conf` to prevent unauthorized access to the agent.

When you log on to the Management Console, the agent starts up and runs as the user name you provide, which immediately authorizes access to the network shares using that user's credentials.

#### **Run the agent as a service with a dedicated user account**

Set up the agent to run as a Windows service with a dedicated user account. This option restricts you to running the agent as a single user account, but does not require the agent to start a new process to re-authenticate, so performance is not affected.

To run the agent as a service with a dedicated user account:

1. On the Build Forge server, click **Administration Tools** → **Services** to open the Windows Control panel. The list of services opens.
2. Open the service for the IBM Rational Build Forge Agent.
3. Provide the user account information for the user you want to run agent commands. For example, provide information for the ClearCase admin user or other user with access to ClearCase dynamic views and VOBs.

## **Trigger variables and agent performance**

Two trigger variables can affect overall agent performance by decreasing the number of messages generated for the step log:

- `_SUPPRESS_ENV_OUTPUT`: if set, suppresses the printing of ENV messages on the step log.
- `_SUPPRESS_LOG_OUTPUT`: if set, suppresses the printing of nearly all messages on the step log.

See the Trigger Variables Reference in Working with Environments. Trigger variables have an effect on a job when they are included in a project environment or step environment.

## bfagent.conf reference

The bfagent.conf file stores settings for how the Build Forge agent runs. The file is in the same directory as the bfagent executable file.

The file lists all settings and internal defaults. Inactive settings are commented out.

### Settings

#### activity\_log *path*

Turns on activity logging. The information is appended to the file specified by *path*. The path must exist and the agent user must have write permission for it.

**Note:** The agent does not report an error if the path does not exist or if it cannot write to the file.

**Important:** There is no limit on file size. The file must be manually deleted. This setting is intended to be used temporarily for debugging the agent. It is not intended as a permanent log for a working agent.

#### allow IP-address-or-range [...]

Use this setting for these conditions only:

- Agents running on Windows
- Agents running in standalone mode on UNIX or Linux when the -s option on startup is used

This setting limits connections to the agent. Connections are allowed only from the IP addresses that match IP-address-or-range. By default connections are allowed from all addresses.

Specify one or both of the items:

- **IP Address:** A fully qualified IPv4 or IPv6 address. For example, for IPv4, 255.192.192.003. The specific IP address is allowed.
- **IP Address range:** A partially qualified IPv4 or IPv6 address. These examples are correct for IPv4, 192.168 or 192.168.63. All IP addresses that match this qualification are allowed.

**Note:** If you are running the agent on a superserver such as inetd or xinetd, use another method to control access. You may want to use a firewall, TCP wrappers (hosts.allow and hosts.deny), or the built-in filtering capability of xinetd.

**bind** This setting allows the user to specify an explicit bind address for the agent. This, together with the "port" setting, determines how the agent will listen for connections when it is started with the -s command-line option. The value given in the bfagent.conf file will force the agent to bind to the IPv4 localhost address; thus, the agent will only receive connections from a console that is on the same computer. Example: bind 255.192.192.003

**Note:** It has no effect on Windows or UNIX agents that are started by the system's service architecture, such as inetd, xinetd, or launchd.



**ccviewroot root-path**

This setting specifies the default view root for this host. See ClearCase documentation on init for more information. The internal defaults are as follows:

- Windows: ccviewroot M:
- UNIX or Linux: ccviewroot /view

**cc\_suppress\_server\_root**

If set, then the view path is the path set by ccviewroot. If not set, then the path set in the server definition is appended to the path set by ccviewroot. This setting does not need a value. If it is present in bfagent.conf, then it is set.

**command\_output\_cache size**

This setting causes the agent to cache output until it reaches the specified size in bytes. The internal default is not to cache. Using a cache can significantly improve agent performance and reduce network overhead. The cache size depends on how much output that commands produce.

Minimum value: 2048. A value of 2048 is used internally if the setting is less than that.

**cygwin**

This setting is used only with agents on Windows.

This setting enables the agent to work on a Windows host using Cygwin, a Linux-like environment. When using Cygwin, a number of Linux tools are available to the agent.

When you use this setting, you might need to set cygwin\_script\_magic and shell settings also. The example shows one way to configure these settings:

```
cygwin
shell C:\cygwin\bin\bash.exe --login -c "%s"
cygwin_script_magic #!/bin/bash
```

The shell setting must match your installation of Cygwin.

**cygwin\_script\_magic**

This setting is used only with agents on Windows when **cygwin** is set.

This setting specifies the `#!` line to use when executing steps. The default is `#!/bin/bash`.

**default\_logon\_domain**

Specifies the domain to use when an authentication request does not include a domain. If not specified, the agent computer's domain is used.

**disable\_telnet**

For best results, use telnet to test the agent connection.

For the agent, there is some built-in processing overhead associated with processing and correctly handling telnet control sequences.

Use this setting to disable the agent from handling special telnet character codes, thereby slightly improving performance. In product environments, use this setting to benefit from the improved performance.

**disable\_transcode**

Turns off processing that the agent performs to convert international data when the operating system is not using UTF-8 encoding. To avoid mixed encodings and data corruption, use UTF-8 for the agent operating system.



If the operating system does not use UTF-8 encoding, the agent must convert data to the correct encoding for the operating system's locale settings.

If your operating does not use UTF-8, use this setting for best results and improved performance of the agent.

#### **enable\_agent\_dll**

This setting enables DLL process tracing, which is a debugging tool.

#### **env\_recursion\_limit number-of-recursions**

Sets the variable-replacement recursion limit for pre-parsing. If not set, the limit is 32.

#### **extensions**

This setting specifies paths to external libraries of functions. The functions can be used as dot commands in a step. If this setting is not specified, external libraries are not loaded.

During parsing, the first token in the step command is taken as the function name. The second token is a string, and the third is an integer timeout value (in seconds).

Requirement: Dynamic loader support in the operating system. For example, in UNIX or Linux you need /usr/include/dlfcn.h. These defaults values are used internally.

- UNIX or Linux: /usr/local/bin/bfextensions.so
- Windows: C:\program files\ibm\build forge\agent\bfextensions.dll

#### **getaddrinfo\_using\_addrconfig**

This setting is used only for running the agent as a standalone service on UNIX or Linux operating systems (bfagent -s). This setting makes the agent use AI\_ADDRCONFIG when calling getaddrinfo() to select a listening interface. By default AI\_ADDRCONFIG is not used.

If you use this setting, the agent ignores interfaces that do not have a properly configured address. It listens only for interfaces that have a properly configured address.

#### **lang lang-code**

Use this setting only when the Management Console does not provide a valid language.

This setting specifies the language that the agent uses to write messages and command output. Typically it is not set explicitly because the agent uses the language that the Management Console specifies. However, setting the language can be useful if the desired locale is not available on the computer. The setting is also useful as a backup, in case the Management Console fails to communicate a language or communicates an invalid language.

The internal default is en, as if it were explicitly set as follows:

```
lang en
```

#### **leave\_tmp\_file**

Use this setting only while you are troubleshooting.

This setting causes the temporary file that is used to hold step commands to be retained, rather than deleted after command execution. In troubleshooting, the file can be compared to the steps as they are displayed in the Management Console.

**Note:** Do not use this setting for typical operations.

#### **locale locale-code.charset-code**

This setting is used only with UNIX and Linux operating systems. Windows handles locales differently.

This setting specifies the language and multibyte character set that localized applications use. This setting works by setting the LANG environment variable for the agent context.

To set up the agent to treat command output as US English UTF-8, use the UTF-8 locale for your operating system. For example, on Linux use the following representation.

```
locale en_US.UTF-8
```

To determine the correct representation of the UTF-8 locale for your operating system, run the **locale -a** command.

If this setting is not specified, the agent uses the locale of the operating system. This setting is a convenience. This setting is especially useful if the default locale of the operating system is not the locale that you want the agent to use. The setting is especially useful if changing the system locale to meet agent requirements is not practical.

#### **magic\_login user:encoded-password**

The agent typically uses administrative privileges such as root or admin to log on to the operating system. The magic\_login setting is an alternative to standard system authentication. With this setting, the system can authenticate your login with a single user name and password.

If the agent is run as the root or admin user, this setting is ignored and normal authentication is attempted.

The agent runs all commands using the permissions of the user who started the agent, not the user name used to log in.

This setting is used in only these situations:

- When running the agent with administrative privileges is not possible. For example, use this setting with UNIX systems that do not work with PAM.
- When running the agent with administrative privileges is not permissible because of security policies.

To configure a login for the agent:

1. Create a server authentication that uses a user name and password. In the Management Console, click **Servers** → **Server Auth**.
2. For this example the user name is build, and the password is MySecretPassword.
3. Create a server that uses the agent. Associate the server authentication with this server in the **Authentication** field.
4. Generate an encoded password for the agent. In the installation directory for the agent, run **bfagent -P** with the password that you choose.

An SMD5 hash-encoded password is returned, as follows:

```
bfagent -P "MySecretPassword"
eca0b7f2f4fbf110f7df570c70df844e1658744a4871934a
```

5. In BFAgent.conf, set magic\_login to use the desired user name and encoded password.

```
magic_login build:eca0b7f2f4fbf110f7df570c70df844e1658744a4871934a
```

6. Start the agent.
7. Test the server connection. In **Servers**, select the server, and then click **Test Server**.

**map** *drive-and-user-spec[; ...]*

This setting specifies a mapped drive. Some systems might require drive mappings. For example, a drive mapping might be required because a shell is run from a shared drive. Mappings specified on the agent are performed before mappings specified by `_MAP` environment variables in the Management Console. This example illustrates two drive mappings:

```
map X:=//host1/share;Z:=//host2/share(username,password)
```

**map\_drive\_is\_failure**

When specified, this setting causes a step to fail upon encountering an unmapped drive specification, before step execution. If this setting is not specified, steps ignore drive failures and attempt to run the step. In that case ensure that the failure generates a meaningful error message.

**no\_preparse\_command**

This setting disables the variable-expansion parsing that the agent typically performs on a command before passing the command to the shell. See also the `_NO_PREPARSE_COMMAND` environment variable, which can be used for an single project or step.

**no\_pty**

This setting is used only with agents that are running on UNIX or Linux systems.

This setting can be used to help prevent the system shell from locking up when the shell interacts with the pseudoterminal of the agent. This setting is typically used with HP-UX and z/OS. You can also use two other methods to help prevent this kind of lockup:

- Use an alternate shell
- Use the `nologonshell` setting

The **no\_pty** setting disables the pseudoterminal allocation.

**Note:** Using **no\_pty** affects some commands. For example, the `ls` command returns output in a single column rather than three columns. If you use this setting, test thoroughly before you deploy the job to a production environment.

**nologonshell**

Use this setting only with agents that are running on UNIX or Linux.

This setting causes the shell that the agent runs to be a normal shell, not a logon shell. This setting is often useful in these cases:

- The logon shells provide verbose output
- The logon shells change environment settings in unwanted ways
- The logon shells attempt to communicate interactively with the user

When set, standard methods of requesting that the shell be a normal shell rather than a logon shell are used. This may not work on all platforms and in such cases, the `shellflag` setting may be used to pass flags to the shell to modify its behavior.

Those behaviors are not desirable for the agent, because it runs as a user without being an interactive user.

**Note:** The Mac OS X 10.5 system uses `/bin/bash`, which does not respond to `nologonshell`. Use `shellflag -l`.

**Note:** The z/OS operating system always uses the `/etc/profile` script for both logon shells and non-logon shells. You might need to change the contents of the script or use another shell if its behavior does not work well with the agent.

See also the **shellflag** setting. Flags can be used to change logon script behavior.

**password\_encrypt\_module** *dll\_path;conf\_path*

Required to enable SSL on the agent. It specifies paths to a DLL and configuration file.

- *dll\_path* is the path to `bfcrypt.dll` (it is typically `./bfcrypt.dll`).
- *conf\_path* is the path to `bfpwcrypt.conf` (it is typically `./bfcrypt.conf`).

**port** *port-number-or-range [...]*

This setting is used only with agents that are running in standalone mode on UNIX or Linux when you issue the **-s** option on startup.

This setting specifies the port that the agent uses to listen for connections with the Management Console.

Specifies the port that the agent uses to listen for connections with the Management Console.

**Note:** The port is set to 5555 by default. For UNIX or Linux the setting is in `/etc/services`.

**shell** *shell\_name [options]*

This setting specifies the default shell. Internal defaults are as follows:

- Windows: `shell cmd.exe /q /c "%s"` unless the following settings are used:
  - If the `cygwin` setting is used, the default is `shell C:\cygwin\bin\bash.exe --login -c "%s"`
  - If the `cygwin` setting is not used, the default is `shell cmd.exe /u /q /c "%s"`
- UNIX or Linux: The shell set for the user account, or `/bin/sh` if the user's shell cannot be determined. Note that you cannot specify parameters in this setting, but you can use the `shellflag` setting to pass them. The agent automatically forces the default to be a logon shell by inserting a hyphen. For example, `/bin/ksh` is sent as `-ksh`. If `shell` is set explicitly, then `nologonshell` is set implicitly. See `nologonshell`.
- *System i*: Set the shell value to `/bin/sh`

You can override this setting from within a step. A step that starts with a line containing `#!` overrides the shell setting, and the `nologonshell` setting is used to run the step commands.

**shell\_compatible\_undef\_vars**

This setting forces the representation of undefined variables to be an empty string. If not set, the representation is the variable name for variables of format `$VAR`, `${VAR}`, or `%VAR%` and the empty string for `#[VAR]`.

**shellarg**

This setting is used only with agents that are running on UNIX or Linux.

Use this setting if it seems that commands are being scrambled. Some shells on Red Hat Linux Enterprise require this setting.

The setting changes the way a command script is passed to the shell. Normally the script is passed through standard input:

```
/bin/sh < /tmp/bfshellscript.sh
```

This setting causes scripts to be run by passing them as parameters:

```
/bin/sh /tmp/bfshellscript.sh
```

**shellflag** *flag*

This setting is used only with agents that are running on UNIX or Linux.

This setting adds a flag when a shell is running. Only one flag can be specified. It is typically used to disable **rc** script processing to reduce output or undesired processing. Examples:

- **csh** and derivatives: use **shellflag -f** to disable **rc** script processing.
- **bash**: use **shellflag --noprofile** to disable profile script processing.

**ssl\_ca\_location** *path*

Specifies the keystore file that contains the certificate authority. If the agent runs as a service, use an absolute path.

**ssl\_cert\_location** *path*

Specifies the keystore that contains the private certificate. If the agent runs as a service, use an absolute path.

**ssl\_client\_authentication** [**true** | **false**]

Set to **true** to require client authentication when a connection is made to the agent. If **true**, the Build Forge engine's certificate must be added to the agent's certificate authority keystore.

**ssl\_cipher\_group** [*grouplist* | **ALL**]

Specifies individual cipher groups to use. Can be set to **ALL**.

**ssl\_cipher\_override** *cyphers*

Overrides the cipher group. Specify the ciphers to use.

**ssl\_key\_location** *path*

Specifies the keystore file that contains the key. If the agent runs as a service, use an absolute path.

**ssl\_key\_password** *password*

Password for the key. This property is stored in clear text by default. You can configure the agent to encrypt this password using its own key or the Build Forge server's key.

**ssl\_protocol** *protocol*

The SSL handshake protocol to use, one of **SSL**, **SSLv2**, **SSLv3**, **SSL\_TLS**, **TLS**, **TLSv1**. The protocol must match the protocol used by the Build Forge server.

**update\_path** *path*

This setting identifies the full path to the Build Forge agent executable. The setting is established automatically during installation. The directory is a default directory for the operating system or the installation directory that you specify.

**Note:** This setting is ignored on Windows agents. The update path is taken from registry keys. The keys are set during agent installation.

#### **win\_reexec\_after\_auth**

Add this setting if you need to run agent commands on a network share file system using Build Forge server authentication credentials. For example, to modify files in a ClearCase dynamic view, the agent must access ClearCase files on a networked shared file system.

The Build Forge agent initially starts up with Windows system account credentials. To run commands, the agent later authenticates with Windows using Build Forge server authentication credentials.

Without this setting, the network share recognizes only the initial Windows system account credentials and ignores the subsequent server authentication credentials that are needed to access and write to files on the network share file system.

The win\_reexec\_after\_auth starts a new process after authenticating with Windows again by using the server authentication credentials and forces the shared file system to recognize the changed credentials.

When you use the win\_reexec\_after\_auth setting, the agent runs as a service and does not distinguish between commands that access network share files and those that do not, so you might notice a performance impact.

---

## **Troubleshooting agents**

This section describes procedures that you can use to troubleshoot agents that are not working correctly. Go through the procedures in order. If you are unable to get an agent working by using these procedures, then contact Support.

### **Testing host name resolution**

Verify that the agent host can be reached from the Management Console host. Use the ping utility from the Management Console host to test the agent host:

```
ping hostname
```

This example session runs on Windows on which both the Management Console and the agent are installed.

```
C:\> ping localhost
```

```
Pinging somehost.city.company.com [127.0.0.1] with 32 bytes of data:
```

```
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
```

```
Ping statistics for 127.0.0.1:
```

```
 Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
 Approximate round trip times in milli-seconds:
 Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

A message similar to the following one indicates a problem:

```
Unknown host
```

The problem lies in the network configuration of the Management Console host. Contact your network administrator.

## Testing the connection

You can test the connection to the agent by using telnet or using a test from the Management Console.

To test the connection from the command line:

1. Connect to the agent with a telnet command. If you are logged on to the host where the agent is running, you can use localhost as the host name.

```
telnet hostname 5555
```

This response indicates a successful connection:

```
200 HELLO - BuildForge Agent v7.0.1.buildnumber
```

2. Check authentication by issuing the following commands, using your login credentials:

```
telnet localhost 5555
username user name
password password
cmd ping
go
```

The following message indicates success:

```
AUTH: set user account to <user name>
```

If the above tests work but jobs are failing and a test of your server shows a user authentication error, check the pluggable authentication modules (PAM) configuration. If you see a message like the following, proceed to the next step.

```
AUTH: unable to set user account to <user name>: unknown account (1)
```

3. Type the following command:

```
cmd ping go
```

The following output of a telnet session is typical. In particular look for RESULT 0 at the end of the output as an indication of success. This test ran on an agent running on Windows.

```
300 DATA s 67
AUTH: Running as: [SYSTEM] in domain [NT AUTHORITY] SID Type [User]
300 DATA s 52
AUTH: Running with Privilege: [Lock pages in memory]
300 DATA s 66
AUTH: Running with Privilege: [Adjust memory quotas for a process]
300 DATA s 63
AUTH: Running with Privilege: [Create permanent shared objects]
300 DATA s 46
AUTH: Running with Privilege: [Debug programs]
300 DATA s 56
AUTH: Running with Privilege: [Bypass traverse checking]
300 DATA s 61
AUTH: Running with Privilege: [Back up files and directories]
300 DATA s 54
AUTH: Running with Privilege: [Change the system time]
300 DATA s 68
AUTH: Running with Privilege: [Remove computer from docking station]
300 DATA s 73
AUTH: Running with Privilege: [Impersonate a client after authentication]
300 HEARTBEAT 1
300 DATA s 16
PLAT: Windows XP
250 RESULT 0
PING: internal loopback test complete
260 EOR
```

To test the connection from the Management Console:



**Note:** Use this method only if a Server has been set up that uses the agent.

1. Go to **Servers**.
2. In the list of servers, click the one to test.
3. Click **Test Connection**.

After the test, the results are available in the **Test Results** tab.

A message similar to the following output indicates a problem.

Could not open a connection to host on port 5555

Either the Management Console is unable to connect to the host or the agent is not functioning.

## Troubleshooting an agent on Windows

To troubleshoot an agent on Windows, do the following:

1. Check for installed executable files. Verify that these files are present in the agent installation directory:

- `bfagent.exe`
- `bfdispatch.exe`

**Note:** Before you continue, determine whether you have a custom configuration. If you have a custom configuration, save `BFAgent.conf` outside of the installation directory, and then restore it after installation.

2. Reinstall the agent. You can solve most Windows agent problems by reinstalling the agent. It updates the executable files and restores registry keys.

## Troubleshooting an agent on UNIX, Linux, or MacOS

To troubleshoot an agent on UNIX, Linux, or MacOS, try these procedures:

- Run **bfagent** from a shell. The correct response is similar to this message:

```
200 HELLO - Build Forge Agent v7.0.1.122
```

If you receive a message similar to the example and there are shared library problems, you will receive messages regarding those problems. You can resolve most shared library problems by setting the path correctly.

- Check that the agent is listening. Use the following command (assuming that the port is the default, 5555):

```
telnet localhost 5555
```

A 200 HELLO response indicates that the agent is listening. If you do not get this response, check your systems network configuration. Verify that the **inetd** configuration is correct, or check with your Linux or UNIX system administrator.

- Check authentication. Issue the following commands, using your login credentials:

```
telnet localhost 5555
username <user name>
password <password>
cmd ping
go
```

A message similar to the following message indicates that the authentication is working correctly:



AUTH: set user account to <user name>

If the previous tests work but builds are failing and a test of your server shows a user authentication error, check the pluggable authentication modules (PAM) configuration. If you see a message similar to the following message, proceed to the next procedure.

AUTH: unable to set user account to *user name*: unknown account (1)

- Check the PAM configuration. Problems with the PAM configuration are common issues on AIX platforms. Depending on your operating system, PAM is configured in one of two ways: with a line in the `pam.conf` file or with a file in the `pam.d` directory.

**Tip:** Solaris 10 is an exception in the following procedure: Delete any lines that specify a module of `pam_dial_auth`, for example `pam_dial_auth.so.1`. Agent authentication does not work if that module is included.

1. Verify that `/etc/pam.conf` exists. If it does not, go to the instructions for `pam.d` later in this topic. If the file does exist, continue to the next step.
2. In the file, create an entry for `bfagent`.
3. Copy the lines for another application, for example, `sshd` or `login`, and then substitute `bfagent` for the `[application]` field.

`[application] [when] [mode] [module]`

The fields are as follows:

- `[application]` is the name of the application that needs to authenticate user
- `[when]` is the type of authentication request
- `[mode]` is the mode of the authentication request
- `[module]` is the authentication module to invoke. The following example shows entries copied from `login` to `bfagent`. Note that the module names may be different from system to system.

<code>bfagent auth requisite</code>	<code>pam_authtok_get.so.1</code>
<code>bfagent auth required</code>	<code>pam_dhkeys.so.1</code>
<code>bfagent auth required</code>	<code>pam_unix_cred.so.1</code>
<code>bfagent auth required</code>	<code>pam_unix_auth.so.1</code>

4. After you set up the PAM entries, try to log in again as described in step 3.
  5. For more information, see PAM documentation at <http://www.sun.com/software/solaris/pam>.
- To troubleshoot PAM which is configured in `pam.d`:
    1. Find the `/etc/pam.d` directory and note that it contains several files, each named for an application. Within each file, each line is formatted this way:  
`[when] [mode] [module]`
    2. Copying a file from another application, such as `sshd` or `login`, and rename it `bfagent`.
    3. After you set up the PAM entries, try to log in again as described in step 3.
    4. For more information, see PAM documentation at <http://www.sun.com/software/solaris/pam/>.



---

## Chapter 12. Post-installation tasks

This section describes tasks to perform after a successful installation.

It discusses the following topics:

- Starting and stopping the engine
- Setting up users
- Verifying the installation
- Troubleshooting common problems

---

### Starting and stopping the engine

The following sections describe how to start and stop the engine.

#### Starting and stopping the engine on Windows

On Windows:

- At **Start** → **Programs** → **IBM Rational Build Forge Management Console**, choose either
  - **Start Engine Service**
  - **Stop Engine Service**

Control Panel: You can also use the **Administrative Tools** > **Services** control panel to start or stop the **IBM Rational Build Forge Management Console** service.

Running in the Foreground: If you have any problems getting the engine to run, run it in the foreground so that you can see the status and error messages it generates: **Start** → **Programs** → **IBM Rational Build Forge Management Console** → **Start Engine (Foreground)**. As the Management Console runs, log output is shown in a console window. To stop the engine in this mode, enter Ctrl-C in the console window.

#### Starting and stopping the engine on UNIX or Linux systems

If you have an rc file installed, use these commands to start and stop the product:

```
$ /opt/buildforge/rc/buildforge start
$ /opt/buildforge/rc/buildforge stop
```

If you do not have an rc file installed, start the product using the following:

```
$ /<bfinstall>/Platform/buildforge &
```

Stop it by determining its process ID and then issuing a kill command:

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

---

## Setting up users

This section describes how to set up user accounts in a Build Forge<sup>®</sup> system.

The following topics are discussed:

- Root user
- Adding user accounts
- Read-only user for reports

### Root user

The root user, the user whose login name is "root", has special characteristics:

- **Created upon installation:** The root user is the only default user that the installation program creates. The default password is "root". (Change the password immediately after installation).
- **No license required:** The root user does not consume a user license. No matter how many users are logged in, you can always log in as the root user. (When someone logs in as root, another user already logged in as root is logged off.)
- **System time zone:** The root user's time zone is the default time zone of the Management Console. The time zone for other users, both in-system and LDAP users, is taken from the root user's time zone by default. Users can set their own time zone after logging in once. All times and logs reported in the system are expressed in the user's time zone.
- **All permissions:** The root user has all available permissions and can edit other users' properties. You cannot remove any access privileges from the root user. Although the root user is not a member of any access groups, the root user can view, edit, or use any data objects in the system.
- **Priority:** The root user is always a priority user.
- **Logging out current users:** The root user can log out users by clicking **Logout User**. (Click **Administration** → **Users** and then click the user's name.)

### 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.

Click **Administration** → **Users** to display a list of current users. A user panel is displayed after the list. The system displays the Name, Login, Email, Limit, Activity (elapsed time since last user activity), and Time Zone of each user.

- To edit a user, click the user's name and edit the properties in the user panel, and then click **Save**.
- To create a new user, specify properties in the user panel when no user is selected. If a user is selected, click **Add User** to clear the panel. Click **Save** when you are done editing user information.
- To log out a user, click the user's name and then click **Logout User**.
- To make a fixed license seat available, first log in as root. Click a user's name, and then click **Purge Seat**. The console removes the user from the list of IDs that count toward the set of fixed licenses. The user is also logged off if he or she is logged in. For fixed licenses, the console counts the number of users who have ever logged in. When the limit is reached, no new users can obtain licenses. Existing users must be deleted or purged so that another user can use a fixed license. Purging a seat does not delete the user from the console. If the user logs

back in, the fixed license count is increased. If you use **Purge Seat** on a floating-license user, the action has the same effect as clicking **Logout User**.

- To delete a user account, click the user's name and then click **Delete**.

If **Delete** is disabled, the user account owns a scheduled job. To delete a user account that has scheduled jobs, first delete the scheduled jobs.

The user record sets default properties for the way that the user can interact with the system and controls the user's login name, password, and password expiration. You can provide the data for a user record to the system through the Management Console or the data can be derived from an LDAP/Active Directory database.

**Note:** When you edit a user whose record is derived from an LDAP database, many of the fields on the User page are disabled. You change these properties in the source database.

To add a new user, click **Add User**, edit the panel, and then click **Update**.

When you display a user record, three tabs are available:

- **Details:** Displays the user properties that you can edit. The available properties are described later in this topic.
- **Current Groups:** Displays the access groups that the user is a member of, either directly or through a group that is a member of another group.
- **Change Groups:** Displays the groups that the user is a direct member of and allows you to add the user to groups or remove the user from them.

For each user, you can set the following properties on the **Details** tab:

**Name** Specify the display name and label for the user.

**Email** Specify the email address where the system can send email notifications to this user.

**Note:** emails are sent only to users explicitly selected for notification.

#### **User Name**

Specify the name that the user types to log on to the Management Console.

#### **Password**

Type the password that the user provides to log on to the Management Console. The field is not displayed for users who are logged on. Use this field to enter a new password or change one. Enter the same password in the **Verified** field.

**Limit** Specify the maximum number of jobs the user can launch in a day. When the limit is reached, the system displays messages that indicate that the user's run quota has been exceeded.

#### **Time Zone**

Specify the user's time zone. The system uses the time zone of the root user as the default time zone for all times posted.

By default, in-system users and LDAP users are assigned the same time zone as the root user. Users can edit their assigned time zones.

**Note:** When upgrading to Build Forge 7.1 from a previous version, you will need to manually reset the time zone of the root user.

#### **Verified**

Repeat the password to verify that you type it correctly.

**Priority Login**

Specify whether the user is a priority user. A priority user can always log in to the system; if there are no more available user licenses, the system logs out the user with the oldest session so that the priority user can log in. The root user is always a priority user.

**Date Format**

Sets the user's preferred date format.

**Language**

Sets the user's preferred language.

**Password Expires**

Specify that the user's password will expire. If this option is checked, then the user's password expires after a number of days have elapsed, as specified in the **Password Expiration Days** system setting.

**Uses screen reader**

Enable the interface to support screen reader features such as dynamic highlighting and focusing.

**Calendar Start Day of Week**

Select the day of the week that the Schedule calendar displays first. The default is Sunday.

---

## Verifying the installation

This section describes how to test an installed and configured Build Forge<sup>®</sup> system.

It discusses the following topics:

- Configuring servers
- Creating a test project
- Running projects

## Server authentication

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

A server authentication stores a login name and password as a single named object that you can associate with one or several servers. Use the Server Authentication page to create and edit server authentications.

## Configuring servers

To make best use of the system's dynamic server selection, you must set up a number of data objects, in a particular order. This topic outlines the minimum requirements for using servers.

1. Create server authentications.

Server authentications provide login names and passwords for servers. You can apply a server authentication to more than one server, so that you do not need to have unique logins for every server, and when you change a login, you can change it for a set of systems.

2. Create collectors for groups of servers.

Collectors both collect properties from servers and assign properties to them. You can use a server without a collector. You must select a server based on default properties like BF\_NAME.

3. Create selectors.

Consider creating the following types of selectors:

- Name-based: Create one selector for each server, which selects the server based on its host name. You can then choose servers by name.
- Operating-system-based: Create one selector for each type of operating system in your environment so that your projects can choose servers by operating system.
- Capacity-based: You can get more specific by choosing servers based on available RAM or hard-disk space.

4. Install an agent on each computer that you plan to use as a Build Forge server.

5. Create a server in the Management Console for each computer you plan to use with Build Forge.

6. Test servers.

Click **Test Connection** to test the connections to your servers. Review their manifests to make sure that they have the properties you expect.

## Creating a test project

To verify that the Build Forge® system is functioning properly, create and run a simple project as explained in “Creating a hello world project” on page 10.


## Running projects

There are several different ways to launch a project.

### Before you begin

This task assumes that you have already created a selector, server, and project.

### Procedure

- While viewing the list of projects, click the  icon in front of any project to start the project immediately. You cannot use this method if a project has no steps, or if it has any environment variables with the **Must Change On Project** action. Running a project this way uses its default values for selector, class, tags, and environment variables.
- While viewing the project's steps, click **Start Project**. This method displays the Start Project page for the project, where you can change project parameters, environment variable values, and select steps to exclude from the run:
  - Select new values for project parameters.
  - Edit the project tag variable values.
  - Edit the project environment variable values. If you want your changes saved as the new defaults for these variables, click the **Save Environment** check box.
  - Select the Jobs Steps tab to display the list of project steps. You can select individual steps to exclude them from this run only.

When you have made your choices, click **Execute** to start the project.

- Select **Jobs** → **Start** and then click the project name. As when you use **Start Project**, this method displays the Start Project page.

Job Details

Job Steps

Project Parameters

Project Environment

Save Environment ☐

Snapshot:

Base Snapshot

Selector:

My selector

Class:

Production

Tag Format:

BUILD\_\$B

Tag Example:

BUILD\_9

Project Tags

☐ Editable Tags

B

9

## Results

While a project is running, view the **Jobs** → **Running** page to check the project status.

To view job results, select **Jobs** → **Completed** to display the completed jobs. Click the Tag Name to access options for viewing job results.

## Troubleshooting common problems

### Upgrading an agent on Solaris requires running pkgrm command

Use the pkgrm BFAgent command to remove the existing Solaris Build Forge agent before running the pkgadd command.

### URL for 7.0 notification templates might not work in later versions

The notification template URL opens the Build Forge job report when you click the URL link in the notification email.

In versions 7.0.1 and 7.1, the URL in the notification templates changed; consequently, the URL might not work when you upgrade from 7.0 to a later version.

If you experience a link error, complete these steps to manually edit the notification templates:

1. Select **Project > Templates**.
2. Click the notification template name to display its properties on the Details tab.
3. In the Body field, locate the URL for the template. The URL should be similar to the one in the following example:

```
http://${CONSOLEHOST}:${CONSOLEPORT}/fullcontrol/index.php?mod=projectruns&action=edit&bfid=${PID}&bfid=${BID}&bfid=${UID}
```

4. Replace the following URL elements with the appropriate 7.0.1 and later URL elements:

URL elements	7.0.1 (and later) URL elements
projectruns	jobs
&amp;	&



URL elements	7.0.1 (and later) URL elements
action=edit	action=build.view
&bfid=\${PID}   &bfid=\${BID}   &bfid=\${UID}	&bf_id=\${BID}

## Product unresponsive

If Rational Build Forge becomes unresponsive, check the logs in  
`<bfinstall>/Apache/tomcat/logs/` (UNIX or Linux) or `<bfinstall>/Apache\`  
`tomcat\logs\` (Windows) for a message similar to the following one:

```
Services: 20075: CRRBF1381I: Established connection to Build Forge Services.
DBD::DB2::st execute failed: [IBM][CLI Driver] SQL30081N A communication error has been detected.
 Communication protocol being used: "TCP/IP".
 Communication API being used: "SOCKETS".
 Location where the error was detected: "XXX.XXX.XXX.XXX".
 Communication function detecting the error: "recv".
 Protocol specific error code(s): "131", "*", "0". SQLSTATE=08001
 August 17, 2010 7:18:14 AM EDT
Database: 20075: CRRBFEEEE: DBD::DB2::st execute failed:
 [IBM][CLI Driver] SQL30081N A communication error has been detected.
 Communication protocol being used: "TCP/IP".
 Communication API being used: "SOCKETS".
 Location where the error was detected: "XXX.XXX.XXX.XXX".
 Communication function detecting the error: "recv".
 Protocol specific error code(s): "131", "*", "0". SQLSTATE=08001

Database: 20075: CRRBF0551I: StackTrace from Process id [20075] called from [
 BuildForge::DB::db2(./PerlApp/BuildForge/DB/db2.pm:65)
 BuildForge::Utilities::SysParams(./PerlApp/BuildForge/Utilities/SysParams.pm:57)
 main(.buildforge.pl:305)
 main(.buildforge.pl:213)
]
Database: 20075: CRRBF0556I: Trying to call [execute] on a non-existent database handle
DBD::DB2::st fetchrow_hashref failed: no statement executing at /PerlApp/BuildForge/DB/db2.pm
 line 78, <$sock> line 1054.
20075: CRRBF0555E: Problem performing Database Operation [fetchrow_hashref] : DBD::DB2::st
 fetchrow_hashref failed: no statement executing at /PerlApp/BuildForge/DB/db2.pm
 line 78, <$sock> line 1054.

Database: 20075: CRRBFEEEE: DBD::DB2::st fetchrow_hashref failed: no statement executing
 at /PerlApp/BuildForge/DB/db2.pm line 78, <$sock> line 1054.

Database: 20075: CRRBF0551I: StackTrace from Process id [20075] called from [
 BuildForge::Utilities::SysParams(./PerlApp/BuildForge/Utilities/SysParams.pm:59)
 main(.buildforge.pl:305)
 main(.buildforge.pl:213)
]
[IBM][CLI Driver] CLI0106E Connection is closed. SQLSTATE=08003 at /PerlApp/BuildForge/DB/Handle.pm
 line 385, <$sock> line 1054.
Database: 20075: CRRBF0555E: Problem performing Database Operation [prepare] :
 query [SELECT * FROM bf_jobcount WHERE bf_engine_id=?]
DBD::DB2::db prepare failed: [IBM][CLI Driver] CLI0106E Connection is closed.
 SQLSTATE=08003 at /PerlApp/BuildForge/DB/Handle.pm line 385, <$sock> line 1054.

Database: 20075: CRRBFEEEE: DBD::DB2::db prepare failed: [IBM][CLI Driver] CLI0106E
 Connection is closed. SQLSTATE=08003 at /PerlApp/BuildForge/DB/Handle.pm
 line 385, <$sock> line 1054.
```

where XXX.XXX.XXX.XXX is the server hosting your database.

This message indicates that there is not a connection between Rational Build Forge and the database.

To re-establish the connection:

1. Shut down Rational Build Forge, as discussed in “Starting and stopping the engine” on page 216.
2. Make sure the database is running.
3. Start Rational Build Forge, as discussed in “Starting and stopping the engine” on page 216.

---

## Chapter 13. Upgrading from a previous version

The following topics describe how to upgrade components from a previous version to version 7.1.2:

- Upgrading Management Console
  - Upgrading a version 7.1.x console
  - Upgrading a version 7.0.2.x console
- Upgrading agents

---

### Upgrading a version 7.1 console

Use this section if you are upgrading from any 7.1.x version to the present version.

To upgrade, perform an update install.

**Note:** When upgrading a version 7.1 console, there are additional tasks to perform to enable performance enhancements in version 7.1.2. See the release page on [jazz.net](http://jazz.net) for details.

### Performing an update installation

#### Before you begin

If you have modified the Tomcat configuration file `httpd.conf`, note that you must reapply those modifications after performing the update. The update process overwrites this file.

#### Procedure

To perform an update installation:

1. Check requirements to determine whether your database, database client, and agent version are supported. If needed, upgrade the database or database client before upgrading the console. You can upgrade the agent after upgrading the console.
2. Stop Build Forge if it is running. Then, if you are running on Windows, check the installation directory for a file named `bfengine.pid`. If it exists, remove it. Installation Manager is not be able to perform the update if it exists. It is most likely to exist if you have run Management Console in the foreground.
3. Start IBM Installation Manager.
4. Before updating, you must set your repository URL to the updated repository. See “Specifying the repository URL” on page 84 for instructions.
5. In Installation Manager, click **Update**.
6. In Update Packages, select a package group for which to find updates, then click **Next**.
7. Select the updates or fixes to install. Select the 7.1.2 installation from the list, then click **Next**.
8. You are shown a list of versions to which you can update. Select the 7.1.2 installation from the list, and then click **Next**.
9. Select the *I accept the license terms* check box, and then click **Next**.

10. The Update Packages page is displayed. Your features are selected for you. Click **Next**.
11. Configure the database and click **Test connection** near the bottom of the screen to test the configuration. After the test passes, click **Next**.
12. On the following page, you are presented with the information you edited in your buildforge.conf file. You are asked the following:
  - a. *Do you want the installer to make the required database modifications?* Select this check box. It causes the installer to make the required changes to your schema. If this check box is not selected, you must run `bfschema -u` manually after installation to apply schema changes.
  - b. *Do you wish to start the console after upgrade?* Select this check box to have the console start automatically after installation is complete.
13. Click **Next**.
14. The features you have selected to install are listed.
15. Click **Update** to begin the installation.
16. When the update has been successfully performed, click **Finish**.
17. Close Installation Manager. Start Build Forge.

**Note:** If you are upgrading from version 7.1.1.x, you will not see step logs from your builds until the Character Large Object (CLOB) migration completes. When you start Build Forge, the migration begins. When the migration completes, Build Forge notifies you using the message Log CLOB migration complete in the log accessed through **Administration** → **Messages**.

---

## Upgrading a version 7.0.2.x console

Use this section if you are upgrading a version 7.0.2.x system to the present version.

To upgrade:

1. Check requirements to determine whether your database, database client, and agent version are supported. If needed, upgrade the database or database client before upgrading the console. You can upgrade the agent after upgrading the console.
2. Upgrade from version 7.0.2.x to version 7.1.1.3.

Download the version 7.1.1.3 packages from the IBM support site, and then follow the instructions in the *IBM Rational Build Forge version 7.1.1.3 Installation Guide*. Due to database schema changes, upgrading from version 7.0.2.x to version 7.1.1.3 requires you to migrate configuration data (projects) and optionally, historical data (logs from jobs).

**Important:** Not all items are migrated, in particular customized notification templates (items in **Projects** → **Templates**)

3. Perform an update install on version 7.1.1.3 to update it to the present version.

**Note:** When upgrading a version 7.1 console, there are additional tasks to perform to enable performance enhancements in version 7.1.2. See the release page on [jazz.net](http://jazz.net) for details.

---

## Upgrading Agents

Upgrade agents by installing new agents over the old ones.

You upgrade an agent by installing a new agent on top of the old one. Follow the installation instructions for installing an agent.



---

## Chapter 14. Uninstalling product components

Use IBM Installation Manager to uninstall product components that you installed using Installation Manager. To uninstall the Build Forge Agent software, use the operating system tools and commands described in this section.

---

### Using Installation Manager to uninstall the product

Use this scenario to uninstall Build Forge product components that you installed with Installation Manager, except the provided DB2 Express database.

To uninstall product components, complete the following steps:

1. Log in to the operating system using the same user account that you used to install the product package.

Close any running programs that you installed with Installation Manager.

2. If the Build Forge engine and service are running, stop them as follows:

Windows	<ul style="list-style-type: none"><li>• <b>If the engine is running in foreground:</b> Press <b>Ctrl + c</b> to stop the engine and Build Forge services. <b>Important:</b> The most reliable method of stopping the engine and services is to use Ctrl + c. If the engine or any Build Forge services are still running when you uninstall, the uninstall will fail.</li><li>• <b>If the engine is running in background:</b> Open a command window and enter the following command to stop the engine and services: <pre>net stop bfengine71</pre></li></ul> <p>These commands stops the Build Forge engine, Apache HTTP server, and Apache Tomcat application server.</p>
UNIX/Linux	<ol style="list-style-type: none"><li>1. In a command window, go to the Build Forge rc directory. <pre>cd /opt/buildforge/rc</pre></li><li>2. Stop the engine. <pre>./buildforge stop</pre></li></ol> <p>This command stops the Build Forge engine, Apache HTTP server, and Apache Tomcat application server.</p>

3. Start IBM Installation Manager.
4. On the Start page click **Uninstall**.
5. On the Uninstall Packages page, from the Installation Packages list, select the product package that you want to uninstall and click **Next**.
6. On the Summary page, review the list of packages that will be uninstalled and click **Uninstall**. The Complete page is displayed after the packages are removed.
7. Click **Finish**.
8. Close the Installation Manager window and exit Installation Manager.  
You must exit Installation Manager before cleaning up Build Forge files, optionally removing the DB2 Express database, or reinstalling the product.

## Manually uninstalling the product if Installation Manager install fails

If you start IBM Installation Manager before the Build Forge engine and services have stopped, the uninstall fails.

To manually uninstall product components after Installation Manager fails:

1. In a command window, change to the Manager directory in the Build Forge installation directory, for example:

Windows	C:\Program Files\IBM\Build Forge\Manager
UNIX/Linux	/opt/buildforge/Manager

2. At the command prompt, type `main.exe uninstall main.res`.
3. When the command finishes running, delete the Build Forge installation directory.
4. Reinstall the product using Installation Manager.

## Post-uninstallation cleanup of Build Forge files

Some manual cleanup is needed after an uninstall.

After you uninstall product components through Installation Manager, manually delete the following directories. If these directories are present, you cannot reinstall Build Forge using Installation Manager.

**Important:** Deleting the Build Forge root directory also deletes the rafw product directory. Rational Automation Framework for WebSphere users should review the uninstall instructions in the Rational Automation Framework for WebSphere Information Center before deleting the Build Forge installation directory.

Directory	Default location
Build Forge installation directory	<b>Windows:</b> C:\Program Files\IBM\Build Forge <b>UNIX/Linux:</b> /opt/buildforge
Shared files directory	<b>Windows:</b> C:\Program Files\IBM\SDPShared <b>UNIX/Linux:</b> /opt/IBM/SDPShared

---

## Post-installation removal of the provided DB2 Express database

Optionally remove the provided DB2 Express database. Removing the DB2 Express database deletes the Build Forge database schema and tables.

Remove the DB2 Express database to:

- Reinstall the product and use a database other than DB2 Express as your Build Forge database
- Reinstall the product and reinstall a new, empty DB2 Express database

To remove DB2 Express, do the following:

1. Open the Windows Add/Remove Programs tool, locate IBM DB2 Express Edition in the list of programs and remove it.



2. Delete the DB2 installation directory. The default installation location is C:\DB2.
3. Delete the DB2 library directory. The default installation location is C:\Program Files\IBM\SQLLIB.

---

## Using Installation Manager to reinstall the product and use an existing DB2 Express

You can reinstall the product and use the existing DB2 Express database, if you did not remove it when you uninstalled other product components. This scenario gives you access to your existing Build Forge projects and job logs.

Before you start Installation Manager, use the check list in the following steps to obtain the required DB2 Express information.

To reinstall and use an existing DB2 Express database:

1. Start Installation Manager.
2. On the Start page click **Install**.
3. Follow the instructions in the Installation Manager wizard to reinstall product components.
4. On the Database Configuration page, make the following selections:
  - a. For *Install DB2 Express*, select **No**.
  - b. For *Do you wish to populate this database at install time?*, select **No**.
  - c. To complete the remaining fields, use the information in the following table:

### Check list: DB2 Express configuration information

✓	Field	Description
	Database host	The provided DB2 Express database is installed on the local host name (127.0.0.1).
	Database name	The database name is BUILD, all uppercase.
	Database schema name	The name of the database schema is BUILD, all uppercase.
	Database user name	The database user name that you provided to create the DB2 Express database.
	Database password	The password that you provided for the database user name.
	Path to the DB2 client libraries	The path to the DB2 client libraries (db2cli.dll) for DB2 Express is C:\Program Files\IBM\SQLLIB\bin.
	JDBC driver location	The path to the JDBC driver (db2jcc.jar) for DB2 Express is C:\Program Files\IBM\SQLLIB\java.

5. Click **Test Connection**.
6. Click **Next** to continue and complete the installation.

---

## Uninstalling the Windows Build Forge Agent

Use the Add or Remove Programs tool for Windows to remove the Build Forge agent software.

**Note:** If you are reinstalling, you can overwrite the same version of the agent software.

1. In Windows, locate Add or Remove Programs. For example, select **Start → All Programs → Control Panel → Add or Remove Programs**.
2. Locate the IBM Rational Build Forge Agent in the list of currently installed programs.
3. Click **Change/Remove**.
4. Follow the instructions to uninstall the agent.

---

## Uninstalling a UNIX or Linux Build Forge agent

Use the following instructions to uninstall the agent software from UNIX or Linux platforms.

### Linux agents

To remove agent software that was installed using the rpm package:

1. Find the agent software and list package names and versions:  

```
rpm -qa | grep bfaagent
```
2. Delete the agent software:  

```
rpm -e bfaagent-<version_number>
```

### Solaris agents

To remove agent software that was installed using the pkgadd program, run the following command:

```
pkgrm BFAgent
```

### Other agents

For other platforms, the uninstall process is manual and varies by platform. Follow the instructions that apply to your platform and super server implementation.

**Note:** To run most commands, you need root access and the /sbin and /usr/sbin directories must be set in your current PATH environment variable.

1. Remove the agent service daemon, bfaagent. Use the instructions for the super server implementation (inetd, xinetd, launchd, or SMF) that applies to your platform.

Super server	Procedure
inetd, common on older UNIX systems	<ol style="list-style-type: none"><li>1. Edit the /etc/inetd.conf file and remove the line for the bfaagent.</li><li>2. Find the process ID for inetd. <pre>ps -ef   grep [i]netd</pre> For BSD-derived systems, such as FreeBSD and Mac OS/X 10.4 and earlier, substitute ps auwwwx for ps -ef.</li><li>3. Read the updated inetd.conf and start inetd. <pre>kill -HUP &lt;PID&gt;</pre></li></ol>

Super server	Procedure
xinetd, common on newer UNIX systems	<ol style="list-style-type: none"> <li>1. To remove the agent service, run the following command: rm /etc/xinetd.d/bfagent</li> <li>2. Find the process ID for inetd. ps -ef   grep [i]netd For BSD-derived systems, such as FreeBSD and Mac OS/X 10.4 and earlier, substitute ps auwwwx for ps -ef.</li> <li>3. Read the updated inetd.conf and start inetd. kill -HUP &lt;PID&gt;</li> </ol>
launchd for Mac OS/X and OpenBSD systems	<ol style="list-style-type: none"> <li>1. Run launchctl.</li> <li>2. Enter stop com.ibm.rational.bfagent.</li> <li>3. Enter the following command: unload /Library/LaunchDaemons/com.ibm.rational.bfagent.plist</li> <li>4. Enter quit.</li> <li>5. Run the following command: rm Library/LaunchDaemons/com.ibm.rational.bfagent.plist</li> </ol>
Solaris System Management Facility (SMF) for Solaris 10	<ol style="list-style-type: none"> <li>1. Run inetadm -d network /bfagent/tcp</li> <li>2. Run svccfg delete -f network/bfagent/tcp</li> </ol>

2. Remove the agent service from the PAM interface.
  - a. Edit /etc/pam.conf and remove all lines that begin with bfagent.
  - b. Run rm /etc/pam.d/bfagent
3. Remove the protocol entry from the etc/services file.  
Edit /etc/services and remove the line for bfagent.
4. Remove the following files installed by the agent:

```

/etc/bfagent.conf
/etc/bfagent.conf-example
/usr/local/bin/bfagent
/usr/local/bin/bfcrypt.dll

```



---

## Chapter 15. Administration

This topic describes administrative operations for the Build Forge® system.

---

### About administration

Use the Administration panel to manage configurations and preferences.

Within the Administration panel you can work with users, security privileges, notification settings, and system settings to configure your Management Console.

To access the Administration panel, in the left menu, click **Administration**.

UI Config Console Reports Logout: Root User

System Help ?

Filter Showing 1 - 72 of 72 Auto Paginate << < Page 1 of 1 > >>

Name	Value
Alert Email Limiting	10/30
Apply inlined steps container environment	No
Apply server environment last	No
Auto-Logoff Minutes	0
AutoClean Audit Log Days	365

Alert Email Limiting Save Revert to the Default

Details

Alert Email Limiting: 10/30

Enter the value for:the maximum number of alert email messages per time in minutes (messages/minutes), 0/0 for unlimited.

Default Value: 10/30

---

### Access groups

An access group is a collection of users that the system uses to control permissions.

Use the **Administration** → **Access Groups** panel to create new access groups, to add or remove users, and to modify group properties. The panel displays a list of existing access groups. Click the name of a group to select it and display its properties in the lower portion of the panel.

[UI Config](#) | [Console](#) | [Reports](#) | [Logout](#) | Root User

---

**Access Groups** | [Add Group](#) | [Help](#)

[Filter](#) | Showing 1 - 4 of 6 | [Display All](#) | Page 1 of 2

Access Group Name	New User Default	LDAP Group DN's
<a href="#">Build Engineer</a>	No	
<a href="#">Developer</a>	Yes	*
<a href="#">Guest</a>	Yes	*
<a href="#">Operator</a>	Yes	*

---

**New Group** | [Save](#) | [Copy](#) | [Delete](#)

[Details](#) | [Users](#) | [Subgroups](#) | [Permissions](#)

☐ Default | Name:  | Owner:

LDAP Group DN's:

- To create a new group, click **Add Group** to clear the fields in the lower portion of the panel (if needed). Then, give the group a **Name** and select an **Owner** group. (The **Owner** group controls 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 DN's 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 DN's 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, and 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 and 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, and 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.

**Note:** LDAP users who have "Map Access Group Mapping" set to "yes" will not show up in the **Users** tab. This behavior ensures that the "Map Access Group Mapping" setting actually performs the group mapping without being contradicted by Build Forge manual settings.

- To nest groups, 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, and then click the **Subgroups** tab in the lower portion of the panel. Select the groups you want to make into subgroups, and then 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, and 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. You control the privileges of users through the groups you assign them to. You assign privileges to groups, and 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 configure the system to send notification messages, the target of the messages must be an access group. See "Setting up notification" on page 291.

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 instead of the database for user authentication. When you use LDAP, instead of defining users in the system, you allow some or all of the users from your LDAP database to access the system. You can also map access groups to LDAP groups. For details about setting up LDAP, see "About LDAP integration" on page 194.

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.

**Note:** A user who is not a member of a server or project's Access Group does not see that object listed on the Server or Project list pages. A user who is not a member of a step's Access Group can see the step in the list for the project, but cannot edit or run it; if the user runs the project that contains the step, the system skips the steps the user does not have access to.

- 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 that you want to allow to run jobs that are relevant to their work, without cluttering their view of the system with other jobs. However, you do not want to allow the programmers to edit the jobs. To create this scenario:

1. Create an access group for this role in your organization (for example, DeviceDriverDevs).
2. Assign the new access group as the **Access** property of all the projects you want the users to be able to run.
3. 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.
4. Assign the permission Execute Builds to the group.
5. Add 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. When you change the Access property of the projects, users who are not members of the DeviceDriverDevs group lose the ability to view, run, 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.

## 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

A user sets access properties based on the user's access groups. A user must be a member of an access group to assign data objects such as projects, serves, or steps to that access group.

For example, 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.

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 run 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 panel below it. The system displays the Name, Login, Email, Limit, Activity (elapsed time since last user activity), and Time Zone of each user.

- To create a new user, start entering properties in the user panel when no user is selected. If a user is selected, click **Add User** to clear the panel. Click **Save** when you are done editing user information.
- To edit a user, click the user's name and edit the properties in the user panel, and then click **Save**.
- To log out a user, click the user's name and then click **Logout User**.

**Note:** If using a flexible licensing scheme, the user interface will then show the user as (Not logged in); if using a fixed licensing scheme, this will only reset the user's logged-in time to 0:00:00.

- To log in as a user without using their password, first log in as root. Click the user's name, and then click **Switch to User**.
- To free up a fixed license seat, first log in as root. Click the user's name, and 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.

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.
- **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.

The screenshot shows the 'Details' tab of a user management interface. At the top, there is a row of buttons: (New User), Save, Copy, Switch To User, Expire Password, Logout User, Purge Seat, and Delete. Below these buttons are three tabs: Details, Current Groups, and Change Groups. The 'Details' tab is selected and contains the following fields:

- User name: [text input]
- Email: [text input]
- Name: [text input]
- Time Zone: [US/Central (dropdown)]
- Date Format: [1/1/03 2:30 PM (dropdown)]
- Uses screen reader: [No (dropdown)]
- Password: [text input]
- Verified: [text input]
- Language: [English US (dropdown)]
- Calendar Start Day Of Week: [Sunday (dropdown)]
- Limit: [Unlimited (dropdown)]
- Priority Login: [No (dropdown)]
- Password Expires: [Yes (dropdown)]
- Truncation: [Default Value (dropdown)]
- Searching: [Case Sensitive (dropdown)]
- Step Log Initial View: [First Page (dropdown)]

For each user, you can set these properties on the **Details** tab:

**Name** The display name and label for the user.

**Email** The email address where the system can send email 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.

**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**

Controls how many characters display 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.

**Searching**

Determines how Filter fields throughout the product perform searches. Values are as follows:

- Case Sensitive (the default) - the search considers the case of each letter.
- Case Insensitive - the search does not consider case.

**Step Log Initial View**

Specifies how the step log is positioned for viewing when it is opened in job results. If set to First Page (the default), the view is positioned at the first page of the log. If set to Last Page, then the view is positioned at the last page of the log.

## 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.
- **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**.

UI Config Console Reports Logout: Root User

Permissions Help ?

Filter Showing 1 - 4 of 105 Display All << < Page 1 of 27 > >>

Name	Group
<a href="#">Add Build Notes</a>	Projects
<a href="#">Add Child Access Groups</a>	Access
<a href="#">Add New Access Levels</a>	Access
<a href="#">Add New Classes</a>	Classes

**Add Build Notes**

**Details**

Name: Add Build Notes Group: Projects Select names in the list below and click Add to add them to the list of members on the right.

Guest	Add >>	Build Engineer
Operator		Developer
	<< Remove	Security
		System Manager

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 portion of the panel. 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.

**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. By using LDAP, users can 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 integration allows you to manage groups in LDAP and have user permissions updated automatically when a user logs in.

You can still manually create and maintain users 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 the Build Forge user interface.

### 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.

**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. If you enable LDAP group mapping after users have logged in, delete the users from the Build Forge Users list and have them log on again. The users' membership to Build Forge access groups is then based on the LDAP group mapping, rather than any manual changes you have made.

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.

**Note:** If you configure more than one domain, individual unique user IDs must be unique across domains. The system allows only one login per unique user. If one user logs in and then another user logs in using the same unique user ID, the first user session is closed. See “Accessing and using the console” on page 7 for more information about user sessions.

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 the **User Name**, **Password**, **Login**, **Confirm**, and **Email** fields disabled, because that information is provided by LDAP.

**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.

## 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.

3. Edit the values for any of the fields, and 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 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 DN**s 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 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:

```
ldapservers.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 LDAPS if you use LDAP over SSL (LDAPS). Additional setup is required for this option. See Enabling secure LDAP (LDAPS).

#### **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 of 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%

## **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
  - 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.
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

To discontinue authentication with LDAP or Active Directory:

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 panel for the setting.




**Note:** For system settings that take numeric values, the Management Console accepts any value that consists of one or more integers (0 through 9). Numeric-grouping characters, such as commas (,), decimals (.), and other noninteger separators, are not supported.

The panel 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
Alert Email Limiting	Sets the maximum number of alert emails the system sends over the specified number of 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.
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 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. Because 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.
Console Port	Port number that the web server uses to listen for Build Forge requests.

Setting	Description
Console URL	<p>URL that the web server uses to listen for Build Forge requests.</p> <p>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 <code>&lt;protocol&gt;://&lt;hostname&gt;[:&lt;port&gt;]</code>. Example: <code>http://myHost:81</code>.</p>
Create Missing Paths	Default: No. If yes, creates paths for projects if the path is not already present.
Database Size Threshold	<p>Threshold of database size at which the console will send a notice. Default: 2G.</p> <p>Note that UI performance slows down as the threshold is approached. Performing a database cleanup or increasing the threshold size restores performance.</p>
Database Size Threshold Notification	User name or Notification Group to which email is sent if Database Size Threshold is reached. If a user name is used, the address in the Email field for the user is used. See Administration > Users.
Default _AGE	Sets the interval, in seconds, between updates of Run Command properties for server manifests. You can also set this interval by defining a variable with the name _AGE in a server's collector. The smaller interval takes precedence.
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.
Disable Authentication for XML Feeds	Determines whether RSS data feeds are authenticated.
Enable Quickstart	<p>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 a 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: .</p> <p>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.</p>
Hard Run Limit	Default: No. If Yes, the system launches a scheduled build of a project if its launch does not violate the project's <b>Run Limit</b> setting. If No, the system ignores the project's <b>Run Limit</b> setting for scheduled builds.
Import Default Secure Access Group	Specifies a default access group for imported projects when the <b>Import with Secure Access</b> setting is set to Y.

Setting	Description
Import Insecure Default Access Group	Specifies a default access group for imported projects when the <b>Import with Secure Access</b> setting is set to N. The default group is used only when the import file lacks an access group.
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.
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. <b>Note:</b> 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 331 ).
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 for the Management Console” on page 34.
Link Debug Mode	When set to Yes, jobs that have adaptor links defined for them run a test of the link instead of running the associated project. The job output contains a single step, which contains output from the adaptor. The data is useful when you are troubleshooting your adaptor interfaces. <b>Note:</b> You can set debugging for an individual adaptor link by setting the State for the adaptor link to Debug. The state takes precedence over the Link Debug Mode setting.
Link Manual Jobs	Determines whether the system runs adaptors through adaptor links when you quickstart a project manually rather than run it from the scheduler.  The link check may produce additional output in the BOM for the job.  If set to N, the link is not checked or run when the job runs.

Setting	Description
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 <b>Run Queue Size</b> 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.
Max Simultaneous Purges	Controls how many purges can run simultaneously. You can purge as many builds as you like, but no more builds than the value in Max Simultaneous Purges will be simultaneously deleted. Default: 20
Maximum Refreshes	Maximum number of times that a page refreshes automatically. Default: 50
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.

Setting	Description
Password Format	<p>Specifies the requirements for user passwords using a format string of six fields separated by periods:  <code>length.char_types.upper.lower.numeric.special</code></p> <p>Here is an example: 5.2.u1.l1.n1.s1. This example is explained below.</p> <p>The first two fields specify the following:</p> <ul style="list-style-type: none"> <li>• Minimum password length (characters)</li> <li>• Minimum number of character types to use (an integer ranging from 1 to 4) in the four remaining fields</li> </ul> <p>The remaining fields specify a character type and frequency requirement. Each field includes a type and a number.</p> <ul style="list-style-type: none"> <li>• Type: one of u (uppercase), l (lowercase), n (number), or s (special). Uppercase (U, L, N, S) indicates that the character is required. Lower case (u, l, n, s) indicates that the character is optional.</li> <li>• 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 <i>if any are used</i>.</li> </ul> <p>The types are as follows:</p> <ul style="list-style-type: none"> <li>• U or u to indicate uppercase characters. This includes all characters that are considered a letter in their respective locales, but are not lowercase. Specifically, this includes characters that are uppercase, title-case, or any letter in single-case languages (such as Chinese).</li> <li>• L or l to indicate lowercase characters. This includes all characters that are considered lower case in their respective locales.</li> <li>• N or n to indicate numeric characters. This includes any character that is considered a digit in its respective locale.</li> <li>• S or s to indicate special characters. Any character that does not fit in the previous three categories. It includes all characters that are neither a letter nor a digit.</li> </ul> <p>Example: the string 5.2.u1.l1.n1.s1 indicates the following password requirements:</p> <ul style="list-style-type: none"> <li>• At least 5 characters long</li> <li>• Must include characters from a minimum of two of the four categories (uppercase, lowercase, numeric, special).</li> <li>• For each type, one character of the type qualifies as a match to count toward the requirement.</li> </ul> <p>Passwords such as abC1x and Abc2% would qualify.</p>
Pause Build Forge Engine	<p>When set to Y, the system completes any current jobs and then pauses the engine. Set it to N to return to normal operation.</p>



Setting	Description
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.
QuickReport Public dir	<p>The file system location of the public report designs.</p> <p>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.</p> <p>In earlier releases, the default file location (<code>../../reports/public</code>) is relative to the application server installation directory, for example: <code>&lt;bfinstall&gt;/Apache/tomcat/webapps/quickReport</code>.</p>
QuickReport Temp dir	<p>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.</p> <p>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.</p>
QuickReport Users dir	<p>The file system location of the private report designs.</p> <p>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.</p> <p>In earlier releases, the default file location (<code>../../reports/users</code>) for private reports is relative to the application server installation directory, for example: <code>&lt;bfinstall&gt;/Apache/tomcat/webapps/quickReport</code>.</p>
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 Adaptor 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”, and then wait one minute. The system resets the templates and then sets the value back to No.
Reset Server Job-Count	<p>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.)</p> <p>After BF_JOBS has been reset for all servers, the Reset Server Job-Count value reverts back to No (the default).</p>
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 Chain Links	Controls whether a launched chain project also launches any attached adaptor links.

Setting	Description
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.
Server Env Before Chain	Determines whether the system sets a step's server environment before (Y) or after (N) setting the chaining project's project environment within the step. The variables in the second environment processed override those in the first environment. The default value is Y, indicating the chaining project's environment is processed second and overrides the step's server 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.

Setting	Description
Server Test Frequency	<p>Used together with the number of enabled servers to determine how frequently to test and refresh the manifest data for servers. The default is 120 minutes (2 hours). A value of 0 means do not check servers.</p> <p>During these checks, the system contacts all enabled servers to verify that:</p> <ul style="list-style-type: none"> <li>• The servers are still reachable</li> <li>• The login information for the servers is correct</li> <li>• The manifest data for the servers is current</li> </ul> <p>Server tests are performed at a minimum of one server per minute. That rate increases if the number of servers is much larger than this setting.</p> <p>The system distributes the testing evenly over the interval. Examples:</p> <ul style="list-style-type: none"> <li>• You have 120 servers and the interval is set to 120 minutes. The system attempts to test one server per minute (120 servers / 120 minutes).</li> <li>• You have 12,000 servers and the interval is set to 1200 minutes. The system attempts to test 10 servers per minute.</li> <li>• You have 10 servers and the interval is set to 120 minutes. One server per minute is checked. A server will be checked as many as 12 times during the refresh interval in this scenario.</li> </ul> <p>Manual server tests started from the console take precedence over these automated tests.</p> <p>The complexity of a server's collector can affect throughput. A collector that performs many manual commands to collect data can require more than a minute to complete.</p>
Server Usage Connect Timeout	<p>Sets the number of seconds after creating an agent connection that the Management Console waits for the connection to open before failing the step.</p> <p>Also sets the timeout value for an existing connection to an agent. Agents are designed to contact the Management Console every 15 seconds. If no contact is made during a timeout period, the agent may have stopped or there might be network communication issues. The step fails if this value is exceeded.</p>
Server Wait Time	<p>Sets the number of seconds between checks to determine whether a server has become available.</p>
Services layer authentication servlet URL	<p>When set, will override the programmatically constructed URL to the services layer authentication servlet.</p> <p>If you are using an alias or a non-default port, then this setting will need to be updated using the following format: <code>http://server:port/rbf-services/AuthServlet</code></p>

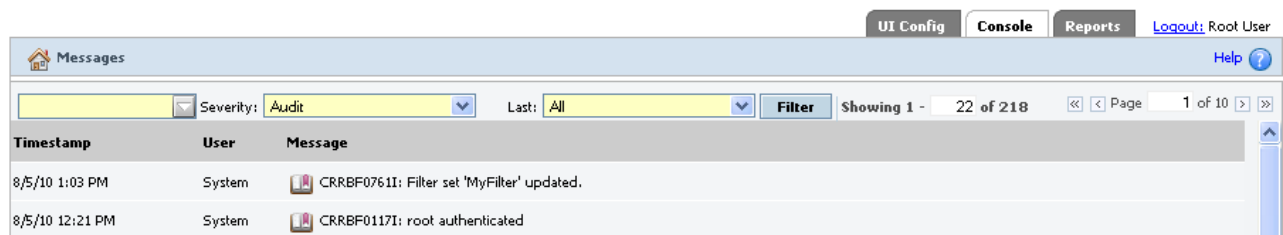
Setting	Description
SMTP Server	Sets the computer to use as an SMTP server when sending email notifications.  The default setting is localhost.
SSO Remote User	Default: No. If Yes, allows single sign-on remote user connections via standard web server authentication.
Stack BuildForge Env Variables	The 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 <i>stacks</i> 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 Max Retries	Controls how many times a step attempts to connect to an agent if the first attempt fails. The step fails if it does not connect in the specified number of retries.
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 and _USE_BFCREDS_DOMAIN special variables (which apply 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 and _USE_BFCREDS_DOMAIN special variables to use the user's credentials when logging into a server.
System Alert Email	The system sends alert email messages to the address defined by this setting.  The default is root@localhost.
System Alert Source	When the system sends alert email 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.
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

Setting	Description
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.
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.
Terminate Threads	Determines whether, when a threaded step fails, all other active thread blocks in the same project are stopped.

## 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 filter the list by using one or more of the provided controls, and then clicking **Filter**:

- Text box: enter a string to search for.
- Severity: select one of the following:
  - All (default)
  - Error
  - Info
  - Warning
  - Audit
- Last: select one of the following:
  - All (default)
  - 12 hours
  - 24 hours
  - 2 days
  - 3 days
  - 4 days
  - 5 days

The list update time depends on the number of messages and the filter criteria. Narrower filters update the list faster. Note that searching for strings on all severities and all time periods takes time to process.

For each message, three fields are displayed:

- Timestamp
- User
- Message

In **Administration** → **System** settings are provided to control the size of the logs:

- AutoClean Audit Log Days
- AutoClean Error Log Days
- AutoClean Info Log Days
- AutoClean Warning Log Days

For more information, see “System configuration settings” on page 200.

## Translated messages

Non-English messages are translated dynamically.

When you are using Build Forge in a language other than en\_US, messages are translated dynamically. Select **Administration** → **Messages** to display the list of messages. Messages are translated in batches of 250 every 15 seconds.

If a message has not been translated yet, the Message field displays Message Not Yet Translated.

**Important:** If a new user whose language is different from the current setting is added, that language is then used for messages. Existing messages are not shown in the previous language or the language of the new user.

## Subscribing to the 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.

## 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.

To view the Security panel, select **Administration** → **Security**.

UI Config Console Reports Logout: Root User

Security Save Update Master BFClient.conf Export Key File New Key File Generate Key

Access: Security

SSL Enabled: No Password Encryption Enabled: No FIPS Enabled: No

**Important:** Only some of the setup is done in the Security panel. For additional information, see “Security features” on page 101.

## 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.
- 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 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.
- **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:*

After 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 which users can edit or delete this keystore.

#### **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 which users can edit or delete this keystore.

**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 certificate.

**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 “Implementing single sign-on” on page 102.

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 which 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 computer, and you log in from another computer 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.

## 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 panel 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 wish to temporarily postpone the processing of new jobs, you should pause the engine.

**Note:** If you want to act directly on the database, for example, to back it up or restore it, then you should stop all running Build Forge services and processes first. Pausing the engine is not sufficient in this case and results in inconsistencies and/or corruption.

To pause the engine, select **Administration** → **System** and locate the **Pause Build Forge Engine** property (use the **Filter** box). The system displays a panel 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

The following sections describe how to start and stop the engine on Windows<sup>®</sup>, Linux<sup>®</sup> 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**

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, use these commands to start and stop the product:

```
$ /opt/buildforge/rc/buildforge start
$ /opt/buildforge/rc/buildforge stop
```

If you do not have an rc file installed, start the product using the following:

```
$ /<bfinstall>/Platform/buildforge &
```

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 computer.

---

## **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.

### **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 216.

## 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<sup>®</sup> or UNIX<sup>®</sup> 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<sup>®</sup> database. If you want to upgrade your license, contact support.

---

## Chapter 16. Servers

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.

- 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.
- A *selector* reads server properties from the manifest. Projects can use selectors to determine which server runs a step.

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.

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.

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 run 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

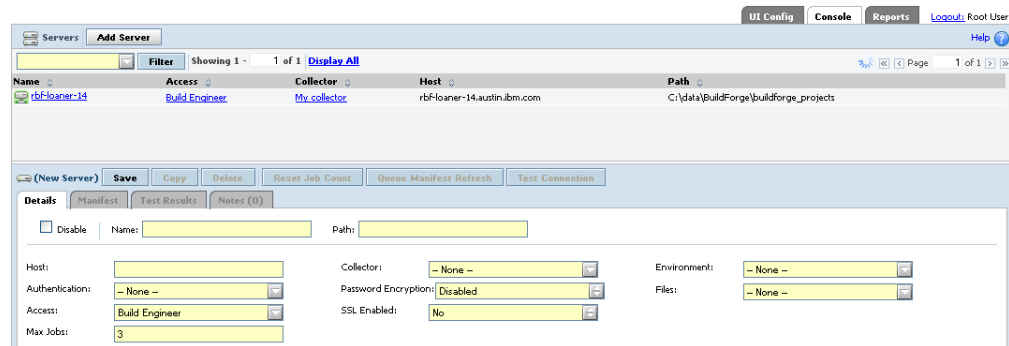
performed using one server from work performed with another because all output is saved in different directories.

- Two logical servers with different environments run their steps with different starting environment values.

## About the Servers panel

Use the Servers panel to specify servers on which to run projects.

In the Management Console, to manage servers, click **Servers** in the left menu.



### Details tab

The Details tab provides the following fields:

**Host** The name of a host that is running the Rational Build Forge agent

#### Collector

The collector to use with the server

#### Environment

The environment to use with the server

#### Authentication

The server authentication (login and password) to use

#### Password Encryption

Determines whether the agent password is encrypted

**Files** The types of file transfers to use with the .get and .put commands

#### Access

The access group of users who can use the server

#### SSL Enabled

Indicates whether you have configured the Rational Build Forge system to use SSL

#### Max Jobs

The most jobs that should run at the same time

### Manifest tab

The Manifest tab shows various server properties. It provides a default set of properties. You can include additional properties by using a collector.

### Test Results tab

The Test Results tab indicates the status of the last test of the connection between the Management Console and the server.



## Notes tab

The Notes tab provides a convenient location for storing and sharing comments about a server.

---

## Creating a server

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

To create a server:

1. Select **Servers** in the left menu. The system displays the New Server panel at the bottom of the main content panel. If you have selected an existing server, click **Add Server** to erase the panel 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.

**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 panel. To verify that you have correctly configured the server, select your server in the list and then click **Test Connection**. The system reports errors if it cannot communicate with the server.

---

## Testing a server

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

To perform a diagnostic test:

1. Click **Servers**→ *server\_name*.
2. Click **Test Connection**.

A server connection test is performed and the server manifest is updated. A progress bar appears in the **Test Results** tab until the refresh is complete.

View the results in the **Test Results** tab. Click **View Test Details** for additional information.

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 a server

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

To disable or enable a server:

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 tab at the bottom of the main panel.
4. Click **Save Server**.

---

## Limiting concurrent jobs on a server

Use the **Max Jobs** server property to specify the maximum number of jobs that a particular server can run simultaneously.

## About this task

The system limits how many processes it attempts to run on any one server. The **Max Jobs** property specifies the limit on the number of processes. The default value for this property is 3.

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

**Note:** The built-in variables BF\_RESERVE and BF\_EXCLUSIVE control the reservation of one or all of a server's job slots. See "Pre-set properties" on page 239 for more information.

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

### Procedure

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**.

---

## Resetting the job count

### 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; if this number is set too low, jobs can fail from job contention.

This selection allows you to reset BF\_JOBS if it does not correctly reset when a job completes, fails, or is canceled.

For example, canceling 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; if this number is set too low, jobs can fail from job contention.

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**.

---

## 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.
- 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.

---

## Server authentication

This topic describes how to grant access to projects to run on servers.

### About server authentications

Use server authentications to associate login credentials to a server.

A server authentication stores a login name and password as a single named object that 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.

### About the server authentication panel

Use the server authentication panel to create and manage server authentications.

To view the panel, select **Servers** → **Server Auth**.

The screenshot shows the 'Server Auth' panel. At the top, there are tabs for 'UI Config', 'Console', 'Reports', and 'Logout: Root User'. The main panel has a header 'Server Auth' and a button 'Add Server Authentication'. Below this is a table with two columns: 'Name' and 'User name'. The table contains two rows: one for 'jsmith' and one for 'jdoe'. Below the table are buttons for '(New Server Authentication)', 'Save Server Authentication', 'Copy Server Authentication', and 'Delete Server Authentication'. The 'Details' section for '(New Server Authentication)' contains the following fields:

- Name:
- Access:
- Login:
- Password:
- Verified:

This panel provides the following fields:

**Name** The name to use for the authentication itself

**Access**

The access group of users who can use this authentication

**Login** The account name to use when logging in

**Password**

The password associated with Login

**Verified**

The password again

## Creating server authentications

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

### About this task

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:

### Procedure

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 panel, and a blank Server Auth Details panel at the bottom.

**Tip:** When you select a server authentication, the system populates the Server Auth Details panel with the selected server's authentication information. To clear the panel 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.

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.

## Results

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

## Overriding server authentication

By using 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. 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 run 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` variable 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.

**Tip:** On Windows, consider setting the variable `_USE_BFCREDS_DOMAIN` as well. Setting this variable to a value of 1 includes the user's domain.

## 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.

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.

## Selectors

This topic describes selectors.

### About selectors

Selectors choose a server resource on which 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 determines the server resource it runs on.

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

### About the Selectors panel

Use the Selectors panel to create and manage the selectors that choose a server resource on which to run a project or step.

To view the panel, select **Servers** → **Selectors**.

The screenshot shows the 'Selectors' panel in a web application. At the top, there are tabs for 'UI Config', 'Console', 'Reports', and a 'Logout: Root User' link. Below these is a header bar with 'Selectors >> All Selectors', an 'Add Selector' button, and a 'Help' link. A filter bar shows 'Showing 1 - 2 of 2' and a 'Display All' link. The main table has four columns: 'Selector Name', 'Access', 'Snapshot', and 'Snapshot Date'. It lists two selectors: 'HelloWorldSelector' and 'My\_selector', both with 'Build Engineer' access and 'Base Snapshot'. Below the table are buttons for '(New Selector)', 'Save', 'Create New Snapshot', 'Make Default', 'Copy', and 'Delete'. At the bottom, the 'Details' tab is active, showing a form with 'Name' and 'Access' fields. The 'Access' field is set to 'Build Engineer'.

Selector Name	Access	Snapshot	Snapshot Date
<a href="#">HelloWorldSelector</a>	<a href="#">Build Engineer</a>	Base Snapshot	----
<a href="#">My_selector</a>	<a href="#">Build Engineer</a>	Base Snapshot	----

Buttons: (New Selector), Save, Create New Snapshot, Make Default, Copy, Delete

Details tab: Name: [text input], Access: Build Engineer [dropdown]

**Details tab:** The Details tab provides the following fields:

**Name** The name for the selector.

**Access**

The access group of users who can use this selector.

**Snapshot tab:** A selector snapshot is an instance of a selector. The Snapshot tab provides the name of the snapshot. Use this tab to view or change the snapshot name and comments about the snapshot.

## 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 used in the system. Use one of these selectors when you want to 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.
  - 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 from 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 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.

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

Property name	Manifest property value	Operator	Selector variable value	Comparison type	Match?
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 points 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, the 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.

---

## Collectors

### About collectors

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

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

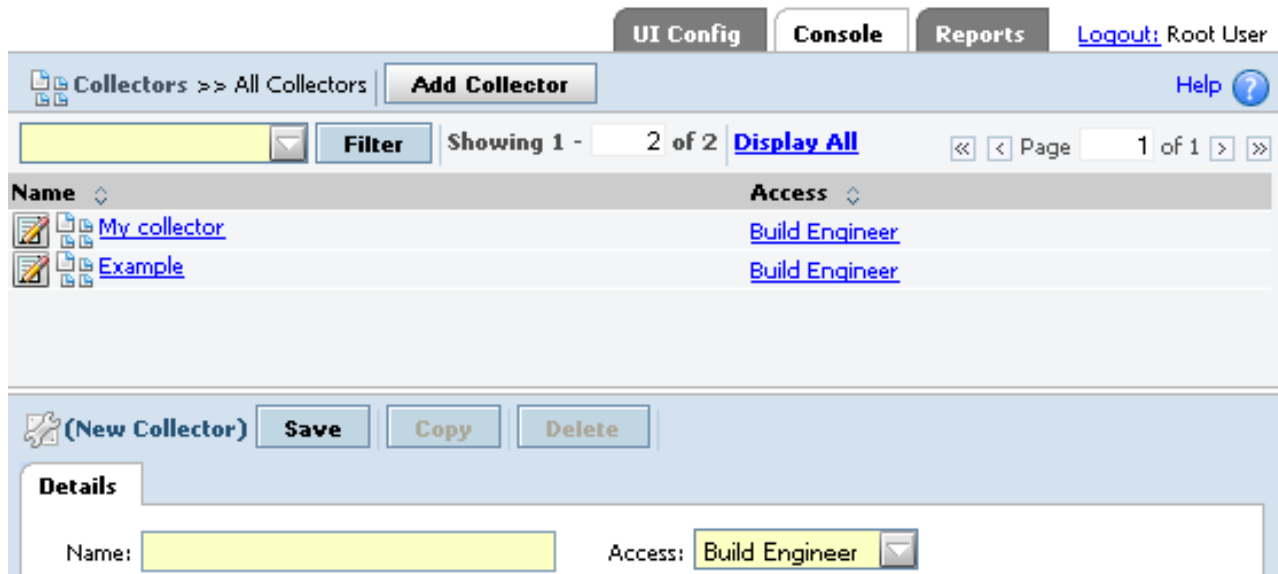
To open the Collectors section for the Servers panel, click **Servers** → **Collectors**.

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 to use collectors.

### About the Collectors panel

Use the Collectors panel to create, edit, and delete collectors. To view the panel, select **Servers** → **Collectors**.



The panel has the following fields.

**Name** The name for the collector.

**Access**

The access group of users who can use this collector.

## Creating collectors

Create collectors to enable dynamic selection of server resources for your projects.

### Procedure

1. Select **Servers** → **Collectors**.
2. Specify a name for the collector.
3. Specify access for the collector.
4. Click **Save**. The panel changes so you can enter variables that determine what data to collect.
5. Create and save one or more variables. For information about the variable properties, see “Collector variable property types.”

### Collector variable property types

Collector variable 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 properties in collectors” on page 240.

#### 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” on page 235.

### 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 are 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 about 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 variable 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 variable 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 properties” on page 239.

---

## 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.
- 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 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 select 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 256 MB 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 at least 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 time 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:

1. Click **Servers**.
2. Click the name of the server resource.
3. 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 between intervals, click **Queue Manifest Refresh**.

## 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 **Queue Manifest Refresh**.

A server connection test is performed and the server manifest is updated. A **Refreshing manifest** progress bar appears in the **Manifest** tab until the refresh is complete.

Automatic server manifest refresh intervals are set by the **Server Test Frequency** system setting. Manual manifest refreshes in the queue are performed before any automatic manifest refreshes in the queue.

## 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.

### About this task

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

- 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 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 define a variable with the name **\_AGE** in a collector to control how often servers using that collector are queried for their manifest properties. Between **Default \_AGE** and **\_AGE**, the system uses the smaller value. To create this variable:

### Procedure

1. Click **Servers** → **Collectors**.
2. Select an existing collector, or create a new one.
3. Add a variable to the collector.
  - a. Select the type **Set Value**.
  - b. Type the **Variable** name **\_AGE**.
  - c. Specify a **Value** in seconds.
  - d. Click the **Save** 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=BF\_NAME, 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
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 256 MB 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 at least 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 time 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.

## Properties reference

The following topics describe properties that affect manifests:

- “Built-in properties”
- “Pre-set properties” on page 239
- “Special properties in collectors” on page 240

### Built-in properties

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

### 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.

**Note:** The availability of a property varies from platform to platform.

*Table 8. Built-in Properties for Collectors and Manifests*

Property	Description
CPU_ARCH	The returned value is a label for an architecture name, as shown: <ul style="list-style-type: none"> <li>• HP-PA: HP Precision Architecture</li> <li>• IA-64: Intel Itanium</li> <li>• MVS: IBM S/390</li> <li>• PPC: PowerPC</li> <li>• PPC-64: PowerPC 64</li> <li>• SPARC: Sun SPARC</li> <li>• X86: x86-compliant architecture used by Intel, AMD, Cyrix, and others</li> </ul>
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. <b>Note:</b> 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. <b>Note:</b> 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_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. <b>Note:</b> 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.



Table 8. Built-in Properties for Collectors and Manifests (continued)

Property	Description
CPU_MHZ	<p>This property reports the processor speed in megahertz. Certain conditions are required for this property to be filled in successfully:</p> <ul style="list-style-type: none"> <li>• In Linux: frequency scaling must be enabled.</li> <li>• In Windows: the ~MHz registry entry must exist and be filled in.</li> <li>• For x86 and x86-64 processors inline assembly must work.</li> </ul>
CPU_MANUFACTURER	<p>This property returns the company name of the processor manufacturer. If the information is not directly available, the names are assumed based on architecture. No value is returned if there is insufficient processor information available. These values are supported:</p> <ul style="list-style-type: none"> <li>• AMD: for their x86 and AMD64 processors</li> <li>• Cyrix: for their x86-compliant processors</li> <li>• DEC: for Alpha and VAX</li> <li>• HP: Hewlett-Packard Precision Architecture</li> <li>• IBM: IBM S/390 and PowerPC G5</li> <li>• Intel: Intel x86 (including Intel64), IA-64 Itanium</li> <li>• Motorola: PowerPC G4</li> <li>• NexGen: x86-compliant processors</li> <li>• National: National Semiconductor x86-compliant processors</li> <li>• Rise: Rise x86-compliant processor</li> <li>• Sis: Sis x86-compliant processor</li> <li>• Sun: Sun Microsystems SPARC</li> <li>• TransMeta: TransMeta x86-compliant processor</li> <li>• UMC: UMC x86-compliant processor</li> <li>• VIA: VIA Technologies x86-compliant processor</li> </ul>
CPU_MODEL	<p>This property returns the manufacturer-specific CPU model numbers. These values are reported as follows:</p> <ul style="list-style-type: none"> <li>• x86 architecture <ul style="list-style-type: none"> <li>– 386</li> <li>– 486</li> <li>– 586</li> <li>– 686</li> <li>– X86_64</li> </ul> </li> <li>• PowerPC architecture <ul style="list-style-type: none"> <li>– 6xx</li> <li>– POWER</li> <li>– RS64</li> <li>– G3</li> <li>– G4</li> <li>– G5</li> <li>– Cell</li> </ul> </li> </ul>
CPU_SERIAL	<p>This property returns the serial number of the CPU or computer. Currently, this functionality is limited to these architectures:</p> <ul style="list-style-type: none"> <li>• x86: Intel or Transmeta serial numbers only. Note: Most x86 processors will not report a serial number. No value is returned in such cases.</li> <li>• MacOS/X: The serial number assigned is retrieved from an I/O registry. To report the property, it must be able to find the CoreFoundation and IOKit frameworks.</li> </ul>

Table 8. Built-in Properties for Collectors and Manifests (continued)

Property	Description
DISK_FREE	<p>For UNIX and Linux, this property returns the amount of free space (in MB) on the file system that the server Path specifies.</p> <p>For Windows, the free disk space (in MBs) on the drive that the server Path specifies.</p> <p>For example, a disk with 4 GB of free space is reported as 4096 MB.</p>
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 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 2048 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	<p>This property reports total RAM, or system memory (in MB).</p> <p>For example, a computer with 2 GB of RAM would be reported as 2048 MB.</p>
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	<p>This property returns the name of the interface that the agent uses to communicate.</p> <ul style="list-style-type: none"> <li>• In Windows, the reported name is the name that ipconfig command returns, for example, Intel(R) PRO/100 VE Network Connection - Packet Scheduler Miniport</li> <li>• In other operating systems, the reported name is the name that ifconfig returns, for example, en0 or eth0 or OSA1.</li> </ul>
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.
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	<p>This property returns the operating system release level of the server.</p> <p>For example, if the server operating system is Microsoft XP Version 5.1.2600, this returned value is 5.</p>

Table 8. Built-in Properties for Collectors and Manifests (continued)

Property	Description
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 properties

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 to populate them.

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

Table 9. Automatically set manifest properties

Property	Description
BF_AGENT_VERSION	<b>Selector.</b> The version number of the agent that is installed on the server.
BF_EXCLUSIVE	<b>Selector.</b> 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	<b>Selector.</b> 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 <sup>®</sup> 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 <sup>®</sup> timestamp, as the number of seconds since January 1, 1970.
BF_LOADRATIO	<b>Selector.</b> 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	<b>Selector.</b> 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.

Table 9. Automatically set manifest properties (continued)

Property	Description
BF_RESERVE	<p><b>Selector.</b> 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.</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• If a step specifies the selected server explicitly, the server uses the reserved slot for that step.</li> </ul> <p>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.</p>

## Special 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.

Table 10. Special set value properties for collectors and manifests

Property	Description
_AGE	<p>This property defines 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 <b>Default _AGE</b>. Between <b>Default _AGE</b> and <b>_AGE</b>, the system uses the value that produces more frequent refreshes.)</p>

## 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 a temporary backup or as part of an official archive.

- 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

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 console, 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. 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 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

Creating a selector snapshot creates a new instance of the selector. The snapshot is not a copy; it is a new selector.

**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 299.

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 <b>Servers &gt; Selectors</b> to display the selectors list.
Follow Selector Includes	If the selector uses the <b>Include</b> property type to include other selectors, a snapshot is created for those selectors. <b>Note:</b> The <b>Include</b> variable type replaces the .include functionality provided in prior releases.



- 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 299.

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

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



<b>OK</b>	<b>Update references:</b> If any objects reference the previous default selector, update the objects to use the new default selector.
<b>Cancel</b>	<b>Do not update references:</b> 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:

- Select **Servers > Selectors**.
- In the selectors list, click the **Snapshot** icon  for the default selector snapshot.
- In the snapshots list, click the **Edit** icon  for the selector snapshot.
- 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	<p><b>Change the selector snapshot name and other snapshot object names:</b> 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.</p> <p><b>For the Base Snapshot:</b> Changes the name of Base Snapshot for all current selector snapshots and all future selector snapshots.</p>
Cancel	<p><b>Change the selector snapshot name but do not change other snapshot object names:</b> 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.</p> <p><b>For the Base Snapshot:</b> Retains the name Base Snapshot for all current selector snapshots and all future selector snapshots.</p>

## Accessing and viewing snapshots in a selector snapshot set

Snapshot a selector 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**.
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**.



---

## Chapter 17. 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 in jobs. Environments can be assigned to servers, projects, and steps. During job execution, a running step inherits variable values from all three environments:
  - 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.

### Environment inheritance

Before the system runs 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 Y is set to 1, then 100 due to the Set action on Y in the server and project environments. Because Y 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.

### 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 a 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.

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 run 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 environment 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.

The screenshot shows the 'Environments' web interface. At the top, there are tabs for 'UI Config', 'Console', 'Reports', and a 'Logout: Root User' link. The main header shows 'Environments > MyEnv' and an 'Add Environment Variable' button. Below this is a table of existing variables:

Name	Type	Value	Action	On Project
MyStandardLevel	Standard Variable	High	Set	Normal
Flavor	Pull-down List		Set	Required
Include	Include	suppress ENV		

Below the table are buttons for '(New Variable)', 'Save Variable', and 'Delete Variable'. The 'Details' tab is selected, showing a form to create a new variable:

Type:  Name:  Value:

Action:  On Project:

To create a new variable:

1. Click **Environments** in the left menu.
2. Click **Add Environment Variable**.
3. Specify a name for the variable.
4. Specify other properties for the variable as desired. See “Environment properties” on page 251 for more information.
5. Click **Save Variable**.

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

## Interpretation of variables in steps

You can use either a UNIX<sup>®</sup>-style or Windows<sup>®</sup>-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:

```
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.

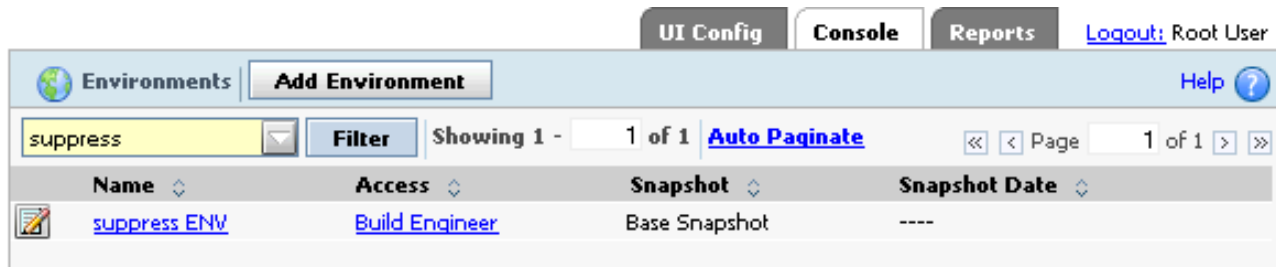
## Interpretation of undeclared variables

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” on page 261.

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

## About the Environments panel

Use the Environments panel to create and manage environments for your projects. To view the panel, select **Environments**.



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

## Details tab

### Environment properties

Specify the following environment properties:

**Name** The name for the environment.

**Access**

The access group of users who can use this environment.

### Variable properties

In the Environments panel, select a variable to view the Variables panel. 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 "Interpretation of variables in steps" on page 250. If a value is in single quotes, variable names are passed as literals rather than interpreted. For example, a variable assignment `MyEnv='$env'` causes the string `$env` to be assigned to `MyEnv`, rather than the value of variable `$env`.
- **Variables in values:** if a variable is in a variable value, that variable is interpreted when a step using the variable is run.
- **Pulldown values:** To specify items for a pulldown variable, set the variable type to Pulldown List, save the variable, and then edit it. Click the Pulldown Options tab to add items to the pulldown. See "Creating pulldowns for a variable" on page 254.

- **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 257.
- **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. After 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 “About variables” on page 249.
- **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 the current value. The OS-specific PATH delimiter is added between the values:
  - Windows: semicolon (;)
  - UNIX or Linux: colon (:)
- **Clear:** The value is set 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.
- **Assign Hidden:** The system assigns the variable, but hides the value in the logs, showing it as “\*\*\*\*\*”. Use this option to hide the password from users who run the project.



**Important:** Use caution if you store sensitive information in hidden variables. It is possible to read the values of hidden variables by inspecting the database or by using an API client. They are not secure for sensitive information, such as passwords.

You cannot edit a variable that is set to Assign Hidden. To change the value of such a variable, delete the variable and then re-create it.

**Note:** If a variable in a step is set to Assign Hidden, all variables in that step are treated as if they were set to Assign Hidden.

**Note:** Assign Hidden variables only apply to projects. If an Assign Hidden variable is passed to an Adaptor, it no longer remains hidden.

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 does not do this for a hidden variable.* The variable is passed directly to the server and the operating system environment of the server interprets the variable.

Therefore, do not use variable names that could be parsed by either operating system. Examples:

- \$name: if this variable is set to Assign Hidden, then when it is interpreted on a UNIX or Linux system, the operating system attempts to resolve it in its environment, not the Build Forge environment.
- %address%. if this variable is set to Assign Hidden, then when it is interpreted on a Windows system, the operating system attempts to resolve it in its environment, not the Build Forge environment.

### On Project

Defines how a variable is used when you manually start a job. This property affects only variables that are used in environments assigned to a project. The property does not affect variables when the job is running. 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. Top-level 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.

**Note:** If a job that contains Must Change variables is started by the scheduler rather than a user, the variable values are not changed. Also, if a job that contains Must Change variables is started by a class property, the variable values are not changed. Class properties include Start on purge, Start on entry, and Start on exit.

## Snapshot tab

An environment snapshot is an instance of an environment. The Snapshot tab provides the name of the snapshot. Use this tab to view or change the snapshot name and comments about the snapshot.

---

## Creating an environment

### Procedure

1. In the left menu, click **Environments**.
2. Click **Add Environment**.
3. Specify a name for the environment.
4. Specify the access group whose members should be able to use the environment.
5. Click **Save**. A variables panel is presented where you can add variables to the environment.

---

## Using variables

The following sections describe procedures for accomplishing common tasks with variables.

## Creating pulldowns for a variable

### About this task


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 runs. It is not preprocessed or evaluated before the step runs.

- c. Click **Create**. The option is added to the list.
- d. Repeat for each desired value.
8. Click **Save Variable**.
9. Set the default option presented. Do this after you have populated **Pulldown Options** and saved the variable. Click the environment. In the **Details** tab, change the **Default Option** property from `--NONE--` to the name of the value you want to be presented by default.  
 Note that when viewing the **Details** tab for the variable, if **Default Option** is not set (its value is `--NONE--`), the first item in the **Pulldown Options** list is presented by default in the **Value** column.
10. 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.
- 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 to create a new option.

## 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.

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

The Microsoft Windows<sup>®</sup> operating system manages mapped drives differently. 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 `'`.

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

Even if the drives map successfully, drive mappings on Windows 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<sup>®</sup> 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.

## 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 run 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 run 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 run before the step command. Some important notes on .source:

- The path specified cannot have arguments.
- On Windows<sup>®</sup> platforms, the script is invoked via call.
- On UNIX<sup>®</sup> 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 331 for more information about using this command, including a list of date format codes.

---

## System variables reference

System-defined variables are available to use in variables.

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.

Step-Level Variable	Value
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 in the installation directory: engine.id. Example: D8531015-6C07-1014-8CA0-BD58317220B3.
BF_HOST	Host name of the logical server (TCP/IP hostname). (This variable is part of the server environment.)
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 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 run. 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 unique ID (BF_SSID) of the While Loop step is <i>appended</i> 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 <sup>®</sup> 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.



Step-Level Variable	Value
BF_PROJECTNAME_PHYS	Project name as used to create the project directory. The system changes characters specified in the <b>Invalid Relative Dir Characters</b> 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 (this variable is part of the server environment).
BF_SERVER_ROOT	Path assigned to the logical server in the server properties (this variable is part of the server environment).
BF_SID	The sequence number of the step result within the build.
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, unique identifier for the current step in the project.
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 selected, 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 ran.)
_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 run. You might want to keep a copy of the build records for the job, for example, for debugging.
CLEARCASE_VIEW	Starts the specified ClearCase view. The view specified in this variable must exist, and the step using this variable must be set to "absolute". On systems running Microsoft Windows, this variable must be used with the cc_suppress_server_root configuration option for the agent in bfaagent.conf.
_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.

Variable	Contents
_ERROR_THRESHOLD	<p>Establishes the maximum number of errors (caught by the Set Fail filters you have defined) allowed. Using this variable, you can establish failure and message thresholds for individual steps or for a project.</p> <p>Use one of the following forms:</p> <ul style="list-style-type: none"> <li>• A value of 5 or F5 indicates that the job should fail if more than 5 errors occur.</li> <li>• A value of N7 indicates that the system should add a message to the job notes when more than 7 errors occur. The message indicates that this threshold was met.</li> </ul> <p>When you use the variable in a step, the system counts the errors in the individual step. Additional forms are available:</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> </ul> <p><b>NOTE:</b> 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.</p>
_EXITCODE_MAP	<p>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.</p>

Variable	Contents
_InterfaceLoggingLevel	<p>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.</p> <ul style="list-style-type: none"> <li>• 0: Exec line plus server connection errors or cancel notification; nothing else</li> <li>• 1: Parsed commands (commands as they will be sent to the server)</li> <li>• 2: Unparsed commands (commands prior to having their local variables set)</li> <li>• 3: Build and environment variable SET lines</li> <li>• 4: Temp and internal variable SET lines</li> <li>• 5: Environment evaluations, email group additions, BOM text logging lines</li> <li>• 6: Block &amp; Sub-block start/end lines</li> <li>• 7: (Default logging level) Agent output that is checked against match patterns, plus the lines that matched the patterns.</li> <li>• 8: All agent output</li> </ul>
_LOG	<p>Specifies a path name to create a log file containing the Build Forge Agent's raw output.</p> <p><b>Note:</b> This log does not include time stamps unless _LOG_TIMESTAMP is also specified. The log data in this file is typically formatted as such: agent code, log bucket, and message.</p> <p>Use this variable to save a copy of the job log on the server. If the file exists, the system appends to it.</p>
_LOG_TIMESTAMP	<p>Prefixes each line of output from _LOG with a timestamp. The value of this variable should be a format string in the same strftime syntax that is used by the .date and .gmdate environment commands.</p> <p><b>Note:</b> Requires _LOG.</p>
_MAP	<p>See "Mapping Windows drives" on page 256 for a discussion of how to use this variable.</p>

Variable	Contents
_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 run. See “Special variables for test projects” on page 471.
_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 471.
_PRISM_FILE_POSTCMD	Used with plug-ins for IDEs. Specifies a command to be run on files after the project step has run. See “Special variables for test projects” on page 471.
_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 471.
_SUPPRESS_ENV_OUTPUT	Specifies that the system omit the environment messages from the log. By default, this variable is not set and all variable values in the environment are printed before a step command runs. The values appear as ENV entries in the step log. The variable can be set to the following values: <ul style="list-style-type: none"> <li>• ALWAYS: always omit the ENV messages</li> <li>• Any other value: omit the ENV messages. However, if the command fails, the ENV messages are printed after the command message. This information may be useful in debugging the command execution failure.</li> </ul>
_SUPPRESS_AGENT_LOG_OUTPUT	When defined (with any value), causes the system to omit nearly all of the log output normally produced by the agent. The similar _SUPPRESS_LOG_OUTPUT variable suppresses output in the engine, while the _SUPPRESS_AGENT_LOG_OUTPUT variable suppresses output in the agent. <b>Note:</b> Using this variable prevents filter matches.

Variable	Contents
_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 run if the current step fails; the string can be set to the name of an executable file or command. <b>NOTE:</b> 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 issued through _TRAP, have the command write its output to a file for later retrieval.
_USE_BFCREDS	When set to 1, the system uses the <i>user's</i> 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 run the commands in the project. You can set this variable for a single step, or for an entire project. <b>Note:</b> If you are using LDAP/Active Directory authentication, the <b>Store User Authentication Locally</b> 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. <b>Tip:</b> On Windows, consider setting the variable _USE_BFCREDS_DOMAIN as well.
_USE_BFCREDS_DOMAIN (Windows only)	When set to 1, the system uses the <i>user's</i> domain in addition to the login credentials that _USE_BFCREDS uses to log in to servers.

---

## Environment snapshots

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.


## 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:

- 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

Creating an environment snapshot creates a new instance of the environment. The snapshot is not a copy; it is a new environment.

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

**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 299.

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 <b>Environments</b> to display the environments list.
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 299.

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.

<b>OK</b>	<b>Update references:</b> For any objects that reference the previous default, update references from the previous default environment snapshot to the new default.
<b>Cancel</b>	<b>Do not update references:</b> 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	<p><b>Change the environment snapshot name and other snapshot object names:</b> 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.</p> <p><b>For the Base Snapshot:</b> Changes the name of Base Snapshot for all current environment snapshots and all future environment snapshots.</p>
Cancel	<p><b>Change the environment snapshot name but do not change other snapshot object names:</b> 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.</p> <p><b>For the Base Snapshot:</b> Retains the name Base Snapshot for all current environment snapshots and all future environment snapshots.</p>



## 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**.

## Chapter 18. Working with projects

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

### About projects

Projects are executable sets of steps, with their own environment group and server properties.

In addition to projects, you can create libraries. A library is like a project but does not have a selector to determine where the steps run.

### About the Projects panel

Use the Projects panel to create new projects and view or edit existing projects. To access the Projects panel, in the left menu, click **Projects**.

The Projects panel provides the following tabs:

- “Details tab”
- “Tags tab” on page 273
- “Registers tab” on page 273
- “Notes tab” on page 274
- “Snapshot tab” on page 274

### Details tab

With the Details tab, you can specify various properties about a project.

The screenshot shows the IBM Management Console interface. At the top, there are tabs for 'UI Config', 'Console', and 'Reports', along with a 'Logout: Root User' link. The 'Projects' tab is active, showing a list of projects. Below the list, there are buttons for 'Add Project', 'Save', 'Create New Snapshot', 'Make Default', 'Copy Project', 'Delete Project', and 'Clobber'. The 'Project Details' tab is selected, showing a form for editing a project. The form includes fields for Name, Access, Max Threads, Run Limit, Pass Chain, Fail Chain, Class, Selector, Environment, Sticky, Start Notify, Pass Notify, and Fail Notify.

Project	Snapshot	Tag	Class	Environment	Selector	Access
HelloWorld	Base Snapshot	BUILD_\$B	Production		My_selector	Build Engineer
Say_hi	Base Snapshot	BUILD_\$B	Production		My_selector	Build Engineer

Name: <input type="text"/>		Access: <input type="text" value="Build Engineer"/>		<input type="checkbox"/> Disable	
Max Threads:	<input type="text" value="Unlimited"/>	Class:	<input type="text" value="Production"/>	Start Notify:	<input type="text" value="None -"/>
Run Limit:	<input type="text" value="Unlimited"/>	Selector:	<input type="text" value="My_selector"/>	Pass Notify:	<input type="text" value="None -"/>
Pass Chain:	<input type="text" value="None -"/>	Environment:	<input type="text" value="None -"/>	Fail Notify:	<input type="text" value="None -"/>
Fail Chain:	<input type="text" value="None -"/>	Sticky:	<input type="text" value="Not Sticky"/>		

**Name** The name of the project. The system uses this name 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 run. Because a project might contain steps that run on different operating systems, avoid special characters and symbols in your project

names. If a project name must use characters that produce an invalid directory name, add the characters to the Invalid Relative Dir Characters system configuration setting. Characters listed in this setting are converted to underscores (\_) when creating project directories. Using an ampersand (&) can unintentionally produce an HTML entity as part of the project name.

#### **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 about access groups, see “Access overview” on page 187.

#### **Disable**

Select this check box to disable the project. When users attempt to run a disabled project, Rational Build Forge displays a message to indicate that the project is inactive and does not run the project.

#### **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 313.

#### **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 and the number of active jobs equals the Run Limit, then the new job stays in the Waiting queue until at least one job completes.
- If a schedule attempts to launch a project and the number of active jobs equals the Run Limit and the Hard Run Limit system configuration setting has a value of Yes, the system does not launch the new job. If the Hard Run Limit is set to No, the system ignores the Run Limit setting for scheduled builds.
- Projects that are launched through an inline are not considered instances of the original project and do not count toward its Run Limit.

**Class** Each project must be assigned to a class, which assigns global properties to groups of jobs. For more information, see “Classes” on page 288.

#### **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 227. If a selector is not specified, the project is added to the Libraries panel instead of the Projects panel. A library uses the selector of the calling step; if that step does not have a selector, the library uses the selector of the calling project.

#### **Pass Chain, Fail chain**

Select the project that runs 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 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 about how environments work together, see “About variables” on page 249.

**Sticky** Enable the Sticky check box to force all the steps of the project that use the default project selector to stay on the same server, and to wait for it to become available if it is busy. For more information about this option, see “Making steps stick with a server” on page 277.

#### **Start Notify, Pass Notify, Fail Notify**

Use these fields to direct the system to send notification emails on project start, pass, and/or fail, by selecting an access group in one or all of these fields.

### **Tags tab**

Use the Tags tab to manage the build tags for a project.

#### **Tag Format**

A string that defines the tags for the project, using plain text and tag variable references. For more information about tag formats, see “Changing the build tag during a job” on page 321.

#### **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 280.

#### **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.

### **Registers tab**

Use project registers to store information that persists across builds.

**Register**

The name of the project register.

**Contents**

The value of the register.

For additional information, see “Project registers” on page 324.

**Notes tab**

Use the Notes tab to store items of interest about a project.

For each note, the tab displays:

- The date and time a user created the note
- The user who entered the note
- The note

**Snapshot tab**

A project snapshot is an instance of a project. The Snapshot tab provides the name of the snapshot.

Use this tab to view or change the snapshot name and comments about the snapshot.

---

## Changing project properties

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

**Project Name**

The name of the project. The system uses this name 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 run. Because a project might contain steps that run on different operating systems, avoid special characters and symbols in your project names. If a project name must use characters that produce an invalid directory name, add the characters to the Invalid Relative Dir Characters system configuration setting. Characters listed in this setting are converted to underscores (\_) when creating project directories. Using an ampersand (&) can unintentionally produce an HTML entity as part of the project name.

**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 about access groups, see “Access overview” on page 187.

**Tag Format**

A string that defines the tags for the project, using plain text and tag variable references. For more information about tag formats, see “Changing the build tag during a job” on page 321.

**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 280.

### **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 313.

### **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 and the number of active jobs equals the Run Limit, then the new job stays in the Waiting queue until at least one job completes.
- If a schedule attempts to launch a project and the number of active jobs equals the Run Limit and the Hard Run Limit system configuration setting has a value of Yes, the system does not launch the new job. If the Hard Run Limit is set to No, the system ignores the Run Limit setting for scheduled builds.
- Projects that are launched through an inline are not considered instances of the original project and do not count toward its Run Limit.

**Class** Each project must be assigned to a class, which assigns global properties to groups of projects. For more information, see “Classes” on page 288.

### **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 227. If a selector is not specified, the project is added to the Libraries panel instead of the Projects panel. A library uses the selector of the calling step; if that step does not have a selector, the library uses the selector of the calling project.

### **Pass/fail chain**

Select the project that runs 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 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 about how environments work together, see “Environment inheritance” on page 247.

**Sticky** Enable the Sticky check box to force all the steps of the project that use the default project selector to stay on the same server, and to wait for it to become available if it is busy. For more information about this option, see “Making steps stick with a server” on page 277.


### Start Notify, Pass Notify, Fail Notify

Use these fields to direct the system to send notification emails 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, complete the following steps:

1. Select **Projects**.
2. Select the **Edit** icon  next to the project you want to copy.
3. Click **Copy Project**.

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

- The steps and all of their properties listed on the step's **Details** tab
- All the project properties listed on the project's **Details** tab, such as the project's class, selector, and other properties
- The project's tag format (found in the **Tags** tab on the project properties panel)

The system does not copy the following properties:

- Tag variables (found in the **Tags** tab on the project properties panel); however, tag variables used in the tag format are copied
- Project registers (found in the **Registers** tab on the project properties panel)
- Step notes (found in the **Notes** tab on the step properties panel)
- Project notes (found in the **Notes** tab on the project properties panel)

---

## Deleting a project

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

Choose one of the following options to delete a project:

### Delete Project button

**Note:** Deleting a project cannot be undone.

The **Delete Project** button deletes projects that have no jobs. To delete a project with this button, you must first delete all of the project's jobs. The button is on the project property editing page and the project step list. To view the project property editing page, complete the following steps:

1. In the left menu, select **Projects**.
2. Click the **Edit** icon  next to the desired project's name.

### Clobber button

**Note:** A clobber operation deletes projects even if they are locked. Also, you cannot undo the clobbering of a project.

The **Clobber** button deletes a project and all of its associated jobs from the Build Forge database. The system asks for confirmation before clobbering a project. The button is on the project property editing page. To view the project property editing page, complete the following steps:

1. In the left menu, select **Projects**.



2. Click the **Edit** icon  next to the desired project's name.

---

## 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.

To view project-level properties, select **Projects**, and then click the **Edit** 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 Default, 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 Default. This property persists across restarts of a project.

When the system starts an inline project, the system uses the inline project's selector as the default selector for the steps of the inline. 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.

---

## Chains: conditional execution of another project or library

There are two types of chain that can be called at the project level:

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

This feature has several uses:

- Employ conditional execution at the project level. Other flow-control capabilities are provided at the step level. In addition, a step can have its own Pass Chain and Fail Chain. See “Controlling execution flow” on page 310 in Chapter 19, “Working with steps,” on page 305.
- Maintain frequently used groups of steps separately from projects that depend on them. Libraries can also be used for this purpose.
- Clean up files after a project passes or fails.
- Call automated test and deployment projects when a software build project passes.

## Chain inheritance from the calling project or library

A chained project or library inherits some characteristics from the calling project:

- A chained project inherits the *class of the calling project* by default. You can change this behavior in **Administration** → **System** by setting Override Class when Chaining to No.
- A chained library inherits the calling step's selector because a library does not have its own selector. If that step does not have a selector, the library inherits the build's selector. Steps of the chained library use the inherited selector, unless those steps have explicit selectors.

A chained project or library otherwise runs with its own characteristics:

- Its server is specified by its own selector.
- It uses its own properties, including its own notification settings and chain settings.
- Its environment is applied after the calling project's environments. Variables from the calling project are renamed and available in the called project. See also "Environment variable inheritance in chained projects."

## Chain nesting

When you chain a project, the called project is nested in the calling project. The maximum level of nesting is 32 levels. The level of nesting may also encounter limits based on the available memory on the host running Management Console.

## Running and interrupting chains

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 329.

## Environment variable inheritance in chained projects

When a project is launched through a pass/fail chain, the system applies environment variables from the calling project. 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.
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 run.

## 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 343.)

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 about these variables, see “System variables reference” on page 258.)

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 280.)

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 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, and 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

## Synchronizing tags

You can synchronize tags across different projects, so that two or more projects use the same variable value, 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 tag variables in the parent project. If found, the child project's variable is set to the parent project's last used value.

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.

Synchronization works only on the tag variable's value. The Auto Increment and Padding properties are not synchronized.

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

For example, consider two projects defined as shown in the following table:

Project	Tag format	Auto Increment	Tag sync
Project A	Project_A_\$B	Yes	-- None --
Project B	Project_B_\$B	Yes	Project A

If you then run the projects alternately (starting with Project A), the completed jobs list shows the tags as follows. The last run is shown first, the same way jobs are shown in the completed jobs list.

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

If you set Auto Increment property to Yes only on the parent project, the results are different. The projects are set up as follows:

Project	Tag format	Auto Increment	Tag sync
Project A	Project_A_\$B	Yes	-- None --
Project B	Project_B_\$B	No	Project A

If you then run the projects alternately (starting with Project A), the completed jobs list shows the tags as follows. The last run is shown first, the same way jobs are shown in the completed jobs list.

Project	Tag
Project B	Project_B_2
Project A	Project_A_2
Project B	Project_B_1
Project A	Project_A_1

## 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

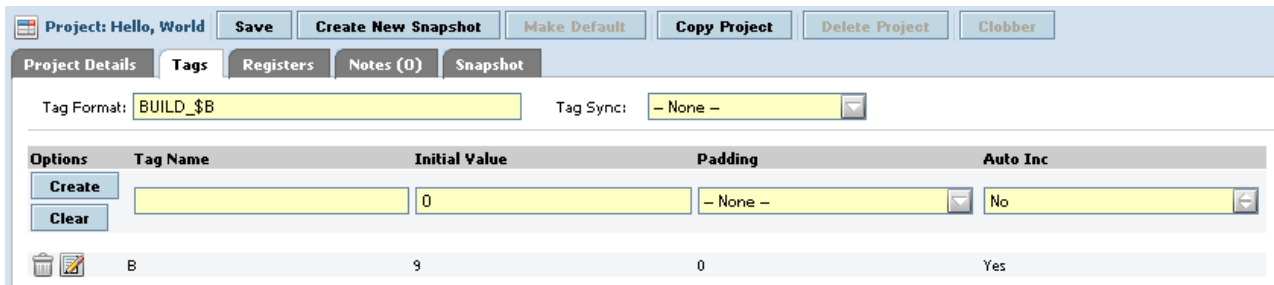
## Creating or editing tag variables



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 existing ones for a project, select **Projects**, and then

click the **Edit** icon  next to the desired project's name. The project properties appear in the bottom of the panel; click the **Tags** tab to display the project's tag variables.

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



Options	Tag Name	Initial Value	Padding	Auto Inc
<input type="button" value="Create"/>		0	— None —	No
<input type="button" value="Clear"/>				
 	B	9	0	Yes

- To edit a tag variable, click the **Edit** icon  next to its name. The system fills the panel with the tag variable's values and changes **Create** to a **Save** button. Change the values and click **Save** to store your changes.
- To delete a tag variable, click the trash can icon next to its name.
- To add a new variable, enter properties for the variable and click the **Create** button.

For information about the properties Tag Name, Initial Value, Padding, and Auto Inc, see “Tags tab” on page 273.

---

## 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 library. 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 step that calls them. If a calling step does not have a selector, the library uses the selector of the step's project. Libraries are typically called by other projects as an Inline for a step or as a Pass chain or Fail chain for a step.

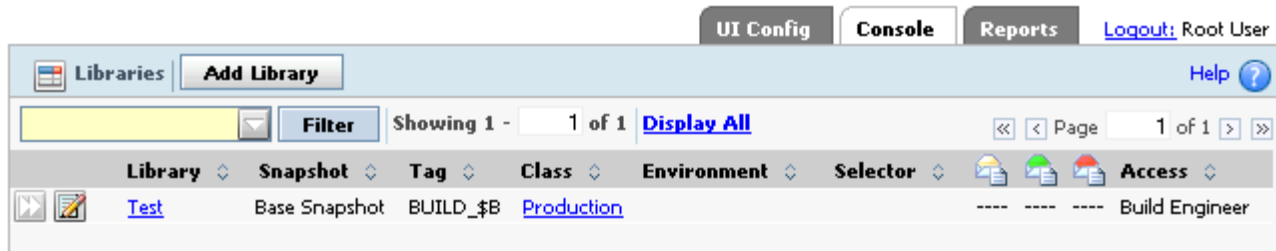
From the Libraries panel, you can view, edit, create, or launch a library. You can run a library by itself, but you must specify a selector when you do this.

**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 into a project by editing the project and choosing a selector for it. When you save a library with a selector, it becomes a project, disappearing from the Libraries list.


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

To access the Libraries panel, in the left menu, click **Libraries**.



## Copying a library

To make a copy of an existing library, complete the following steps:

1. Select **Libraries**.
2. Select the **Edit** icon  next to the library you want to copy.
3. Click **Copy Library**.

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

- The steps and all of their properties listed on the step's **Details** tab
- All the library properties listed on the library's **Details** tab, such as the class and other properties
- The library's tag format (found in the **Tags** tab on the library properties panel)

The system does not copy the following properties:

- Tag variables (found in the **Tags** tab on the library properties panel); however, tag variables used in the tag format are copied
- Library registers (found in the **Registers** tab on the library properties panel)
- Step notes (found in the **Notes** tab on the step properties panel)
- Library notes (found in the **Notes** tab on the library properties panel)

---

## Log filters

This topic describes how to create and use log filters.

### About log filters

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

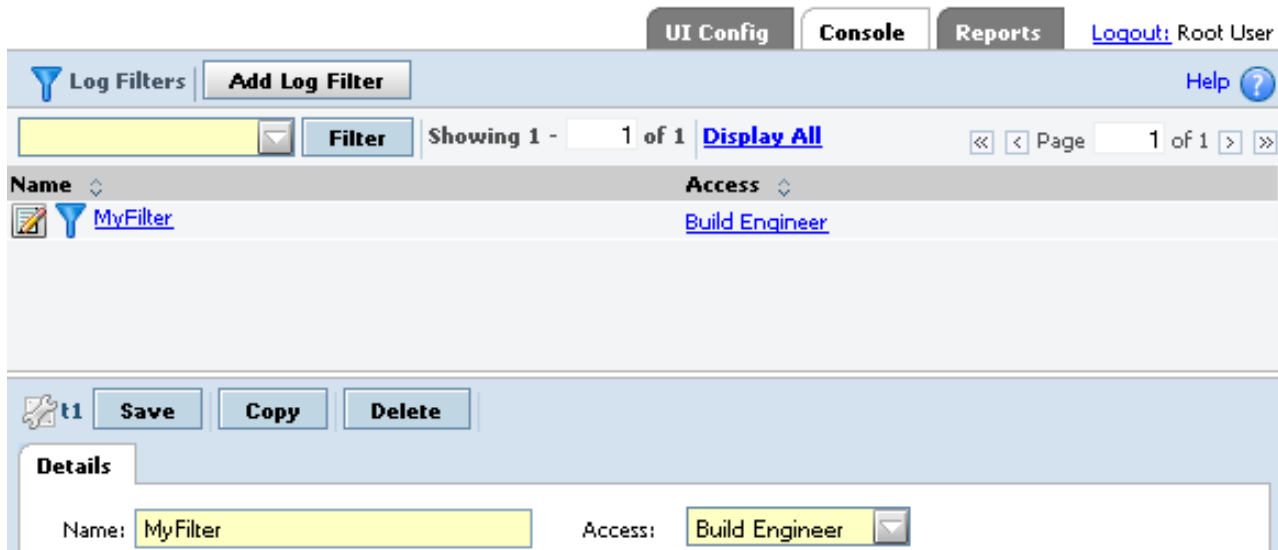
If filtering is not set up, Build Forge determines the success or failure of a step command by the command's exit status, where 0 is success and 1 is failure. If multiple commands are used in the Command property, only the exit status of the last command issued affects the step result status. Note that 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.

### About the Log Filters panel

Use the Log Filters panel to create, edit, and delete log filters. To view the panel, select **Projects** → **Log Filters**.



The panel has the following fields.

**Name** The name for the log filter.

**Access**

The access group of users who can use this filter.

### 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 panel.
2. At **Name**, enter a name for the log filter, and then click **Save**. The Management Console saves the log filter and displays the New Pattern panel.
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 285.
  - 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 286.
  - c. At **Notify**, optionally select an access group to send members a notification email when the filter is activated.
  - 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.”

## 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. A log filter does not, however, apply to adaptor output.

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 panel.
4. At **Result**, select the log filter that you want to run each time the step runs.

## 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 283.

### 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 <i>Production</i> anywhere in the string.

Expression	Matches
^Production	Matches <i>Production</i> at the beginning of the string.
Error:.*[0-9]\$	Matches a line that contains <i>Error</i> followed by any set of characters terminated by a number at the end of the string.
[Ww]arning	Matches <i>Warning</i> or <i>warning</i> .
.*	Matches any character 0 or more times. The dot (.) matches any character, and the asterisk (*) matches 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. For example, consider this line of output:

```
exception retrying exception
```

Using 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*.

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.

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

```
retrying.*exception
```

## 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 283.

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

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 Warning and continues searching the current line for the filter patterns in the set. (Note: the Warning status is a passing status; any pass chains assigned to this step will run.)	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	<p>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.</p> <p>The adaptor relationship list pairs users and objects (such as changed files). For details, see the Adaptor XML Reference.</p> <p>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 email notification.</p> <p>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 the MyFile.c object in the relationship list are sent email notification of the error.</p> <p>Error: Invalid token on line 55 of MyFile.c</p>	not applicable

Filter Action	Definition	Step Results
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 email 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 261.

## 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.

---

## Classes

This topic describes how to create and use classes.

### 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 jobs 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.

## About the Classes panel

Use the Classes panel to add, edit, and delete classes. To view the panel, select **Projects** → **Classes**. The system displays a list of classes. Select a class to edit its properties.

The screenshot shows the 'Classes' panel in a software interface. At the top, there are tabs for 'UI Config', 'Console', 'Reports', and a 'Logout' link for 'Root User'. Below the tabs, there's a 'Classes' section with an 'Add Class' button and a 'Filter' button. A table displays two classes: 'Production' and 'Scratch'. The table has columns for 'Build Class', 'Access', 'Delete', 'Days', 'Count', and 'Which'. Below the table, there's a 'Details' section for editing a class. It includes a 'Name' field, an 'Access' dropdown menu, and several other fields: 'Delete Files', 'Days', 'Count', 'Which', 'Start on purge', 'Start on entry', and 'Start on exit'. Each of these fields has a dropdown menu.

Build Class	Access	Delete	Days	Count	Which
Production	Guest	Console Data	Forever	Unlimited	Any Build
Scratch	Guest	Everything	5	Unlimited	Any Build

**Details**

Name:  Access:

Delete Files:  Start on purge:

Days:  Start on entry:

Count:  Start on exit:

Which:

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

### Properties for deleting jobs automatically:

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.

**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 “Classes” on page 288.

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 servers 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 servers 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.

**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.

\* 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.

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

**Properties for launching projects on 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.

**Note:** If these properties launch a job that contains Must Change variables, the variable values are not changed.

---

## Setting up notification

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

Notifications are sent to access groups, so design your access groups with notifications as well as security in mind. Notifications are always sent to groups, not individual users directly, but you can set up groups that contain only one user, if needed.

If you have set up access groups to be hierarchical (an access group contains subgroups), then notification works as follows:

- If the access group for notification is a parent group and contains one or more users, notification is sent only to the users in the parent group.
- If the access group for notification is a parent group and contains no users, only subgroups, then the notification is sent to all subgroup users.

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 email to the group members when the appropriate event occurs. Any references to times in these email messages are based on the time zone of the Management Console sending the notification.

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 296 for details.

Before you can use notification, you must:

- Configure the SMTP Server system setting so that the system knows what SMTP server to use to send email. 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 emails that the system sends out by editing notification templates. See “Customizing notification templates” on page 293.

## About notification templates

Notification templates provide a means of sending customized messages to users about events in the system. The system includes a number of templates that 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 email 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 294).

### About the Templates panel

Use the Templates panel to set up notification about events. To view the panel, select **Projects** → **Templates**.

The screenshot shows the 'Templates' panel in the Rational Build Forge interface. At the top, there are tabs for 'UI Config', 'Console', 'Reports', and a 'Logout: Root User' link. The 'Templates' tab is active, showing a list of templates. The list has columns for 'Description', 'Project', and 'Step'. Below the list, there is a 'Details' section with three dropdown menus: 'Type' (set to 'Job .break Message'), 'Project' (set to 'HelloWorld'), and 'Step' (set to '- All -'). There are also buttons for 'Add Template', 'Filter', 'Showing 1 - 4 of 13', 'Display All', 'Save', and 'Delete'.

The panel lists all the templates that come with Rational Build Forge. In addition, the panel has the following fields so you can create your own notification templates:

**Type** The type of message

**Project**

The project for which to send the message

**Step**

The particular step in the specified project for which to send the message

## Configuring your SMTP server

To configure your SMTP server, select **Administration** → **System** → **SMTP Server**. The system displays an edit panel 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 200.

## 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.

If you have multiple notifications set within a single step, only the first match results in an email notification.

## Notification exercise

The following procedure describes how to set up and try email notification. The exercise requires an SMTP server and an email account.

1. Set up your SMTP server as described in “Configuring your SMTP server” on page 292  
Make sure you have a user account that sends email to an email 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 emails: one to indicate the start of the project, and one to indicate its success or failure. If you do not receive the emails, verify that you used the correct SMTP server value.

## 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 that 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 email 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 294.

### 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, step, or both. 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.

- **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**, and 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 build purges, describes the type of deletion performed.
BID	Specifies the job ID number. Used to construct links back to the Management Console to access reports.
CONSOLEHOST	The host name of the Management Console computer.
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.

Variable	Contains
DURATION	For steps, specifies the run time in seconds for the step and any steps it inlined.
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 builds, specifies the link name.
MESSAGE	Contains the error or message text for failure or alert messages.
ONFAIL	For steps, holds the continue property for the step.
PATH	For steps, 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 email template leverages.
SELECTOR	Contains the selector name for a step or project.
SERVER	Contains the selector name for a step or project.
SID	For steps, specifies the Step ID number. Used to construct links back to the Management Console to access reports.
SRVRHOST	For steps, 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	For steps, shows the log information for the current step in the job, excluding the environment setup actions that appear in the detailed log.

Variable	Contains
TAG	Contains the tag string for a job. The same value as \$BF_TAG.
TAILNORMALLOG	For steps, works like STEPNORMALLOG, but only displays the end of the log.  For builds, works like FULLNORMALLOG, but only displays the end of the log for each step.  The number of lines displayed is controlled by the <b>Tail Log Amount for Mail Template</b> system setting.
UID	Specifies the User ID number. Used to construct links back to the Management Console to access reports.
USEREMAIL	Contains the email 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 "Failed But Continued."

---

## 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 a snapshot to create 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.

**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

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. The table has columns for these 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 and their settings (log filters, notification groups, and so on)	environments of the project and its steps	project notes
project tags	environments added by an environment variable of type Include for snapshot environments	
	inline projects or libraries and their steps	
	chained projects or libraries and their steps	
	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

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

**Note:** The management console does not display snapshots for projects after the first 2999 projects.

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.
  - 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 <b>Projects</b> or <b>Jobs</b> > <b>Start</b> to display the projects list.
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.





Object	Description
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.
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.



<b>OK</b>	<b>Update references:</b> For any objects that reference the previous default, update references from the previous default project snapshot to the new default.
<b>Cancel</b>	<b>Do not update references:</b> 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.
5. At **Name**, enter the new name.
6. **Optional:** At **Comment**, enter a comment.
7. **Important:** On the popup, choose OK or Cancel.

OK	<p><b>Change the project snapshot name and other snapshot object names:</b> 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.</p> <p><b>For the Base Snapshot:</b> Changes the name of Base Snapshot for all current project snapshots and all future project snapshots.</p>
Cancel	<p><b>Change the project snapshot name but do not change other snapshot object names:</b> 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.</p> <p><b>For the Base Snapshot:</b> Retains the name Base Snapshot for all current project snapshots and all future project snapshots.</p>

## 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 Fast Start icon or the Start Project page to start the default project snapshot.

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

1. Select **Projects**.
2. In the projects list, click the **Fast 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 Fast Start icon or the Start Project page to start a nondefault project snapshot.

To start the nondefault project snapshot by using the Fast 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 **Fast 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 nondefault 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.  
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**.

---

## Chapter 19. 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 issued in order. A step contains one or more commands and has step properties that affect its behavior.

### About the Steps panel

#### Details tab

Step properties specify how to run a step, handle its output, and what to do when the step completes. A step can also run 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.

Step: [ Add New Step ] [ Save Step ]

**Details** [ Notes (0) ]

Name:  Active:  Access:

Directory:  Path:

Step Type:  Inline:

Command:

Environment:  Selector:  Broadcast:

Timeout in minutes:  Result:  On Fail:

Thread:  Pass Chain:  Pass Wait:

Pass Notify:  Fail Chain:  Fail Wait:

Fail Notify:

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 run. The system automatically creates a unique directory for every job. The **Directory** field provides a

convenient way to run 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 run in a path found by adding together the server, project, job, and step directories.
- **Absolute:** Step commands are run 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 run. This property affects the contents of Command and the project specified in Inline, if any.

- **Regular:** The step is run once.
- **Conditional:** The step is run 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 run. Instead, the Else Command and Else Inline are run if they are specified.
- **While Loop:** The step can be run multiple times. It is run 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.

The selector is evaluated each iteration of the While Loop to determine the server to use for the iteration.

**Inline** Specifies a project or library to run inline with the current project. The steps from the project or library are run using the environment and most of the properties of the current project. However, the system uses the inline project's selector as the default selector for the steps of the inline. 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 run 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**

One or more commands. The commands can be operating system commands, dot commands, or a combination of both. See “How steps run” on page 311.

**Condition**

Shown only if you have selected a step type of Conditional or While Loop.

- Conditional: the command is run if the condition evaluates to true.
- While Loop: the command can be run multiple times as long as the condition evaluates to true. You can set the limit using Max Iterations.

A condition can be a 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 about how to use them, see “Condition functions” on page 316.
- 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 “Interpretation of variables in steps” on page 250 for more information about 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 run 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.

**Selector**

Specifies a selector to use to choose a server for this step. If left as **Default**, the step runs 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 runs them serially or in parallel, depending on the broadcast step's **Thread** property.

**Timeout in minutes**

Specifies how many minutes the system waits for the current command to produce output (default is 5 minutes). A value of 0 means that the step does not timeout if the step properly connects to the agent. 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. (A step with a "Warning" status is counted as passing and will launch a pass chain).

**Pass Wait**

If checked, the system suspends the current project until the pass chained project completes. If this step (or its project) is canceled, the chained project is also canceled. If it is not checked, the chained project is started asynchronously and the current project continues to the next step.

**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. (A step set to continue on failure is counted as failing, and will launch any fail chains assigned to the step.)

**Fail Wait**

If checked, the system suspends the current project until the fail chain project completes. If this step (or its project) is canceled, the chained project is also canceled.


**Notes tab**

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:

1. Click the **Notes** tab.
2. Write the new note in the text field.
3. Click **Submit**.


To edit a note:

1. Click the **Notes** tab.
2. Click the **Edit** icon  next to the note you want to edit. Make your edits.
3. Click **Submit**.

To delete a note:

1. Click the **Notes** tab.



2. Click the **Trash can** icon . A prompt asks if you are sure that you want to delete the note.
3. Click **OK**.

---

## Adding a step

### About this task

This procedure adds a step to the end of the project. For information about how to insert a step as the first step or between existing steps, see “Additional step operations” on page 310.

### Procedure

1. Select **Projects**, and then click on a project name. The system displays the list of steps for the selected project.
2. Click **Add Step** at the top of the main panel. The system displays an empty step details panel.
3. Enter values for the properties. **Name** and **Command** are required.
4. Click **Save Step**.

---

## Editing a step


### Procedure

1. Select **Projects**, and then click on a project name. The system displays the list of steps for the selected project.
2. Click the step name. The system displays the step's properties in the lower portion of the panel.
3. Make your edits to the properties.
4. Click **Save Step**.

---

## Disabling a step

### Procedure

1. Select **Projects**, and then click on a project name. The system displays the list of steps for the selected project.
2. 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 run 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**, click the job name, and then click **Job Steps**. The step is visible but cannot be made active for the job you are starting.

### Note:

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.

---

## Additional step operations

To work with steps, select **Projects**, and then click on a project name. The system displays the list of steps for the selected project.

Click the **Actions** icon  in front of the step name to display additional options:

- **Insert New Step:** Insert a step above the selected 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 COPY 0. 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:
  - 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
- **Delete Step**

---

## 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. The steps for the inlined project or library are indented in the step log.
- **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 run steps marked for threading in parallel. Use the Thread property of a step to mark it for threading.
- **Broadcasting:** You can run 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 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.

## How steps run

In a step definition, the command property contains operating system commands, dot commands, or a combination of both.

You can run 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.

Before the system runs a step, it constructs the step environment. Variables are set using values specified in the server environment, project environment, and step environment. See “Environment inheritance” on page 247. By default, the variables are parsed and then available to use in commands. See “Interpretation of variables in steps” on page 250.

## Specifying a shell

You can use the `#!` directive to specify the shell to use to run the commands. This works on Windows<sup>®</sup> as well as Linux<sup>®</sup> and UNIX<sup>®</sup> 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 computer.

**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.

## How the system splits a step into parts

Rational<sup>®</sup> Build Forge<sup>®</sup> splits a step into parts, with each part formed by a set of operating system commands or a single dot command. For example, the following step has six parts.

```
cmd1 # Part 1
cmd2
.dot_cmd1 # Part 2
```

```
cmd3 # Part 3
cmd4
.dot_cmd2 # Part 4
.sleep 30 # Part 5
.dot_cmd3 # Part 6
```

The step parts run in sequence. The environment from each part is passed to the next part. When an error occurs in any part, the processing stops immediately.

Previous releases required each part from the step above to be a single step.

**Note:** Do not create references between parts. Do not create jumps between parts, such as using a GOTO in one part and its destination label in another part.

## Inlines: including the steps of a project or library

Use the Inline property of a step to include all of the steps from a specified project or library.

To use an inline in a step, set the Inline property to the name of a project or library. When the step runs, the following happens:

1. The step runs the command or commands in the Command property.
2. The step runs the steps of the project or library that is specified in the Inline property.

If you want to run the inlined steps but do not have a use for Command property, use `.sleep 0` in the Command property.

### Inline inheritance from the calling step

All steps from the called project or library are run in the context of the calling step. The inlined steps inherit the environment of the calling step.

However, the system uses the inline project's selector as the default selector for the steps of the inline.

### Effects of the inline steps' status on the calling step's status

If the command for the calling step passes, then the status of the execution of the inlined steps is considered, as follows:

- If the command for a step fails, the job normally stops. However, if the On Fail property for the step is set to Continue and an Inline is specified, the inline runs.
- If all steps return an execution status of Pass, the calling step is marked Pass. If a Pass Chain is specified for the calling step, it runs.
- If any of the inlined steps return an execution status of Fail, the calling step is marked Fail. If a Fail Chain is specified for the calling step, it runs.

This behavior allows you to trace execution status easily through deeply nested inlines and chains.

### Inline nesting

When you inline the steps of a project or library, the called steps are nested in the calling step.

The maximum level of nesting is set by the system setting **Max Inline Depth**. The default is 32. The nesting is not tested at job start time. If a running job exceeds the limit, it fails at the point where the limit was exceeded.

The level of nesting may also encounter limits based on the available memory on the host running Management Console.

## Pass Chains and Fail Chains for steps

Individual steps can have a Pass Chain and a Fail Chain.

A step's Pass Chain and Fail Chain are run independently of the project's Pass Chain and Fail Chain.

The Pass Chain and Fail Chain properties are set to the name of a project or library. They work in the same way as chains set for a project. See “Canceling chained projects when wait enabled” on page 278.

## 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*. Thread blocks can continue into steps that are part of an Inline. For example, if a step in a project contains an Inline and the first step of that Inline is also threaded, the two steps are part of the same thread block. They run concurrently. The thread block follows threaded steps, including nested Inline steps, until a Join step or a nonthreaded step is encountered. Take care to avoid race conditions when using nested Inline steps. A race condition can be caused by an inlined threaded step depending on results or data from the threaded parent step.
- A thread block is terminated by a step whose Thread property is set to **Join** or when it encounters a nonthreaded step. 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
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 nonbroadcast 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 run 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 run 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 that run 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 (a project 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, 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:

1. Set the step type to Conditional.
2. Set the condition to an expression that can be evaluated.
3. Fill in the command to be run.
4. If desired, specify an inline to run. (You can leave Command blank if Inline is set.)

If Condition evaluates to true when the job runs, the step is run. If specified, the Inline project or library also runs. 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.)

During job execution, the step result is marked Pass if the condition is evaluated successfully and the commands in Command or Else command are run 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:

1. Set the step type to While Loop.
2. Set Condition to a command or an expression that can be evaluated.
3. Fill in the command to be run.
4. If desired, specify an inline to run. (You can leave Command blank if Inline is set.)
5. If desired, set Max Iterations to the maximum number of times you want the step to run. Use this limit during development until you are satisfied that the condition you specify works correctly. The default is 100.
6. 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 run 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 runs. *Important:* the condition function must be in double quotes in the condition attribute: `condition="condition_function"`.

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* contains 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.

**Note:** Before Rational Build Forge version 7.1.2, this function returned true if string *a* was in string *b*.

**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.

***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>=b*** Tests that *a* is greater than or equal to *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 or equal to *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

## Launching projects from steps

Use the .run or .runwait command to launch a project from a step.

A project launched this way acts like a chain. The project runs using its own selector and environment. See “.run and .runwait” on page 344.

---

## Customizing log output

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

### Labeling log output for a step

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

#### Before you begin

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).

#### About this task

You can include an uppercase label at the beginning of any line of output. The label is used until the step ends or a new label is encountered.

A label has the following syntax:

- Contains only uppercase letters followed by a colon (Labels cannot contain spaces, punctuation, numbers, or lowercase letters. "SPACESHIPS:" is valid. "Space Ships:" is not.)
- Has at least three characters

You can create a label using the **echo** command in a step. The system recognizes the first argument to the **echo** command as a label if it follows the label syntax.

**Note:** The system recognizes text at the beginning of a line in any build output as a label if it follows the syntax, even if that text is not an argument to the **echo** command.

As the example shows, you can set the label to an existing output label name as well as new names.

#### Procedure

1. Create a new step in a project. The project in this example is named Say\_hi.
2. Name the step LabeledLogOutput.
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.

#### Results

Note the SPACESHIPS checkbox in the category header and output lines 354 and 355, which are labeled SPACESHIPS.

```
79 7/9/10 10:24 AM EXEC Locale set to 'English_United States.1252'
250 7/9/10 10:24 AM EXEC Locale set to 'English_United States.1252'
349 7/9/10 10:24 AM EXEC Performing variable expansion on command line
353 7/9/10 10:24 AM EXEC start [C:\data\BuildForge\buildforge_projects\Say_hi\BUILD_1@RBF-14]
354 7/9/10 10:24 AM SPACESHIPS Voyager I
```

```

355 7/9/10 10:24 AM SPACESHIPS Voyager II
356 7/9/10 10:24 AM EXEC You can add text to existing categories as well
357 7/9/10 10:24 AM EXEC end [C:\data\BuildForge\buildforge_projects\Say_hi\BUILD_10RBF-14]

```

**Note:** You can select or clear the SPACESHIPS check box to show or hide the category.

**Note:** When adding text to an EXEC category, be sure to avoid special characters (for example, parenthesis). On some systems, these can be viable if surrounded by quotes, but in general it is a better practice to simply use plain text.

## 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
- [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 it opens a new browser window or tab and shows the page. Please note, however, that a log filter matching part of the [URL] address will cause the link to become nonfunctional.

---

## Working with job data

These topics show 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:

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

```
.strsub _MAJORVERSION_ $MAJORVERSION README.TXT
.strsub _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.
 - 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 <EnvGroupName>[(<SnapshotName>)] "<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. The value of the variable does not persist past the current 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 `.tset` command to create a new variable during a job.

Setting multiple variables

You can set more than one variable at the same time 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"
.tset env "Z=9" "Z2=54"
```

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`"
```

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.

The case-sensitivity of your underlying database determines the cases you can use when you create register names.

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 341), 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.

Table 11. Special registers

| Register | Contains |
|----------|--|
| ! | Contains the command output lines that matched Fail filter patterns.
Note: This register is visible only in the scope of the step in which the filter is applied. After Rational Build Forge processes the step with the filter, the contents of the register that were potentially set by that step filter are no longer visible. |
| @ | Contains the command output lines that matched Pass filter patterns.
Note: This register is visible only in the scope of the step in which the filter is applied. After Rational Build Forge processes the step with the filter, the contents of the register that were potentially set by that step filter are no longer visible. |
| = | 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: <ul style="list-style-type: none"> • 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 **Edit** icon  next to the desired project's name. The project properties appear in the lower portion of the panel; click the **Registers** tab to display the project's registers. The tab provides a panel for managing registers:
 - To create a new register, enter a name and contents, and 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 panel.
 - To edit a register, click the **Edit** icon  next to the register's name in the list. The system populates the register panel with the register's contents. Make your changes, and then click the **Save Edited Register** button.

Anyone who has access to a project can view and edit its project registers.

Note: The case-sensitivity of your underlying database determines the cases you can use when you create register names.

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` commands (to copy whole directory trees).

Important: The server resources must have file copying enabled. It is not enabled by default. See “Enabling file copying on a server resource.”

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:

- Reference entry for `.get`
- Reference entry for “`.rget`” on page 343
- Description of how paths in steps are resolved, in “Working directories for jobs” on page 367

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`
- Reference entry for `.rput`
- Description of how paths in steps are resolved, in “Working directories for jobs” on page 367

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 ran affects the step result status. When the server runs 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.

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 283.

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 after the batch file runs. 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 run 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.

There is a separate list of dot commands for use with environment variables. See "Using dot commands in variables" on page 257.

Dot command syntax

Syntax specifications: each dot command description includes a syntax specification, using the following notation:

- 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.

If a dot command accepts environment variables for parameters, you can also use:

- Backticks (`) for command execution
- Square brackets ([and]) to indicate a range (Also known as range notation.)

Consider the following .set example:

```
.set envgroupname "F00[1,3-5]=`cat foofile`"
```

In this case, the agent runs `cat foofile`, which lists the contents of `foofile`. Using range notation, lines 1 and 3 through 5 are extracted. The `.set` command then uses those lines to update the existing variable 'FOO' within the environment group named `envgroupname`.

.bom

```
.bom addcategory "category"
.bom setcolumn "category" "section" "column" [...]
.bom data category "section" "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. Use the `addcategory` option to specify them.

```
.bom addcategory "category"
```

Sections

A section defines columns of data within a category. Use the `setcolumn` option to specify sections and columns within the sections. 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 by using the `-p` option and identifying the parent section.

Sections and columns must be defined before data options attempt to add data to them.

```
.bom setcolumn "category" "section" "columnheader" [...]
.bom setcolumn "category" "section" -p parentsection "columnheader" [...]
```

Data Data populates the columns defined in a section. The section and columns must already be defined using `setcolumn`.

```
.bom data category "section" "column=value" [...]
```

The following example shows the order in which categories, sections and columns, and data must be specified.

```
.bom addcategory "Spaceships"
.bom setcolumn "Spaceships" "Section1" "ShipName" "WarpSpeed" "Tonnage"
.bom setcolumn "Spaceships" "Subsection1" -p "Section1" "ShippingDate" "ShippingManifest"
.bom data "Spaceships" "Section1" "ShipName=SpaceShipOne" "WarpSpeed=9" "Tonnage=10000"
.bom data "Spaceships" "Subsection1" "ShippingDate=123" "ShippingManifest=456"
.bom data "Spaceships" "Section1" "ShipName=Freighter" "WarpSpeed=6" "Tonnage=20000"
```

This example shows in the BOM as follows:

| ShipName | WarpSpeed | Tonnage | | |
|--------------|-----------|---------|--------------|------------------|
| SpaceShipOne | 9 | 10000 | | |
| | | | ShippingDate | ShippingManifest |
| | | | 123 | 456 |
| Freighter | 6 | 20000 | | |

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.

If you omit a column from a data line, the system uses the value from the previous row, as the following example shows.

```
.bom addcategory "Spaceships"  
.bom setcolumn "Spaceships" "Section1" "ShipName" "WarpSpeed" "Tonnage"  
.bom data "Spaceships" "Section1" "ShipName=SpaceShipTwo" "WarpSpeed=3" "Tonnage=30000"  
.bom data "Spaceships" "Section1" "ShipName=Tanker" "Tonnage=50000"
```

The result is that the WarpSpeed value from SpaceShipOne is repeated:

| ShipName | WarpSpeed | Tonnage |
|--------------|-----------|---------|
| SpaceShipTwo | 3 | 30000 |
| Tanker | 3 | 50000 |

.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 <code>_BOM</code> : <code><build_tagname>_BOM.xml</code> . |

Examples

```
.bomexport  
.bomexport myproj.xml  
.bomexport path/to/myproj.xml  
.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 Stopped. 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 email message to the specified access group when it stops the job.

.bset

```
.bset env "<VarName>=<Value>" [...]  
.bset selector <SelectorName> [<SelectorSnapshotName>]  
.bset server <ServerName>  
.bset buildserver <ServerName>
```

The `.bset` command changes project settings temporarily during a job.

Note: The `.bset` command affects the base build environment for later steps that use that environment. Do not specify a `.bset` command that relies on a previous `.bset` command in the same step. For example, do not specify steps similar to the following combination in the same step.

```
.bset env "VAR1=VALUE1"  
.bset env "VAR2=$VAR1"
```

The command has the following options:

- **env** changes the value of one or more project environment variables for a running job. The change takes effect immediately in the current step. For more information about using the command this way, see “Changing the build tag during a job” on page 321. 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 (after any Inline steps for the current 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). If you use the **selector** option in an Inline step, the option affects any steps that the Inline step inlines and any subsequent steps in the inlined project; it does not affect the calling step or any other steps at or above the level of the calling step.

Steps that have an explicit selector set are not affected.

Use the optional `<SelectorSnapshotName>` to specify a snapshot of the specified selector.

- **server** changes the default project server during a job. The new server setting takes effect in the next step (after any Inline steps for the current step). 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 following command runs the `SelectAServer.sh` script and provides its output as the server name for the `.bset server` command:

```
.bset server `SelectAServer.sh`
```

- **buildserver** changes the default project server during a job. The new server setting takes effect in the next step (after any Inline steps for the current step). It differs from the **server** option because it applies to inlined steps as well as project-level steps.

Note: Avoid using multiple `.bset` commands in threaded steps.

.buildstatus

`.buildstatus <result>`

This command forces the build to have the specified `<result>` after the build has finished, regardless of the results of the build's steps.

Valid values for `<result>` are:

- P** Sets the result to Passed
- F** Sets the result to Failed
- W** Sets the result to Warning
- B** Sets the result to Stopped (as if the `.break` command had been specified)

.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.

Note: The `.date` command is re-evaluated every step. To preserve a specific time, use `.date` with `.bset env`. For example, use `$ORIGTIMESTAMP = .date %d-%b-%Y.%H:%M:%S` and then `.bset env "TIMESTAMP=`echo $ORIGTIMESTAMP`"`.

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.

| 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. |

| Date Conversion Specifier | Description |
|---------------------------|--|
| %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 run 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 to run by using the `entry_name` option. The `entry_name` must match the name attribute specified for an `<interface>` element in your adaptor template. If the `<interface>` element specified in `entry_name` does not exist or cannot be found, the default `<interface>` element is run instead. In the following example, the entry name is `DefectFunction`.

If you are using an adaptor link, the adaptor is called automatically and the first interface function in the adaptor template is run. To run a different interface, in the adaptor template, set the default attribute to `true` (`default="true"`) on the interface that you do want to run.

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:

```
<bfinstall>/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 the system runs 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 `$n` syntax. `$1` refers to the first value in the group; `$2` to the second value in the group, and so on. 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
```

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 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 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 and a period to match a literal period character. Example:

```
\\\\.
```

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 <recipient>
```

This command sends an email to *recipient*, 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 and its 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 runs from the current server/path and retrieves the file from the specified server/path. The destination path name is relative to the step's current working directory. The source path name is relative to the specified server base path. Server must specify a logical server that allows the .get operation for files (see "Enabling file copying on a server resource" on page 325). 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 367.

If the server name you are using has spaces in it, put the server name and the path to the file in quotes. Example: .get [<relative_path>/]"file server:[<relative_path>/]file]"

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 ran 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, and 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 run 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 run.

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 runs the project that is labeled primary in the file. This project has attribute `primary="1"` on its `<project>` element.

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 get the current definition for that project; when a step that is imported by the `.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.

Note: Step XML generated from Build Forge versions previous to 7.1 is not supported and will likely fail, as those elements are referenced by name rather than ID.

Note: `.load` uses the import utility, and as such, applies access group ownership according to the import utility's settings. See "How access groups are assigned to imported objects" on page 403 for further information.

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, and 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 runs 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.

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 a project ID within the database. This allows you to create your own .load files without having to include inlined projects within them.

Note: Using .load with XML files generated by Build Forge versions 7.0.x and older is not supported and will not work properly, as the older syntax references objects by name rather than UUID.

- 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
- The commandStore attribute contains a copy of the contents of the <command> element, with certain characters XML-escaped. For example, in <command> you might have ", ', &, <, >, or a newline. The characters would be represented in commandStore respectively by ", ', &, <, >, and
.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<buildforge schema="7.115014" 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" commandStore="echo Hello World">
  <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"
  commandStore=".export $BF_PROJECTNAME_PHYS.xml&#10;copy /Y $BF_PROJECTNAME_PHYS.xml ..\">
  <command>.export $BF_PROJECTNAME_PHYS.xml
copy /Y $BF_PROJECTNAME_PHYS.xml ..\"</command>
</step>
</project>
<project access="6" active="Y" name="Sleepytime" primary="0"
  selectorId="Choose_local" maxthread="0" increment="Y" tagsync="0"
  buildclass="Production" sticky="N" envId="0" tag="SLEEP_$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" commandStore=".sleep 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.

.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 run 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 to run by using the `entry_name` option. The `entry_name` must match the name attribute specified for an `<interface>` element in your adaptor template. If the `<interface>` element specified in `entry_name` does not exist or cannot be found, the default `<interface>` element is run instead. 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 run. To run a different interface, in the adaptor template, set the default attribute to `true` (`default="true"`) on the interface that you do want to run.

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:

```
<bfinstall>/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`

Register A is written to the file `data.txt`, located in the step's working directory.

- `.pop ver +data.txt`
The contents of register `ver` are appended to the file `data.txt`.
- `.pop Alpha`
The contents of register `Alpha` are written to the step's log.
- `.pop ALPHA > BETA`
Makes the contents of register `BETA` the same as the contents of register `ALPHA`.
- `.pop A >> B`
The contents of register `A` are appended to register `B`.

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 runs 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. To create such a job, make a `.purge` command the first step in the project and make 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's working directory are put in register Alpha.
- `.push +B ..\newdata.txt`
The contents of the file `newdata.txt` in the parent directory of the step's working directory are appended to register B.
- `.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 runs 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 325). Only single files can 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 367.

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.

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 run is taken as the remainder of the arguments and thus the `.retry` command must be the final dot command for a step. For example, consider this command:

```
.retry 3 myscript.sh arg1 arg2 arg3
```

It runs “myscript.sh arg1 arg2 arg3” up to three 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.

Note: The specified directory must exist before attempting to use `.rget`. If the specified directory does not already exist, the command fails.

.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 367 for more information about how the system constructs paths.

Note: You cannot use environment variables in this command.

Note: The specified directory must exist before attempting to use `.rput`. If the specified directory does not already exist, the command fails.

.run and .runwait

```
.run [-c "<condition>"] "<ProjectName>" ["<ProjectSnapshotName>"]
```

```
.runwait [-c "<condition>"] "<ProjectName>" ["<ProjectSnapshotName>"]
```

You can use the `.run` and `.runwait` commands to launch a chained project from a step command. To specify a snapshot of the project, use the optional `<ProjectSnapshotName>` parameter.

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. The launching step waits while the launched project completes. When the launched project completes, the system sets the result value of the launching step to pass or fail according to the completion status of the launched project.

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.

Important: Any project launched via the `.run` or `.runwait` commands does not produce a chain link icon in the Build Results page.

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.

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.
```

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's working directory when the command runs. 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. 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 about using these commands, see “Adding baselines and checkpoints with the .scan command” on page 359.

.semget

```
.semget <semaphore_name>
```

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 496 for more information about using this command.

.semput

```
.semput <semaphore_name>
```

Use the .semput command to release the semaphore with the name <semaphore_name>. See “semaphore” on page 496 for more information about using this command.

.set

```
.set env <EnvGroupName>[(SnapshotName)] "<VariableName>=<DesiredValue>" [...]
```

The .set command assigns a value to an environment variable. You can specify additional variables and values. Enclose each variable and its value in quotes. This command sets the variable for the default snapshot for the environment group unless you specify a snapshot. If you specify a snapshot, enclose it in parentheses with no space between the environment group name and the snapshot name.

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 runs 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 runs on a *specified* environment, 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 about using this command, see “Working with job data” on page 320. Also see the similar command “.bset” on page 330.

.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.

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, steps, and tags 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.

Option	Description
-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.
-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 run 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 to run by using the `entry_name` option. The `entry_name` option must match the name attribute specified for an `<interface>` element in your adaptor template. If the `<interface>` element specified in `entry_name` does not exist or cannot be found, the default `<interface>` element is run instead. 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 run. To run a different interface, in the adaptor template, set the default attribute to true (default="true") on the interface that you do want to run.

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:

```
<bfinstall>/interface
```

.stop

`.stop <state of build>`

`.stop` forces the build to immediately stop processing. Use this command to terminate a build. The possible states of build are: (P)ass, (F)ail, (W)arn, or (B)reak.

.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 files for the `<source>` string; where a match is found, the system replaces the `<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 fails:

```
.strsub _VERSION_ 2.34 *.txt
```

However, you can use variables in the command, so a command such as the following works 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.

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 only the first instance 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 can use the `/x` flag to suppress backslashes:

.strsub/x

.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 run 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 an <interface> element in your adaptor template. If the <interface> element specified in *entry_name* does not exist or cannot be found, the default interface element is run instead. 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 run 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:

```
<bfinstall>/interface
```

.tset

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

The .tset command changes project settings temporarily during a step. You can use .tset to set a variable that does not yet exist.

The command takes effect in the current step. It takes effect for all commands in the step and for any Inline specified for the step. However, threading can affect this behavior. Example:

1. A step is threaded and it specifies an Inline.
2. The first step of the Inline is also threaded.

In the example, the .tset command takes effect for the first step (1), but not the first step of the Inline (2). Both steps are part of the same thread block. They run independently. See “Threading: running steps in parallel” on page 313.

The values set by the .tset command are written to the job record. They do not update the database record for the environment. Later jobs are not affected by the changes.

.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 page you reach through **Jobs** → **Completed** and is removed from the list on the Locked page.

Chapter 20. Working with jobs

This topic describes how to run, view, and manage jobs in the Management Console.

About jobs

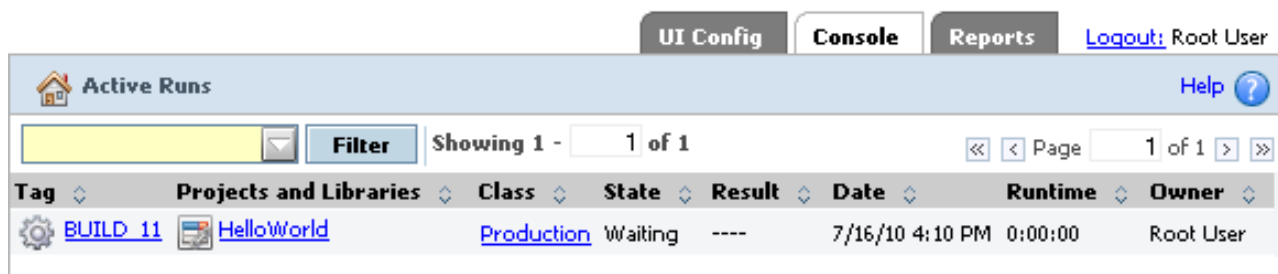
A job is a project that is executing or has finished executing.

You can monitor jobs using the **Home** panel and the **Jobs** panel. In addition, you can use the **Jobs** panel to start jobs, view job results, and manage semaphores.

About the Home panel

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

To access the **Home** panel, in the left menu, click **Home**.



Active Runs							
Filter		Showing 1 - 1 of 1		Page 1 of 1			
Tag	Projects and Libraries	Class	State	Result	Date	Runtime	Owner
BUILD_11	HelloWorld	Production	Waiting	---	7/16/10 4:10 PM	0:00:00	Root User

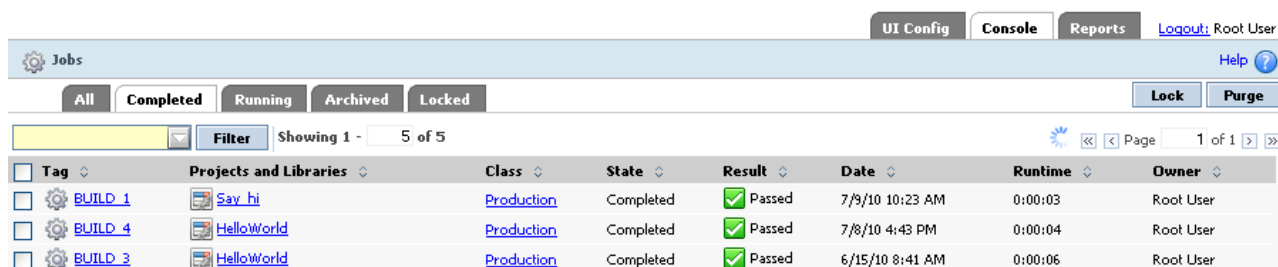
Use the **Home** panel to view your recent or current jobs. You also use it to view notifications and system messages. Select a menu item to view the following:

- **Active Runs** - your currently executing jobs
- **Completed Runs** - your completed jobs
- **System Messages** - the system message log (To view messages though, use **Administration** → **Messages**.)

About the Jobs panel

Use the **Jobs** panel to work with running jobs and to view job results.

To access the **Jobs** panel, in the left menu, click **Jobs**.



Jobs							
Filter		Showing 1 - 5 of 5		Page 1 of 1			
Tag	Projects and Libraries	Class	State	Result	Date	Runtime	Owner
BUILD_1	Say hi	Production	Completed	Passed	7/9/10 10:23 AM	0:00:03	Root User
BUILD_4	HelloWorld	Production	Completed	Passed	7/8/10 4:43 PM	0:00:04	Root User
BUILD_3	HelloWorld	Production	Completed	Passed	6/15/10 8:41 AM	0:00:06	Root User

Jobs tabs

With the Jobs tabs, you can 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 times. All completed jobs follow, sorted by their completion times. The check boxes are greyed. You can click any of the links. No special actions are available.

Completed

All completed jobs are presented, sorted by their start times. Running, archived, and locked jobs are not presented. You can purge or lock completed jobs. Select the check boxes for the desired jobs, and 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 times. You can cancel one or more running jobs. Select the check boxes for the desired jobs, and then click **Cancel**.

Archived

All archived jobs are presented, sorted by their start times. 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, and then click **Purge**.

Locked

All locked jobs are presented, sorted by their start times. You can unlock one or more locked jobs. Select the check boxes for the desired jobs, and then click **Unlock**.

Note: You cannot delete locked jobs directly. To delete them, first unlock them, go to the **Completed** tab, and then use **Purge**.

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:

Tag The build tag, shown as a link. Click it to see the logs for the job steps.

Projects and Libraries

Shown as a link. Click it to see the project used for the job.

Class Shown as a link. Click it to see the class definition of the class used for the job.

State One of Waiting, Running, Completed, Archived, or Locked.

- The Waiting state means that the job has not been started, either because the Running queue is full or because the job is waiting for a semaphore. The Running queue size is defined by the system setting **Run Queue Size**.

- Jobs in the Running queue normally are executing. However, if the Running queue contains multiple jobs for the same project, whether one or more of them is currently executing depends on the **Run Limit** property in the project. See “Changing project properties” on page 274.

Result One of Passed, Failed, Failed But Continued, Warning, Stopped, or Canceled.

Date Before the job starts: The date and time the job was requested.
After the job starts: The date and time the job was started.

Runtime

For running jobs, the current elapsed time. For completed jobs, the total elapsed time.

Owner

The user who started the job.

Filtering and sorting the jobs list

You can filter and sort the jobs list in any tab.

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, and then click **Filter**.
 - Existing filter: Click the arrow next to the box, and 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.

Before you begin

A project must have an environment and a selector defined to run as a job.

Procedure

- *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.

Job DetailsJob Steps

Project Parameters

Snapshot:Base Snapshot
Selector:My selector
Class:Production
Tag Format:BUILD_\$B
Tag Example:BUILD_9

Project Environment


Job will start with the default environment.

Save Environment

Project Tags


Editable Tags

B9


- 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 disabled: .

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.
- 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.
 - Select the Job Steps tab to display the list of project steps. You can select individual steps to exclude them from this run only.

After you make your choices, click **Execute** to start the project.

Results

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

About this task

You can view job results in all job tabs.

Procedure

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.
 - Result: The step result, Passed, Failed, Canceled
 - 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)
 - Chains: If a step started another job, a link to that job is shown.
 - A menu appears at the bottom of the left, side menu, showing the following items:
 - Results: Shows the list of steps in the main panel.
 - 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 (any jobs launched with .run and/or .runwait will not produce this icon).
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**), and then click the step. When viewing a step log, click **Display All** to show all the steps. To view fewer steps, enter a value for *m* in the **Showing 1 - m of n** control and press **Enter**.

5. To filter the step log: Check or clear categories, and then click **Refresh**.

```
1 7/8/10 4:43 PM STEP      Step using selector 'My selector'.
2 7/8/10 4:43 PM MANIFEST  BF_LAST_UPDATE=1278618783
3 7/8/10 4:43 PM MANIFEST  BF_NAME=rbf-14
4 7/8/10 4:43 PM MANIFEST  BF_LOADRATIO=0.3333333333333333
5 7/8/10 4:43 PM MANIFEST  BF_JOBS=1
6 7/8/10 4:43 PM MANIFEST  BF_AGENT_VERSION=7.1.2-0-0008
7 7/8/10 4:43 PM MANIFEST  BF_LAST_REFRESH=1278618783
```

RSS data feed for jobs status

You can 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.

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 Job** button near the top of the panel.

1. Click **Restart Job**; the system displays a Restart page.
2. Select options. Set the **Sync Commands** property if you want the system to get any updates to the commands in its steps from the project record; if you do not set 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.
- 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.

For information about the restart of jobs that have adaptor link steps, see “About adaptor links” on page 436.

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 run 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 run 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.

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 439 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 run, 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 346.

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 runs.

.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 is 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	.scan baseline	config.c execute.c	Baseline: config.c execute.c
3	Add data file	config.c execute.c data.txt	
4	.scan checkpoint	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	.scan checkpoint	config.c execute.c 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 <bfinstall>/Platform directory on UNIX and Linux.

Description

Use the `bfomexport` command to export the Bill of Materials (BOM) for a job to an XML file. After collecting BOM information, `bfomexport` 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.

You must run the `bfomexport` command from the Build Forge installation directory for the Management Console and from the `/Platform` directory on UNIX or Linux.

Syntax

```
bfomexport [-f filename] [-p projectID | -P projectName]
           [-b buildID | -t buildTag] [-L] [-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 <code>bfexport</code> command with the <code>-l</code> option lists project IDs.)
-P projectName	The name of the project.
-b buildID	The build ID.
-t buildTag	The build tag name.
-L	Include step logs.
-H	Help message.

Scheduling jobs

Use the **Schedules** panel to schedule projects 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.

To view job schedules, click **Schedules**. A calendar for the current month is shown, with a panel beneath it used to create and modify schedule entries.

Each day in the calendar displays the number of projects scheduled for that day. Mouse over a day to see the names and schedule parameters for jobs that are scheduled for that day.

If you have more than one project scheduled, the system displays a dropdown box and **Filter** button to allow you to filter the calendar by project.

The screenshot shows the Rational Build Forge Schedules interface. At the top, there's a navigation bar with 'Schedules' and 'Add Scheduled Run' buttons. Below this is a 'Calendar' view for July 2010. The bottom section is the 'Schedule Details' form, which has two tabs: 'Schedule Details' and 'Environment'. The 'Schedule Details' tab is active, showing fields for Description, Access, Owner, Project, Class, Mode, Environment, Auto-Sync Environment, Selector, Minutes, Hours, Dates, Months, and Days. The 'Environment' tab is also visible, showing fields for Environment, Auto-Sync Environment, and Selector.

To place a project in the scheduler, do the following:

1. Click **Schedules** in the left menu.
2. Click **Add Scheduled Run**.
3. Enter a description for the schedule entry in **Description**.
4. Select a project from the **Project** list.
5. The current snapshot for the project is shown in a field underneath **Project**. If you want a different snapshot to run for the scheduled job, select a snapshot from the list.
6. For **Mode**, choose Active (the default), Inactive, or Once. If set to Once, the job runs only once, at the next time the time settings match. If set to Active, the job runs each time the time settings match.
7. Optional: you can override the following project properties. Settings you choose here are in affect only for this instance of the scheduled job.
 - **Access:** if specified, the scheduled job uses the indicated Access property.
 - **Owner:** if specified, the scheduled job runs as if manually launched by the indicated Owner.
 - **Class:** if specified, the scheduled job uses the indicated Class.
 - **Environment:** A copy of the environment in this field is made for this scheduled job, whether it is set explicitly or left at Project Default. If the environment is specified, you can also set starting values for the variables.

Important: After the copy is made, changes to the original environment and its variables are not automatically updated in the copy of the environment made for the scheduled job. You can update the copy to reflect changes made to the original environment manually or automatically. For a manual update, click **Resync Environment**. For automatic updates, set **Auto-Sync Environment** to Yes. With automatic updates, the environment is updated every time the scheduled job runs. Be aware though that the Environment tab is only available for schedules that have **Auto-Sync Environment** set to No.

- **Selector:** if specified, the job uses the specified selector.

8. Specify the time settings. Fill in values for Minutes, Hours, Dates, Months, and Days. See “Schedule parameters” for the values to use.
9. Click **Save Schedule**.
The Next Run column displays Calculating, and then displays the next scheduled time that the job will run.

After the schedule is created, the schedule name appears in the calendar. An icon next to it displays its Mode. You can change the mode by clicking the icon. It changes each time it is clicked.

- Green circle: Active
- Blue circle: Once
- Red circle: Inactive

When a scheduled job attempts to run, the system checks the job queue, the **Hard Run Limit** system setting, and potentially the **Run Limit** property of the project. If one or more jobs of a project are already running when the schedule activates, the system behavior depends on the **Hard Run Limit** system setting, as explained in the following table.

Hard Run Limit value	System checks Run Limit value?	Systems launches job?
Yes	Yes	Yes if the number of running jobs is less than the Run Limit value No if the number of running jobs equals the Run Limit value
No	No	Yes

Note: Set the **Run Limit** value to 1 and **Hard Run Limit** to Yes to skip a run if the prior run has not completed.

Schedule parameters

This section describes the parameters that you can use to specify when to run a job.

You use a series of fields to specify the times for a job to run.

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 runs 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 runs 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 asterisks 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.

Days/Dates examples:

Values					
Desired schedule	Minutes	Hours	Dates	Months	Days
Run job at 1:01 a.m. on the first day of each month. Dates uses a literal value; Days uses an asterisk.	1	1	1	*	*
Run job at 1:01 a.m. on every Monday in the month. Dates uses an 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 Dates and Days use a literal value.	1	1	1	*	1
Run job at 1:01 a.m. every day. Both Dates and Days 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:

1. Create a schedule as usual, but select “Class Purge Schedule” in the Project field.
2. Select a class in the Class field.

Administering jobs

You can lock, archive, and delete jobs.

Locking jobs

You can lock a project to prevent it from being automatically deleted.

To lock a job, click the **Completed** tab in the **Jobs** panel, select a build, and then click **Lock**.

To unlock a job, click the **Locked** tab in the **Jobs** panel, select a locked build, and then click **Unlock**.

The system does 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.

Click the **Locked** tab to view jobs that are in the locked state. Such projects do not appear in the **Completed** page.

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** panel.

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 “Classes” on page 288.

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 365.

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, and then click the **Purge** button).

For example, if you generated numerous Production jobs 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, and 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 runs 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.

Note: When it runs 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 Project name: My Project Tag: Job 5	C:\BuildForge\My_Project\Job_5\ The system creates only the bold portions of the path. You must create the server directory before running the project, or it fails.

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 runs a step, it can start from the directory it constructed for the job, or if the step's **Path** option is set to Absolute, it can ignore the project and tag directories.

- When the step's **Path** setting is Absolute, the system constructs the path for the step by adding together the **server path** and the step's **Directory** field. The value in the **Directory** 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 Directory field: /bin Step's Path setting: Absolute	C:\BuildForge\bin Use this form to access directories located in the server directory.
Server's Path field: C:/BuildForge Step's Directory field: / (the default value) Step's Path setting: Absolute	C:\BuildForge
Server's Path field: C:/BuildForge Step's Directory field: C:/temp Step's Path setting: Absolute	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® computers.

Important: Setting the path for a command to the root directory can result in the unintended deletion or modification of system files. Run your commands in a different directory if possible. However, if you are automating system administration tasks and require the path to be the root directory, set the server's **Path** to the root directory, the step's **Directory** to ../, and the step's **Path** to Absolute.

- When the step's **Path** option is set to Relative, the system constructs the path for the step by adding together the *server path*, the *project name*, the *tag*, and the step's *Directory field*. The value in the **Directory** 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 Directory field: /bin Step's Path setting: Relative	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 **Directory** field do not exist, the step fails; the system does not create these directories. The portion of the path specified by the Step's **Directory** 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 the **Path**, either Relative or Absolute, and uses that as the default on new steps.

Slashes in paths

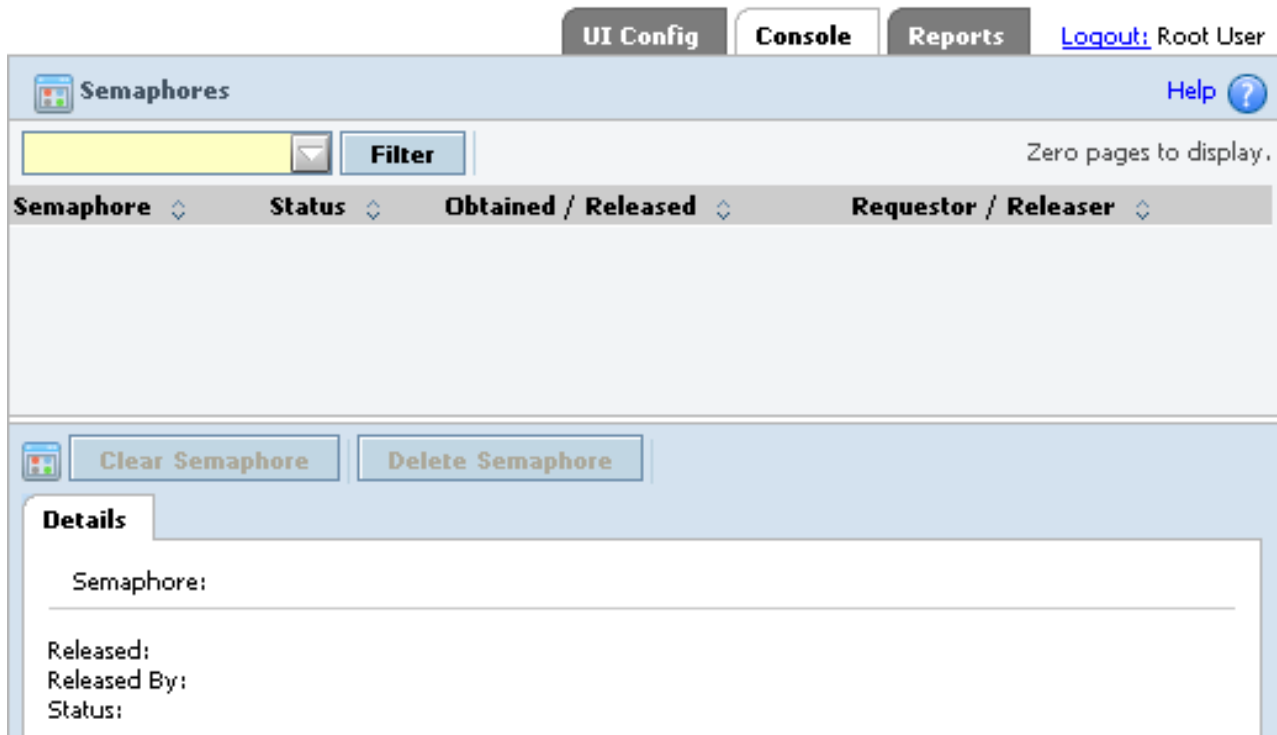
When you enter a path in a **Path** or **Dir** field, the system converts any back slashes to forward slashes. When the system creates the actual path, if the server being used is a Windows® computer, the system converts any forward slashes to back slashes. Therefore, you can use either type of slash 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 canceled 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	<code>.semget \$BF_PROJECTNAME_PHYS</code>
Run driver creator	<code>prntdrivermaker.exe windows</code>
Release semaphore	<code>.semput \$BF_PROJECTNAME_PHYS</code>

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.
- 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**.

Chapter 21. 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 panels and the reports that you create using the Quick Report tool.

To get started, click the **Reports** tab in the upper-right corner to display choices for reporting:

- Select the **Performance** panel to view standard reports.
- Select the **Queries** panel to view and run predefined query reports.
- Select the **Quick Report** panel if you want to create your own reports. Quick Report is a licensed option of Rational Build Forge.

Performance and Queries panels

The Performance panel 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 panel also has a Queries panel that includes predefined query reports. To run these reports, provide the required data, and 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 panels 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, and 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. Also, 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.

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 panel, the permissions used are those of the user who opens the report page.
- For the predefined reports in the Queries panel, 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” on page 380.

Exporting results for built-in reports to a CSV file

Several built-in reports can 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 panel (**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.
 - Save the file to your local workstation or another location on your network.

Standard reports for Performance

Standard reports display results based on current data when you open a standard report page.

The Performance panel 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 statistics for projects

The Performance panel 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** panel.

The job performance report for all projects is displayed as a table of results.

Viewing job duration times for a project

The Performance panel includes a job duration report that graphs the durations of all the project's jobs.

Data source: Rational Build Forge database

Report format: Graph

To view the project jobs duration report:

1. Open the **Reports** tab.
2. Click the **Performance** panel.
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 panel 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 examples:

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.
- **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** panel.
3. Choose a project in the list, and then click **Analyze**.

The analysis reports for steps and servers and the critical path analysis report are displayed as tables of results.

Predefined reports for Queries

Provide a date range or other required data to run a predefined report using current project and job data.

The Queries panel 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 panel 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** panel.
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 panel 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** panel.
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 panel 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:

1. Open the **Reports** tab.
2. Click the **Queries** panel.
3. Use the default time range or specify a time range for the report results. Use the UNIX time format to specify the times. UNIX time format is the number of seconds from January 1, 1970. You can use tools such as the one at http://www.onlineconversion.com/unix_time.htm to calculate UNIX time from a conventional date-time format.

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, 2009, enter the following UNIX times:

From: 07/21/09 00:00 AM	1248134400
To: 07/22/09 00:00 AM	1248220800

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 panel 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** panel.
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.
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 panel 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 346. The .scan command completes the following tasks:


- creates a list of files in a step'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** panel.
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.

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 342.

To get started, click the **Reports** tab to display choices for reporting, and 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:

- 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 run time in seconds for the step and any steps it inlined.
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.
- 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 Level	The user-assigned access group for the project.
Project Name	The user-assigned project name.
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 Level	The user-assigned access group 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 run time in seconds for the step and any steps it inlined.
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 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.

Field Name	Description
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.

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 run time in seconds for the step and any steps it inlined.
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 to 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 checkpoint command in a project, for the files scanned, the BOM Data field displays MD5 values.
BOM Path	If you include the .scan checkpoint command in a project, for the files scanned, the BOM Path field displays the file path.
BOM Type	If you include the .scan checkpoint 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.
Result Description	The description field in the result record, which typically contains a copy of the step 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 run time in seconds for the step and any steps it inlined.
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” on page 389.


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.

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:

- 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" on page 389.

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 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 data value units are displayed on the chart's 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 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 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" on page 389.

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.
- 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.

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 Report fields: Build Tag, Project Name, Build Count, Build State Report options: Group by Project Name

Sample Report Name	Description
SampleCapacity- RuntimeByProject	<p>Description: Creates a bar chart that shows the average build duration for jobs by project</p> <p>Report type: Capacity</p> <p>Report format: Bar chart</p> <p>Report fields: Build Average Duration (y-axis) and Project Name (x-axis)</p> <p>Report options: Group by Project Name</p>
SampleProject-TotalsByProject	<p>Description: Creates a table that reports on the number of steps passed and failed by project</p> <p>Report type: Project</p> <p>Report format: Table</p> <p>Report fields: Project Name, Passed Step Count, Failed Step Count, Step Count</p> <p>Report options: Group by Project Name</p>
SampleQuality- PercentSuccess	<p>Description: Creates a table that reports on the percentage of jobs passed and failed by project</p> <p>Report type: Quality</p> <p>Report format: Table</p> <p>Report fields: Project Name, Build Count, Percentage Builds Failed, Percentage Builds Passed</p> <p>Report options: Group by Project Name</p>
SampleResource- DurationOverTime	<p>Description: Creates a chart that shows build start times and build durations for each build</p> <p>Report type: Resource</p> <p>Report format: Table</p> <p>Report fields: Build Duration (y-axis) and Build Start Time (x-axis)</p> <p>Report options: Ascending sort sequence</p>

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.

Procedure

1. Select the **Reports** tab, and 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” on page 380.
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” on page 385.
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” on page 380.
 - 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.

Adding report output to the job BOM

Optionally use the .report command to add report results to the job BOM.

Procedure

1. Use the Quick Report tool to create a report. See “Creating a report using a provided report type” on page 391.

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 342.

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.

About this task

Report results are displayed in Quick Report results view.

Results

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.

About this task

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.

Before you begin

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*.

Procedure

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 you begin

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.

Procedure

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.
 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.

Chapter 22. 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, which is *<bfinstall>* on Windows and *<bfinstall>/Platform* on UNIX and Linux.

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, and LD_LIBRARY_PATH, must be manually set, or the utilities will not run.

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 run 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:

- 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, which is in <bfinstall> on Windows and in <bfinstall>/Platform on UNIX and Linux.

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
```

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.

Option	Description
<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. Requires -s.
-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.
-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. Requires -s.

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 335.

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.


Note: The Build Forge user interface may refuse to import a file that is larger than 2M. In this case, either use the `bfimport` command-line tool directly or adjust the "upload_max_filesize" PHP environment variable.

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**.

[UI Config](#)[Console](#)[Reports](#)[Logout: Root User](#)

 **Import**[Help ?](#)

[Reset](#) [Import](#)

Project XML File: [Browse...](#)

Select entities to process:

☒ Import Projects

☐ Include Chained Projects

☒ Import Server Definitions☒ Import Environment Variables☒ Import Project Class☐ Import Users☐ Import Project Specific Templates☐ Import Filters☐ Import Collectors☐ Import Selectors☐ Import LDAP Domain Settings

When the names are the same:

☐ Replace Entities☒ Rename Entities

[Reset](#) [Import](#)

- 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.
- 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.

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” on page 403.

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” on page 403.

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.

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, which is in `<bfinstall>` on Windows and in `<bfinstall>/Platform` on UNIX and Linux.

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"
```

Option descriptions

Option	Description
<code><file_name.xml></code>	<p>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 <code>bfexport</code> command or the <code>.export dot</code> 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 issue the <code>bfexport</code> command.</p> <p>If the file name contains spaces, you must quote the name.</p>

Option	Description
-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 <code><project_name>_IMPORT_<number></code> .
-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 <code><project_or_library_name>_IMPORT_<number></code> in the UI.
-S	Imports the selector objects that are defined in the UI. On rename, a selector is imported to the UI as <code><selector_name>_IMPORT_<number></code> .
-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 <code><server_name>_IMPORT_<number></code> . 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 <code><environment_name>_IMPORT_<number></code> .
-c	Imports classes that are referenced by projects. On rename, a class is imported as <code><class_name>_IMPORT_<number></code> .
-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 <code><collector_name>_IMPORT_<number></code> .
-u	Imports users who are members of the access groups designated to receive email notifications, if the -g option is specified for bfexport. On rename, information for users is imported as <code><users>_IMPORT_<number></code>
-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 <code><template_name>_IMPORT_<number></code> .

Option	Description
-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 <code><filter_name>_IMPORT_<number></code> .
-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 <code><LDAP_domain_controller>_IMPORT_<number></code> .
-r	Replaces the imported objects instead of renaming them. By default, imported objects are renamed and the following naming convention is used: <code><object_name>_IMPORT_<number></code> 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: <code><object_name>_IMPORT_<number></code> For details, see “Renaming and replacing objects on import.”

How access groups are assigned to imported objects

The **Import with Secure Access** system setting controls how access groups are assigned to imported objects.

- If **Import with Secure Access** is set to Yes (the default), the **Import Default Secure Access Group** setting specifies the access group. Build Engineer is the default.
- If **Import with Secure Access** is set to No, the **Import Insecure Default Access Group** setting specifies the access group. Developer is the default.

If an access group for **Import Insecure Default Access Group** is not specified, the most recently created access group is used. If no access groups have been created, the Default access group is used.

Access group assignment and security

As a security measure, Build Forge does not allow you to export or import access groups directly. If it did, a user with access to the console host could directly manipulate the XML files of exported objects, and then import them. Assigning access groups upon import assures that the access groups are assigned only by authorized users.

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.

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.

Chapter 23. Linking to Web resources in the UI Config tab

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 external resources such as 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	<ul style="list-style-type: none">• 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, the tab is hidden.
7. At **Visible**, select one option:
 - True (the default): Select to show the tab in the UI.
 - 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.

Chapter 24. Build Catalyst

Build Catalyst accelerates make-based C and C++ software builds. It interprets and analyzes your existing make files and runs an accelerated build. The accelerated build can use techniques such as parallel builds and distributed builds to reduce overall build time.

Build Catalyst is provided with IBM® Rational® Build Forge® but requires additional installation. To use Build Catalyst with Rational Build Forge, install Build Catalyst on a host where a Rational Build Forge agent is installed. When installed and configured, the Build Catalyst accelerated build can be called by a Rational Build Forge step. The integration with Rational Build Forge projects allows access to the broader build automation tools that Rational Build Forge provides. For example, calling the refactored make-based builds may be part of a larger build scenario involving other builds. The Rational Build Forge project can also run pre-build and post-build procedures.

Build Catalyst is highly compatible with makefiles that were written for GNU Make 3.80. If you have a makefile written for GNU Make 3.80, you can use Build Catalyst to build your source code without changing the makefiles in most cases. Some cases might require minor changes. Build Catalyst supports these builds:

- **Parallel builds:** Build Catalyst can identify nondependent targets and build them in parallel. Building in parallel makes better use of resources, such as multicore processors, in the build computer. By using resources efficiently, Build Catalyst accelerates builds and reduces build times.
- **Distributed builds:** Distributed builds are like parallel builds; however, the build activity is distributed among multiple computers.

The Build Catalyst documentation is for new and experienced users of Build Catalyst, who are familiar with software build concepts.

Supported operating systems

You can use Build Catalyst on several Linux®, Solaris, and Microsoft® Windows® operating systems.

These operating systems support Build Catalyst:

- Red Hat Enterprise Linux 4, 5
- Solaris 9, 10 (SPARC)
- Microsoft Windows XP Professional
- Microsoft Windows Server 2003

Note: Only the Linux and Solaris operating systems support distributed builds.

Installation overview

You install Build Catalyst from an archive file on the Linux®, Solaris, or Microsoft® Windows® operating system.

For more information, see “Supported operating systems.”

Build Catalyst has an installer that is separate from the IBM® Rational® Build Forge® installer. To use Build Catalyst with Rational Build Forge, install Build Catalyst on the same computers on which you install the Rational Build Forge agents. You can install Build Catalyst on any computer that you plan to use as a build host, provided that the computer runs one of the supported operating systems. The computer need not have the Rational Build Forge agent installed if you do not plan to have Rational Build Forge jobs that use Build Catalyst on the given computer.

IBM distributes Build Catalyst as an archive file that contains the following items:

- A “dump file” that includes all the installable files (binary files, various built-in makefiles, and so on)
- The installation script

Installing Build Catalyst on Linux and Solaris operating systems

You install Build Catalyst on Linux® and Solaris operating systems by using the Build Catalyst archive file.

About this task

To use Build Catalyst for a distributed build, install it on all the computers that are used in the distributed build.

Procedure

1. Download the Build Catalyst archive file from <http://hostname/buildcatalyst>, where *hostname* is your Management Console host.
2. Extract the archive file into any directory.
3. Become superuser, by entering `su`, if you are not the superuser.
4. Go to the directory where you extracted the installer, and then go to the new subdirectory that contains the `install.sh` file.
5. Run `./install.sh` in that directory.

- a. Specify an installation directory. (The default directory is `/opt/rational/buildforge/buildcatalyst`.)

If an installation already exists (even if it is in a location other than the one you provided), the installer warns that the existing installation will be overwritten or rendered unusable if you proceed with the installation. (An existing installation not in the default directory can be rendered unusable because the current installation process creates symbolic links from the default installation directory to the newly specified installation directory.) The installer then prompts for the installation directory again. After you provide an installation location, the installation writes the files to the provided location.

- b. Specify which `rsh` or `ssh` binary file to use for distributed builds (The default program is `rsh`, if found in the path.)

The Build Catalyst binary file is installed as `rafmake` in the `bin` directory in the installation directory. The installation creates the following items:

- A symbolic link from `/opt/rational/buildforge/buildcatalyst` to the installation directory.
- A symbolic link from `etc/rafbe_starter` in the installation directory to the shell binary file. The shell binary file is either `rsh` or `ssh`.

- A log file for the Rational Automation Framework build engine, or rafbe, at `/var/adm/rational/buildforge/logs`

Results

You can now start rafmake from your IBM® Rational® Build Forge® jobs to build your C or C++ software. You can also start rafmake independently of your Rational Build Forge jobs.

Installing Build Catalyst on Windows

You install Build Catalyst on Microsoft® Windows® by using the Build Catalyst archive file.

Before you begin

Verify that Microsoft Visual C++ 2005 SP1 Redistributable Package version 8.0.50727.762 is installed on your computer. You can verify which package is installed by using the **Add or Remove Programs** item in the Control Panel.

Important: Build Catalyst does not use any other version of this package. You must have version 8.0.50727.762 installed.

If required, you can download the correct version from www.microsoft.com/downloads/.

For more information about Windows support, see “Supported operating systems” on page 409.

Procedure

1. Download the Build Catalyst archive file from <http://hostname/buildcatalyst>, where *hostname* is your Management Console host.
2. Extract the archive file to any folder.
3. Open that folder and then open the created folder that contains the `install.bat` file.
4. Run `install.bat`. The Build Catalyst binary file is installed as the `rafmake.exe` file. The bin folder that the file is saved to is indicated in the following table.

Table 12. Installation folders

Operating system type	Location
32-bit operating systems	C:\Program Files\IBM\Build Forge\buildcatalyst
64-bit operating systems	C:\Program Files (x86)\IBM\Build Forge\buildcatalyst

Results

You can now start rafmake from your IBM® Rational® Build Forge® jobs to build your C or C++ software. You can also start rafmake independently of your Rational Build Forge jobs.

Build Catalyst examples

Use these examples to better understand how you can use Build Catalyst for simple builds, parallel builds, and distributed builds.

Simple builds

For simple builds (builds that are not parallel or distributed), run the **rafmake** command as follows:

```
rafmake [ -f makefile ] [ options ] ... [ targets ] ...
```

where

- *makefile* is the makefile to use
- *options* include the options explained in “rafmake utility reference” on page 414
- *targets* are the target files to build

The following list provides examples by operating system.

- Linux and Solaris
 - Unconditionally build the default target in a particular makefile with all its dependencies:

```
% rafmake -u -f project.mk
```
 - Build a specific target in a particular makefile with verbose output:

```
% rafmake -v -f project.mk a.out
```
- Windows
 - Build the default target in a default makefile with the particular value that the INCL_DIR macro sets:

```
Y:\> rafmake INCL_DIR=C:\src\include
```

Parallel builds

The **rafmake** command supports parallel builds, which separate builds into independent components that can be built at the same time to reduce the overall build time.

The syntax to run parallel builds is as follows:

```
rafmake -J num [ -f makefile ] [ options ] ... [ targets ] ...
```

where

- *num* is the number of concurrent build jobs that build independent targets at the same time
- *makefile* is the makefile to use
- *options* include the options explained in “rafmake utility reference” on page 414
- *targets* are the target files to build

Running parallel builds provides efficient resource utilization, especially with multicore processors. The performance gain from parallel builds depends on the type of the build. For example, in the case of a CPU-intensive build, the optimum number of concurrent build jobs would be the same as the number of processors (or cores). If you specify more than that, you might not realize a significant performance gain. For an IO-intensive build, however, increasing the number of parallel jobs reduces build time. Try a few builds with different numbers to determine the optimal number of concurrent build jobs for your resources.

The following list provides examples of parallel builds by operating system.

- Linux and Solaris
 - Build a default target in a particular makefile, in parallel, with three concurrent build jobs:

```
% rafmake -J 3 -f project.mk
```
- Windows
 - Build a specific target in a default makefile with the value that the INCL_DIR macro sets, in parallel, with five concurrent jobs:

```
Y:> rafmake -J 5 INCL_DIR=C:\src\include software.exe
```

Distributed builds

Parallel builds can use resources only on one computer. Distributing a build takes parallel builds a step further by distributing the build load across multiple computers.

Note: To use Build Catalyst for a distributed build, install it on all the computers that are used in the distributed build.

To distribute a build, your environment must meet the following requirements:

- The source code is shared between the different computers.
For all the computers on which the build is to be distributed, the same source code must be accessible by the same path. Typically, having the source code on NFS-mounted shares (and mounting them on the same path on all computers) ensures that the same source directory is available on all build computers in the same path.
- The login through the chosen remote shell (rsh or ssh) is configured not to require passwords.
When a user starts a distributed build from a host computer, the **rafmake** command internally starts the build engine (rafbe) on all the computers where the build is to be distributed. The build engine is started using either the **ssh** or **rsh** command, which is configured when installing Build Catalyst. The `/opt/rational/buildforge/buildcatalyst/etc/rafbe_starter` file is a symbolic link to the remote shell that the user chose during installation. The user who starts the build must be able to log in without a password to all the computers that are using the chosen remote shell. By logging in without a password, the build engine can be started silently on the various computers. Consult the **rsh** or **ssh** command manual to set up a login that requires no password.
- A build hosts file describes how to distribute the builds on the computers.
A build hosts file typically contains several lines with the host names or IP addresses of many computers. These computers are where the builds are to be distributed. The file also contains lines that describe other options, such as `-idle num`. This type of line specifies how idle a computer must be for builds to be scheduled on the computers that are listed below it. Idleness is assumed to be 50 if it is not explicitly mentioned. Here is an example build hosts file:

```
### Contents of build hosts file
host1.mydomain.com
-idle 25
host2.mydomain.com
host2.mydomain.com
192.168.0.200
-idle 75
host3.mydomain.com
### End of build hosts file
```

In this example, the build is distributed to four computers: host1, host2, host3, and 192.168.0.200. The idleness for the host1.mydomain.com computer is assumed to be 50, because it is not explicitly mentioned. For the host2 and 192.168.0.200 computers, the idleness is specified as 25. This means that if one of those computers is less than 75% loaded or busy (or at least 25% idle), builds are scheduled on that computer. If the computers are less than 25% idle, builds are not scheduled on those computers. Also, the example lists the host2 computer twice. As a result, two parallel build threads can be started on the host2 computer. Depending on computer configurations, you can tweak your build hosts file to use resources optimally for builds.

After you create the build hosts file, you can start the distributed build by using the following command syntax:

```
rafmake -J num -B build_hosts_file [ -f makefile ]
      [ options ] ... [ targets ] ...
```

The *num* value for the -J option specifies the total number of build threads for all computers.

Without the -B option, the **rafmake** command would perform a parallel build instead of a distributed build. When you specify the -B option and a valid build hosts file, the build engine schedules the distributed builds on the different computers that are specified in the build hosts file.

rafmake utility reference

The interface for Build Catalyst is the make utility called rafmake. The interface has various options that control its behavior.

The syntax for the command depends on the operating system, as shown in the following table.

Table 13. Options by operating system

Operating system	Options
Linux and Solaris (Build a target.)	rafmake [-f <i>makefile</i>] ... [-ukinservwdpqUN] [-J <i>num</i>] [-B <i>bldhost-file</i>] [-c <i>compat-mode</i>] [-C <i>change-directory</i>] [-A <i>BOS-file</i>] ... [<i>macro=value</i> ...] [<i>target-name</i> ...]
Windows (Build a target.)	rafmake [-f <i>makefile</i>] ... [-ukinservwdpqUN] [-J <i>num</i>] [-c <i>compat-mode</i>] [-C <i>change-directory</i>] [-A <i>BOS-file</i>] ... [<i>macro=value</i> ...] [<i>target-name</i> ...]
All (Display version information for the rafmake utility.)	rafmake { -ver/sion -VerAll }

Option descriptions

The **rafmake** command supports most of the common options of the GNU version 3.80 make command. The **rafmake** command also provides additional options.

Tip: You can combine options that do not take arguments, for example, -rNi.

Table 14. Option descriptions

Option	Description
<code>-f <i>makefile</i></code>	Uses the <i>makefile</i> as the input file. If you omit this option, the rafmake command looks for input files named <i>makefile</i> and <i>Makefile</i> (in that order) in the current working directory. You can specify more than one <code>-f <i>makefile</i></code> argument pair. Multiple input files are effectively concatenated.
<code>-u</code>	(Unconditional) Rebuilds all specified targets and all of their dependencies, regardless of whether they need to be rebuilt. (See also <code>-U</code> .)
<code>-k</code>	Abandons work on the current entry if it fails, but continues on other targets that do not depend on that entry.
<code>-i</code>	Ignores error codes that commands return.
<code>-n</code>	(No-execute) Lists command lines, including those that begin with an at sign (@), from the <i>makefile</i> for targets that need to be rebuilt, but does not run them. Exception: A command that contains the string \$(MAKE) is always run.
<code>-s</code>	(Silent) Does not list command lines before running them.
<code>-e</code>	Environment variables override macro assignments in the <i>makefile</i> file. (However, <i>macro=value</i> assignments on the command line or in a build options specification override environment variables.)
<code>-r</code>	(No rules) Does not use the built-in rules in the <code>buildcatalyst-home-dir/etc/builtin.mk</code> file (Linux and Solaris) or the <code>buildcatalyst-home-dir\etc\builtin.mk</code> file (Windows). When used with the <code>-c</code> option, the <code>-r</code> option also disables reading platform-specific startup files. See the <code>-c</code> option for more information.
<code>-v</code>	(Verbose) Slightly more verbose than the default output mode.
<code>-w</code>	(Working directory) Prints a message that contains the working directory both before and after running the <i>makefile</i> .
<code>-d</code>	(Debug) Quite verbose and includes a list of the environment variables that the rafmake command reads during the build. Use this option only when debugging <i>makefiles</i> .
<code>-p</code>	(Print) Lists all target descriptions and all macro definitions, including target-specific macro definitions and implicit rules. Does not run anything.
<code>-q</code>	(Query) Evaluates <i>makefile</i> targets but does not run the build scripts. The rafmake command returns one of these responses: <ul style="list-style-type: none"> • 0 if the targets are up to date • 1 if any targets need to be rebuilt
<code>-U</code>	Unconditionally builds goal targets only. Does not build subtargets. If you do not specify any target on the command line, the default target is the goal. (The <code>-u</code> option unconditionally builds both goal targets and build dependencies.)
<code>-N</code>	Disables the default procedure for reading one or more build option specification (BOS) files.
<code>-J <i>num</i></code>	Enables parallel building capability. The maximum number of concurrent target rebuilds is set to the integer <i>num</i> . If <i>num</i> =0, parallel building is disabled. (This is equivalent to not specifying a <code>-J</code> option.) Alternatively, you can specify <i>num</i> as the value of the RAFMAKE_CONC environment variable.

Table 14. Option descriptions (continued)

Option	Description
<code>-B bldhost-file</code>	<p>Uses the <i>bldhost-file</i> file as the build hosts file for a parallel build. If you do not specify the <code>-B</code> option, the rafmake command uses the <code>.bldhost.\$RAFMAKE_HOST_TYPE</code> file in your home directory. When you use the <code>-B</code> option, you must also use the <code>-J</code> option or have the <code>RAFMAKE_CONC</code> environment variable set.</p>
<code>-c compat-mode</code>	<p>(Compatibility) Invokes the rafmake command in a compatibility mode.</p> <p>(Alternatively, you can use the <code>RAFMAKE_COMPAT</code> environment variable to specify a compatibility mode.)</p> <p>You can set the <i>compat-mode</i> variable to one of the following values:</p> <p>gnu</p> <p>Emulates the Free Software Foundation's Gnu make program. To define built-in make rules, the rafmake command reads <code>gnubuiltin.mk</code> instead of <code>builtin.mk</code>.</p> <p>This mode is the default compatibility mode.</p> <p>std</p> <p>Runs the rafmake command with the IBM Rational® ClearCase® clearmake command compatibility mode enabled.</p> <p>On Linux and Solaris systems only, the <i>compat-mode</i> variable can also have one of the following values. The <code>-c</code> option is Linux and Solaris platform-independent. However, some modes try to read system-specific files; if those files do not exist, the command fails.</p> <p>sun</p> <p>Emulates the standard make(1) command that SunOS systems provide.</p> <p>aix</p> <p>Emulates the standard make(1) command that IBM AIX® systems provide.</p>
<code>-C dir</code>	<p>Changes the directory. The rafmake command changes to the <i>dir</i> directory before starting the build.</p>
<code>-A BOS-file</code>	<p>Specifies BOS files to be read immediately after the ones that are read by default. Use this option multiple times to specify multiple BOS files.</p> <p>When you use <code>-N</code> with this option, rafmake reads the specified BOS files instead of the default BOS files.</p> <p>Alternatively, you can specify a colon-separated list of BOS file path names (Linux and Solaris) or a semicolon-separated list of such path names as the value of the environment variable <code>RAFMAKE_OPTS_SPECS</code>.</p>
<code>-ver/version</code>	<p>Prints version information about the rafmake command.</p>
<code>-VerAll</code>	<p>Prints version information about the rafmake command and the libraries (Linux and Solaris) or the DLLs (Windows) that the rafmake command uses.</p>

Build Catalyst environment variables

Build Catalyst supports several environment variables to simplify usage.

The variables are grouped by operating system in the following sections:

- “Environment variables that are common to Linux, Solaris, and Microsoft Windows”
- “Environment variables for Linux and Solaris only” on page 418
- “Environment variables for Windows only” on page 419

Environment variables that are common to Linux, Solaris, and Microsoft Windows

RAFBE_INIT_TIMEOUT

Specifies the maximum number of minutes that the **rafmake** command waits for a new Rational Automation Framework build engine (rafbe) to request an initial build context.

Default: 3 minutes

RAFBE_PN

Sets the full path name that the **rafmake** command uses to call the build engine (rafbe) on a local or remote host during a parallel build.

Default:

- Linux and Solaris: *bfinstall/etc/rafbe*
- Windows: *bfinstall/bin/rafbe*

RAFBE_START_TIMEOUT

Sets the maximum time allotted for starting the build engine.

Default: 30 seconds

RAFBE_TIMEOUT

Specifies the maximum number of minutes for the build engine to wait for the **rafmake** command to reply after the build engine requests a target build.

Default: 180 minutes

RAFMAKE_CONC

Sets the concurrency level in a **rafmake** build. This variable takes the same values as the **-J** option. Specifying the **-J** option when you use the **rafmake** command overrides the setting of this variable.

RAFMAKE_COMPAT

Specifies a **rafmake** command compatibility mode. This variable takes the same values as the **-c** option. Specifying the **-c** option when you use the **rafmake** command overrides the setting of this variable.

Default: None

RAFMAKE_MAKEFLAGS

Provides an alternative or supplementary mechanism for specifying **rafmake** command options. The **RAFMAKE_MAKEFLAGS** environment variable can contain the same string of key letters that is used for command-line options, except that options that take arguments are not valid. Options you specify on the **rafmake** command line override the setting of this environment variable if there is a conflict.

Default: None

RAFMAKE_OPTS_SPECS

Provides a list of path names, separated by colons (Linux and Solaris) or semicolons (Windows), each of which specifies a build option specification (BOS) file that the **rafmake** command reads. You can use this variable instead of specifying BOS files on the **rafmake** command line with one or more **-A** options.

Default: Undefined

RAFMAKE_SHELL_FLAGS

Specifies **rafmake** command options to pass to the subshell program that runs a build script command.

Default:

- Linux and Solaris: **-e**
- Windows: None

RAFMAKE_SHELL_REQUIRED

Forces the **rafmake** command to run build scripts in the shell program that you specify with the **SHELL** macro. To make the **rafmake** command run build scripts in the shell program, set this variable to **TRUE**. To configure the **rafmake** command to run build scripts directly, unset the variable.

Default: The **rafmake** command runs build scripts directly

RAFMAKE_VERBOSITY

Specifies the **rafmake** command message logging level as follows:

1 Equivalent to **-v** (verbose) on the command line

2 Equivalent to **-d** (debug) on the command line

0 or undefined

Equivalent to standard message logging level

If you also specify **-v** or **-d** on the command line, the higher value prevails.

Default: 0

MAKEFLAGS

Lists one or more flags to pass to the **make** command. The **rafmake** command reads the contents of the **MAKEFLAGS** environment variable at startup. Then the command amends the variable to include flags that are not specific to Build Catalyst that are passed on the command line. Flags that are specific to Build Catalyst are passed through the **RAFMAKE_MAKEFLAGS** environment variable, and if the **rafmake** command detects these flags in the **MAKEFLAGS** variable, the command moves them to the **RAFMAKE_MAKEFLAGS** variable.

Flags passed through the **MAKEFLAGS** variable are as follows: **-I**, **-p**, **-N**, **-w**, **-e**, **-r**, **-i**, **-k**, **-n**, **-q**, **-s**

Flags passed through the **RAFMAKE_MAKEFLAGS** variable are as follows: **-A**, **-B**, **-N**, **-v**, **-c**, **-U**, **-M**, **-u**, **-d**

Default: None

Environment variables for Linux and Solaris only

RAFBE_STARTER_PN

Specifies the path name of the **rafbe_starter** link, which is by default a path to the **rsh** command. However, in environments where starting a remote

shell is not allowed or is undesirable, you can, for example set this environment variable to the path for the **ssh** command.

Default: /usr/bin/rsh

RAFMAKE_BLD_HOSTS

Specifies one or more build hosts on which the **rafmake** command can build targets.

Default: Undefined

RAFMAKE_HOST_TYPE

Determines the name of the build hosts file to use during a parallel build (**-J** option): `.blldhost.$RAFMAKE_HOST_TYPE` in your home directory. (Your home directory is determined by examining the password database.)

Specifying a **-B** option on the command line overrides the setting of this variable.

C Shell Users: Set this variable in the `.cshrc` file, not in the `.login` file. The parallel build facility calls a remote shell, which does not read the `.login` file.

You can also code `RAFMAKE_HOST_TYPE` as a **make** macro.

Default: None

Environment variables for Windows only

RAFMAKE_NO_ESC_PATT_CHARS

Overrides the escape character (`\`) in the **rafmake** command's GNU-compatible mode. For example, both the **rafmake** and GNU **make** commands assume that the `\%` string indicates the literal character, `%`. The commands do not treat the rule as a pattern rule. To prevent the **rafmake** command from using the escape character to indicate a literal character, set this environment variable to any non-null value.

RAFMAKE_PNAME_SEP

Sets the path-name separator for path names that the **rafmake** command constructs. This variable can be set in the makefile, in a BOS file, on the command line, or as an environment variable.

Default: If this variable is not set or is set to any value other than a forward slash (`/`) or a backslash (`\`), the **rafmake** command uses a backslash (`\`) as the path-name separator.

Chapter 25. Direct integrations with Build Forge

Build Forge integrates easily with applications that has a command-line interface.

To set up this kind of integration, you do the following:

1. Install the Build Forge console on a host
2. Install the Build Forge agent on the application host (or a host that can access the application)
3. In the console, create a Server resource and Server authentication. Configure the Server resource to access the Build Forge agent you installed.
4. Configure the agent and host environments as necessary for commands to run on the application. This may be as simple as creating a user account for Build Forge to use, then assuring that the running agent has its PATH set up correctly. It may also require that you install and configure a client that is used to run commands in the application. For example, Rational ClearCase and ClearQuest require the use of client applications to run commands.

Chapter 26. Adaptor integrations

Adaptors provide special capabilities for interacting with external applications.

Sample adaptor templates 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.

An adaptor's behavior is defined by an XML file whose elements are specified by an included DTD. Build Forge executes the adaptor in association with a project step.

Adaptors give you additional tools for integration-based builds:

- Internal condition logic: based on settings of internal variables
- Command definition: application commands can be built up out of command statements and variables.
- Response scanning: you can define patterns to scan for in response to each command.
- Dynamic notification groups: notification can be based on data gathered from the application. For example, the group of team members to notify can be built up out of only the members who checked in code changes.
- Script execution: adaptors can run scripts on the Build Forge host. This can be independent of scripts and commands run on the application host.

Integrating with Rational ClearCase

BuildForge can work with Rational ClearCase for code source management.

This topic describes additional setup requirements for integrating with Rational ClearCase and summarizes the characteristics of the provided sample adopter templates.

The ClearCase adaptor template samples provide methods of analyzing changes to a baseline. Typically change analysis is used for build avoidance: if a baseline component has not changed, it is not rebuilt

Additional setup requirements for ClearCase adaptors

See for general requirements In addition to fulfilling the general requirements for adaptors, do the following:

1. Install the ClearCase full client on the agent host.
2. Set up the environment for the agent so that commands can be run through the ClearCase client.

ClearCase adaptor template samples

The following adaptor template samples are provided.

ClearCaseBaseline

1. Scans a directory in a ClearCase view.

2. Writes branch and version information reported by ClearCase to the BOM report.

Variables used:

- INT_STREAM
- VIEW
- PROJECT_VOB
- CCSERVER
- UNIXCLIENT

ClearCaseByBaselineActivities

1. Creates a new baseline from the contents of a ClearCase view.
2. Compares the new baseline and the baseline from the previous adaptor execution to identify change activity.
3. For each change activity, writes the following information to the BOM report: activity, files changed, user, date, comments, and version.
4. For each changed file, writes change details (from diff command output) to the BOM report.

Variables used:

- CurDate
- LAST_RUN
- BASELINE
- VIEW
- VOB_PATH
- PROJECT_VOB
- CCSERVER
- UNIXCLIENT

ClearCaseByBaselineVersions

1. Creates a new baseline from the contents of a ClearCase view.
2. Compares the new baseline and the baseline from the previous adaptor execution to identify changed files.
3. For each changed file, writes the following information to the BOM report: file name, version, date, user, and comments.
4. For each changed file, writes change details (from diff command output) to the BOM report.

Variables used:

- CurDate
- LAST_RUN
- LABEL
- BASELINE
- VIEW
- VOB_PATH
- PROJECT_VOB
- CCSERVER
- UNIXCLIENT
- _CHAR_NATIVE

ClearCaseByDate

1. Queries a ClearCase view for changes between two dates. The default dates are the current timestamp and the timestamp of the previous adaptor execution.
2. For each changed file, writes the following information to the BOM report: file name, version, date, user, and comments.
3. For each changed file, writes change details (from diff command output) to the BOM report.

Variables used:

- CurDate
- LAST_RUN
- LABEL
- BASELINE
- VIEW
- VOB_PATH
- PROJECT_VOB
- CCSERVER
- UNIXCLIENT
- _CHAR_NATIVE

ClearCaseByLabel

1. Creates and applies a new label to the contents of a ClearCase view.
2. Compares the new label and the label from the previous adaptor execution to identify changed files.
3. For each changed file, writes the following information to the BOM report: file name, version, date, user, and comments.
4. For each changed file, writes change details (from diff command output) to the BOM report.

Variables used:

- CurDate
- LAST_RUN
- LABEL
- BASELINE
- VIEW
- VOB_PATH
- PROJECT_VOB
- CCSERVER
- UNIXCLIENT
- _CHAR_NATIVE

ClearCase adaptor variables

This table is a reference for the variable lists for the adaptor templates.

Environment variable name	Description
BASELINE	For ByBaseline adaptors, when you use your adaptor to generate differences by baseline, the system uses this value as the baseline.

Environment variable name	Description
CCSERVER	Set this variable to the name of the host that has the ClearCase client and Build Forge agent installed.
CurDate	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.
LABEL	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.
LAST_RUN	For ByDate adaptors, the system uses this value to determine whether any changes have occurred; the value is 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.
PROJECT_VOB	When you use the ByBaseline adaptor, set this variable to the name of your Project VOB (only used with UCM ClearCase). Example: \ProjectVob
UNIXCLIENT	Used to set platform-specific information. Set to 0 if the client is running on Windows. Set to 1 if the client is running on UNIX or Linux.
VIEW	Set this variable to the name of the ClearCase view that you want to use with the adaptor.
VOB_PATH	Set this value to the name of your component VOB, and optionally, its subdirectories. Use a comma-separated list for multiple names.
_CHAR_NATIVE	Used internally and always set to 1.

The following trigger variables may also be used to start views and mount vobs. However, they are independent of the views and vobs set by the adaptor environment.

- CLEARCASE_VIEW
- _CLEARCASE_VIEWS
- _CLEARCASE_VOBS

See “Trigger variables reference” on page 261.

Integrating with Rational ClearQuest

BuildForge can work with Rational ClearQuest to update build records.

Build Forge has two integrations with ClearQuest:

- Automatic build record creation or update based on job status. This capability is automatically activated when the required environment variables are set in the project environment.
- ClearQuest adaptors

These capabilities are completely independent of each other. In particular, the adaptor is associated with an environment created for it. The variables in that environment are independent of variables set to activate automatic build records.

Setting up automatic generation of build records

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.

When you configure a project to update a ClearQuest database, the system performs creates or updates build records as follows:

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 automatic build record update through special environment variables. To link a project to a ClearQuest database, make sure the variables in the following table 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. However, 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.

In order to activate automatic updates of build records from Build Forge jobs, the following environment variables must be set for the project. They do not work at the project level.

Table 15. Environment variables required for automatic updates of build records

Variable	Description
CQ_DBNAME	Required. Name of the ClearQuest database you want to update.
CQ_DBSET	The ClearQuest database set value. Not required. Defaults to blank.

Table 15. Environment variables required for automatic updates of build records (continued)

Variable	Description
CQ_INTERACTION	<p>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.</p> <p>To enable build record creation, set this environment variable to ON.</p> <p>Note: If you are using one of the ClearQuest adaptors, set this environment variable to OFF. The adaptor interacts with the build records directly.</p>
CQ_PASSWORD	Required. Password to use when logging into the ClearQuest database. Not required; defaults to blank.
CQ_RELEASE_NAME	Required. The name of the release within the ClearQuest database that you want to update.
CQ_USER	Required. Username to use when logging into the ClearQuest database.

Additional setup requirements for ClearQuest adaptors

See “Adaptor requirements” on page 439 for general requirements. In addition to fulfilling the general requirements for adaptors, do the following:

1. Install the ClearQuest full client on the Build Forge console host.
2. Install the CQperl utility (ClearQuest Perl API) on the Management Console system and add the CQperl installation directory to the system path statement.
3. Add the CQ_INTERACTION environment variable to your project and set it to OFF.
4. Open the CQ Maintenance Tool and set up a connection to the ClearQuest database. This ensures that the Build Forge Management Console can successfully create the build record and populate it with information about the build.

You do not need to install an agent. The ClearQuest adaptor communicates directly to ClearQuest through the client, using the ClearQuest Perl API.

Important: the ClearQuest adaptor can be called only with a dot command in a step. It is not a source adaptor, so an adaptor link cannot be used.

ClearQuest adaptor template samples

The following adaptor template samples are provided.

ClearQuestBaseClearCaseByDate

1. Queries a ClearCase view for changes between two dates. The default dates are the current timestamp and the timestamp of the previous adaptor execution.
2. 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.

3. For each changed file, writes the following information to the BOM report: the file name, defect ID, defect status, and any ClearQuest errors.

Variables defined in the adaptor template:

- CurDate
- LAST_RUN
- VIEW
- VOB_PATH
- CQ_USER
- CQ_PASSWORD
- BFSEVER
- UNIXCLIENT
- _CHAR_NATIVE

ClearQuestClearCaseByActivity

1. Finds ClearQuest defect records associated with a list of ClearCase activities.
2. 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.
3. Writes the following information to the BOM report: files associated with ClearCase activity IDs and the ClearQuest defect status.

Variables defined in the adaptor template:

- CurDate
- VIEW
- VOB_PATH
- ACTIVITIES
- CQ_USER
- CQ_PASSWORD
- PROJECT_VOB
- BFSEVER
- UNIXCLIENT
- _CHAR_NATIVE

ClearQuestUCMClearCaseByDate

1. Queries a ClearCase view for changes between two dates. The default dates are the current timestamp and the timestamp of the previous adaptor execution. It uses Rational Unified Change Management (UCM) to produce its results.
2. For each changed file, writes the following information to the BOM report: the file name, defect ID, defect status, and any ClearQuest errors.

Variables defined in the adaptor template:

- CurDate
- LAST_RUN
- VIEW
- VOB_PATH
- CQ_USER

- CQ_PASSWORD
- BFSEVER
- UNIXCLIENT
- _CHAR_NATIVE

ClearQuest adaptor variables

This table is a reference for the variable lists for the adaptor templates.

Table 16. Environment variables required for Rational ClearQuest integration

Variable	Description
ACTIVITIES	For the ClearQuestClearCaseByActivity adaptor, a space-delimited set of activity IDs. Example: SAMPL0001@\ProjectVob
BFSEVER	Set this variable to the name of the host for the Build Forge console.
CQ_PASSWORD	Required. Password to use when logging into the ClearQuest database. Not required; defaults to blank.
CQ_USER	Required. Username to use when logging into the ClearQuest database.
CurDate	Supplies the current date to the adaptor, using a .date command to generate the date. Do not change this value.
LAST_RUN	For ByDate adaptors, the system uses this value to determine whether any changes have occurred; the value is 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.
UNIXCLIENT	Used to set platform-specific information. Set to 0 if the client is running on Windows. Set to 1 if the client is running on UNIX or Linux.
VIEW	Set this variable to the name of the ClearCase view that you want to use with the adaptor.
VOB_PATH	Set this value to the name of your component VOB, and optionally, its subdirectories. Use a comma-separated list for multiple names.
_CHAR_NATIVE	Used internally and always set to 1.

Integrating with CVS

BuildForge can work with CVS for source code management.

This topic describes additional setup requirements for integrating with CVS and summarizes the characteristics of the provided sample adaptor templates.

The CVS adaptor template samples provide methods of analyzing changes to a baseline. Typically change analysis is used for build avoidance: if a baseline component has not changed, it is not rebuilt

Additional setup requirements for CVS adaptors

See “Adaptor requirements” on page 439 for general requirements. There are no additional requirements for CVS.

CVS adaptor template samples

The following adaptor template samples are provided.

CVSv1Baseline

1. Scans a CVS directory on a Build Forge agent for changed files.
2. Writes the following information to the BOM report: changed file name, status, working version, repository version, and sticky tag. Scans a directory in a ClearCase view.

Variables used:

- CVSROOT
- MODULE
- CVSCLIENT

CVSv1ByDate

1. Queries a CVS view for changes between two dates. The default dates are the current timestamp and the timestamp of the previous adaptor execution.
2. Writes the following information to the BOM report: change type, date, user name, version, and file name.
3. For each changed file, writes change details (from diff command output) to the BOM report.

Variables used:

- LAST_RUN
- CurDate
- CVSROOT
- MODULE
- BRANCH
- CVSCLIENT

CVSv1ByTag

1. Applies a new tag to a CVS module.
2. Compares the differences between the newly tagged module and a module tagged during the previous adaptor execution.
3. Writes the following information to the BOM report: file name, revision, state, date, time, change author, and commit comments.
4. For each changed file, writes change details (from diff command output) to the BOM report.

Variables used:

- LAST_RUN
- CurDate
- CVSROOT
- MODULE
- BRANCH
- LAST_TAG
- CVSCLIENT

CVSv2ByDate

1. Queries a CVS view for changes between two dates. The default dates are the current timestamp and the timestamp of the previous adaptor execution.
2. Writes the following information to the BOM report: change type, date, user name, version, and file name.
3. For each changed file, writes change details (from diff command output) to the BOM report.

Variables used:

- LAST_RUN
- CurDate
- CVSROOT
- MODULE
- CVSCLIENT

Integrating with Perforce

BuildForge can work with Perforce for source code management.

This topic describes additional setup requirements for integrating with Perforce and summarizes the characteristics of the provided sample adaptor templates.

The Perforce adaptor template samples provide methods of analyzing changes to a baseline. Typically change analysis is used for build avoidance: if a baseline component has not changed, it is not rebuilt

Additional setup requirements for Perforce adaptors

See “Adaptor requirements” on page 439 for general requirements.

1. Install the P4 command-line client on the same host as the agent.
2. Set up a Perforce user account for Build Forge to use.
3. Set up the agent environment so that it can access the client.

Perforce adaptor template samples

The following adaptor template samples are provided.

PerforceByDate

1. Queries a Perforce client for changes that occurred since the adaptor execution.
2. Writes the following information to the BOM report: change, date, time, user, Perforce client, and comments.
3. Writes change details (from diff command output) to the BOM report.

Variables set in the adaptor:

- LAST_RUN
- CurDate
- P4PORT
- BFCLIENT
- P4CLIENT
- FILESPEC

Additional required variables for the environment:

- P4USER
- P4PASSWD

You cannot use the Assign Hidden property for these variables. The user and password are written to the step log in clear text.

PerforceByRev

1. Queries a Perforce client for changes that occurred since the last repository revision.
2. Writes the following information to the BOM report: change, date, time, user, Perforce client, and comments.
3. Writes change details (from diff command output) to the BOM report.

Variables set in the adaptor:

- LAST_RUN
- CurDate
- P4PORT
- BFCLIENT
- P4CLIENT
- FILESPEC

Additional required variables for the environment:

- P4USER
- P4PASSWD

You cannot use the Assign Hidden property for these variables. The user and password are written to the step log in clear text.

Integrating with StarTeam

BuildForge can work with StarTeam for source code management.

This topic describes additional setup requirements for integrating with StarTeam and summarizes the characteristics of the provided sample adaptor templates.

The StarTeam adaptor template samples provide methods of analyzing changes to a baseline. Typically change analysis is used for build avoidance: if a baseline component has not changed, it is not rebuilt

Additional setup requirements for StarTeam adaptors

See “Adaptor requirements” on page 439 for general requirements.

1. Install the StarTeam command-line client on the same host as the agent.
2. Set up a StarTeam user account for Build Forge to use.
3. Set up the agent environment so that it can access the client.

StarTeam adaptor template samples

The following adaptor template samples are provided.

StarTeamBaseline

1. Queries the folder for a StarTeam view to gather information about files.
2. Writes the following information to the BOM report: file name, status, revision, and branch.

Variables set in the adaptor:

- USER
- PASS
- HOST
- PORT
- PROJECT
- VIEW
- DIR
- BFCLIENT

Additional required variables for the environment:

- P4USER
- P4PASSWD

You cannot use the Assign Hidden property for these variables. The user and password are written to the step log in clear text.

StarTeamByDate

1. Uses the StarTeam API to query a StarTeam view to identify changes between the current date and the previous adaptor execution.
2. Writes the following information to the BOM report: changed files and directories, user, version, date, and change comments.
3. Writes change details (from diff command output) to the BOM report

Variables set in the adaptor:

- STA_USER
- STA_PASS
- STA_HOST
- STA_PORT
- STA_PROJECT
- STA_VIEW
- STA_DIR
- BFCLIENT
- LASTRUN
- CURDATE
- EMAILCHANGES
- STARTEAM80JAR

Additional required variables for the environment:

- P4USER
- P4PASSWD

You cannot use the Assign Hidden property for these variables. The user and password are written to the step log in clear text.

Integrating with Subversion

BuildForge can work with Subversion (SVN) for source code management.

This topic describes additional setup requirements for integrating with SVN and summarizes the characteristics of the provided sample adopter templates.

The SVN adaptor template samples provide methods of analyzing changes to a baseline. Typically change analysis is used for build avoidance: if a baseline component has not changed, it is not rebuilt

Additional setup requirements for SVN adaptors

See “Adaptor requirements” on page 439 for general requirements. There are no additional requirements for SVN.

SVN adaptor template samples

The following adaptor template samples are provided.

SubversionByDate

1. Queries Subversion for repository changes that occurred between a past date and the current date.
2. Writes the following information to the BOM report: change type, revision, user, file or directory, and change date.
3. Writes the following information to the BOM report: file name, status, revision, and branch.

Variables used:

- SVN_CLIENT
- SVN_OPTS
- SVN_REPOSITORY
- SVN_LAST_REV
- SVN_LAST_DATE

SubversionByRev

1. Queries Subversion for changes to a repository that occurred between the current revision and an earlier revision.
2. For each change, writes the following information to the BOM report: revision, user, change type, file or directory path, and change date.
3. Writes change details (from diff command output) to the BOM report.

Variables used:

- SVN_CLIENT
- SVN_OPTS
- SVN_REPOSITORY
- SVN_LAST_REV
- SVN_LAST_DATE

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.

About adaptors

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.

About the Adaptors panel

Use the Adaptors panel to create and edit adaptors. To view the panel, select **Projects** → **Adaptors**.

The screenshot shows the 'Adaptors' panel in Rational Build Forge. At the top, there are tabs for 'UI Config', 'Console', 'Reports', and a 'Logout: Root User' link. The 'Adaptors' tab is active, showing a table with one adaptor named 'MyAdaptor' of type 'Source'. Below the table, there are buttons for 'New Adaptor', 'Save Adaptor', 'Copy Adaptor', and 'Delete Adaptor'. The 'Details' section for the selected adaptor shows the following fields: Name (empty), Type (Source), Template (None), Access (Build Engineer), and Interface (empty).

The panel has the following fields:

Name A unique name for the adaptor that does not contain spaces

Type The adaptor type

Template

None or one of the adaptor templates included with Rational Build Forge

Note: Updating the Rational Build Forge version clears this setting.

Access

An access group to restrict adaptor viewing and editing to the group members

Interface

The XML interface you provide or one provided by a template

About adaptor links

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).

- If the adaptor runs and passes, then the remaining steps in the job are run.
- If the adaptor fails, then the job does not run at all. It is then purged and the build tag rolled back.

Note: Plan your system so that only one project can use an individual adaptor link at a time. Set the Run Limit setting for the project to 1. If two jobs run using the same adaptor link, then a failure does not roll back the build tag.

Note: You can set the special environment variable `_CI_BUILD_KEEP` to 1 to prevent a failed adaptor-link job from purging. This should be done only when debugging.

- Automatically populates an environment with application environment variables.
- Allows you to control whether the adaptor code is run 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. To use the same adaptor with another project, you must create another instance of the adaptor.

When you restart a job, the restart of any adaptor link steps in the job behaves as follows:

- If the State of the adaptor link is Inactive, the adaptor link step fails and causes the restart run to fail.
- If the State of the adaptor link is Active and the link fails, the adaptor link step fails and causes the restart run to fail.

In both of these cases, the build is not removed. However, the build status is set to Fail.

About the Adaptor Links panel

Use the Adaptor Links panel to connect an adaptor to a project and associate an environment to the adaptor. To view the panel, select **Projects** → **Adaptor Links**.

The screenshot shows the 'Adaptor Links' panel in a web application. At the top, there are tabs for 'UI Config', 'Console', and 'Reports', along with a 'Logout: Root User' link. The main panel has a header with 'Adaptor Links' and an 'Add Adaptor Link' button. Below this is a table with columns for 'Adaptor', 'Project', and 'State'. The table contains two rows: one for 'MyAdaptor' linked to 'Sav_hj' and another for 'MyAdaptor' linked to 'HelloWorld', both with a state of 'Active'. Below the table are buttons for 'New Adaptor Link', 'Save', 'Populate Env', and 'Delete'. At the bottom, a 'Details' section contains dropdown menus for 'State' (set to 'Active'), 'Adaptor' (set to '— Adaptor List —'), 'Entry Point' (set to '— Default —'), 'Project' (set to '— Projects —'), and 'Environment' (set to '— None —').

The panel has the following fields:

State One of Active, Debug, or Inactive

Note: These states apply only to the one adaptor link. You can set all adaptor links to the Debug state by using the Link Debug Mode system configuration setting. The state takes precedence over the system configuration setting.

Adaptor

The adaptor to link to a project

Entry Point

The entry point into the XML interface

Project

The project that is to use the adaptor

Environment

The environment for the project to use

Adaptor templates

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:

bfinstall\interface for computers running Microsoft Windows operating systems

bfinstall/Platform/interface for computers running UNIX or Linux operating systems

Adaptors and projects

To run 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.

Adaptors and environment variables

The adaptor requires environment variables to run 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.

Adaptors and notification

Most adaptor templates send email notification to users. For example, when the ClearCaseByDate adaptor runs, it sends pass email notification to users who changed source code files. If no files were changed, it sends a fail email notification.

You can optionally modify notification for adaptors:

- 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 does not exist), the XML fails, and all subsequent notifications 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” on page 447.

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 requirements

This topic identifies installation and configuration requirements and software requirements for Build Forge adaptors.

- Confirm that the external application version is supported by the Build Forge. See “Integration requirements for other products” on page 37 for the systems and versions supported.
- Install a Build Forge agent on the computer where the external application is running. Exception: ClearQuest, which requires that you install a ClearQuest client on the Build Forge host. There are additional requirements for ClearCase.
- Install a license key for the Build Forge Adaptor Toolkit if you want to use application templates other than ClearCase or ClearQuest.

When the requirements are satisfied, see “Setting up an adaptor” for instructions on setting up an adaptor.

Setting up an adaptor

This topic gives you information about creating and configuring adaptors:

- **Creating an Adaptor from a Template:** describes how to create an adaptor from an adaptor template.
- **Creating an Environment:** describes options for associating an environment with an adaptor.
- **Adding an Adaptor to a Project:** describes options for adding an adaptor to a project. Once the adaptor is added, it can be run by the project or a project step.
- **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.

Setting up an adaptor: task summary

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 and cannot contain spaces.
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.
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 **Edit** icon . The panel 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 adaptor properties using Perl comparison operators. String literals, numbers, or variables can be used for the comparison.

The syntax for the condition attribute is:

```
condition="true(<lvalue> <operator> <rvalue>)"
condition="false(<lvalue> <operator> <rvalue>)"
condition="hastext(variable)"
condition="isempty(variable)"
```

Specify one of four types:

Type	Description
true	Evaluation succeeds if lvalue and rvalue are equal.
false	Evaluation succeeds if lvalue and rvalue are not equal.
hastext	Evaluation succeeds if the value length is greater than 0.
isempty	Evaluation succeeds if the value 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 Operators
eq	==
ne	!=
gt	>
lt	<
ge	>=
le	<=

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.

Adaptor properties that 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 adaptor 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 adaptor 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**.

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.

Creating an adaptor from an adaptor template

Every adaptor is 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 and cannot contain spaces.
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**.

If you want to customize the adaptor template rather than use it as-is, please see “Customizing Adaptor Templates” on page 448.

Creating an environment for the adaptor

Adaptors require environment variables to run 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 when using an adaptor link

Use this method if you are using an adaptor link to connect the adaptor to a project.

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**. 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 when using adaptor dot commands

Use this method if you are running the adaptor from a project using an adaptor dot command. You have to add environment variables as well as create the environment.

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:

`<bfinstall>\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**.

Adding an adaptor to a project

To run 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.

Using an adaptor link to run the adaptor for a source project

The Adaptor link connects the adaptor to a project and connects application environment variables to the adaptor. The adaptor must be a source control 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.
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.

Using a dot command to run an adaptor in a step

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

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:

1. Make changes to your source files
2. Run the Build Forge project with the linked adaptor
3. Check the BOM report for information about changed source files
4. Check email for pass or fail notification

To test a linked adaptor:

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, and then select the **BOM** tab.

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:

1. Make changes to your source files
2. Run the Build Forge project with the adaptor dot command
3. Check the BOM report for information about changed source files
4. Check email 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, and 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, email 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.

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 have set the Enable Quickstart system configuration setting. The following steps assume you have set Enable Quickstart.

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.

Customizing Adaptor Templates

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:

- “Setting up an adaptor” on page 440
- “Adaptor template structure” on page 454
- “Adaptor reference” on page 454

This section gives you information about the following tasks:

- **Resetting adaptor templates:** describes the need to reset adaptor templates after you have upgraded Build Forge, added an adaptor template, or modified an adaptor template.
- **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 email notification.

- **Example: Removing change details from a BOM report:** describes how to remove change details from the BOM report.

Resetting adaptor templates

About this task

Resetting adaptor templates copies the latest adaptor templates from the `<bfinstall>\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 (either by copying and modifying an existing template or actually creating a new one).

Procedure

1. Select **Administration** → **System**.
2. In the list, select **Reset Adaptor Templates**.
3. At Reset Adaptor Templates, select **Yes**.
4. Click **Save**.

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:
`bfinstall\interface` for computers running Microsoft Windows operating systems
`bfinstall/Platform/interface` for computers running UNIX or Linux operating systems
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**.

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:
bfinstall\interface for computers running Microsoft Windows operating systems
bfinstall/Platform/interface for computers running UNIX or Linux operating systems
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” on page 449.

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:
bfinstall\interface for computers running Microsoft Windows operating systems
bfinstall/Platform/interface for computers running UNIX or Linux operating systems
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” on page 449.
9. Create a project for the adaptor.
10. Create an environment for the adaptor. See “Creating an environment for the adaptor” on page 443.
11. Create an adaptor using the new adaptor template. See “Creating an adaptor from an adaptor template” on page 443.
12. Add the adaptor to the project. See “Adding an adaptor to a project” on page 445.
13. Run the adaptor project to test the adaptor. See “Testing the adaptor” on page 446.

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 that does not have spaces and associate it with the adaptor template. See “Creating an adaptor from an adaptor template” on page 443.
3. Add the adaptor to the project using an adaptor dot command or an adaptor link.
 - The following example uses the .source adaptor command to add the adaptor to a project step that 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 runs when the project runs:
 - Place the <interface> element definition for the function to run as the first <interface> element in the template file
 - Set the default attribute for the <interface> element to true (default="true") for the function to run

Example: enabling email notification

Adaptor templates can be configured to send email 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, and 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 email 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 email to this group. The following setup sends email 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. Email 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 email user names in the SMTP server.

1. Open the ClearCaseByDate adaptor template in an XML editor.

- 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"/>`
- Set up notification to send email to this group. The following setup sends email 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="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>
```

- 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. Email notification is sent to that group when the build project completes.

Important: If the `<notify>` directive fails (for example, if the user the email is addressed to does not exist), the .xml fails, and all subsequent notifications fail.

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:

- Open the adaptor template in an XML editor.
- 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"/>
```

- 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>
```

- Find the `<bomformat>` section, and then find the `<section>` element for the `diff` command output. Remove the following lines:

```
<section name="diff">
  <field name="diff" title="Change Details"/>
</section>
```

- 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.

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:

bfinstall\interface for computers running Microsoft Windows operating systems

bfinstall/Platform/interface for computers running UNIX or Linux operating systems

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.

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 327.

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 455.

```

<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>

<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 an <interface> element or 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 condition="*function*" attribute to control whether the <adduser> element adds users who caused changes to a temporary access group. The value of *function* is an expression that evaluates to true or false. If the expression evaluates to true, the user is added to the temporary group; otherwise, the user is not added. See "Condition functions" on page 316.

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 condition="*function*" attribute to control whether the <bom> element is written to the BOM report. The value of *function* 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. See "Condition functions" on page 316.

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 to specify the display name for the category. A structure of <section> elements that 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 identify 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 an 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..." />
```

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 `<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 `condition="function"` attribute to control the execution of the commands in the `<execute>` element. The value of *function* is an expression that evaluates to true or false. If the expression evaluates to true, the system executes the commands; if the expression evaluates to false, the commands are not executed. See “Condition functions” on page 316.

```
<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 `condition="function"` attribute to control whether the `<field>` element is written to the BOM report. The value of *function* 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. See “Condition functions” on page 316.

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 computer, especially in the \integration directory.

When a <run> element calls the <command> element that 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>, <onproject>, and <adduser> 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 run 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 make 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:

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 `<adduser>`, `<setenv>`, `<bom>`, and `<field>` as subelements. See the reference information for “`resultsblock`” on page 460 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 runs the project steps. The element takes a required result attribute that 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 runs 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 use the state attribute to specify a value.

The <ontempenv> wraps a <step> element, which is run only if the temporary environment variable name and state exists after the <run> element commands are run.

```
<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> element 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 286.

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 <resultsblock> 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 then process different ranges by using different `<resultsblock>` elements. The following pseudocode shows the structure of a `<resultsblock>` element.

```
<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.

Attributes:

condition

Optional, in the form `condition="function"`. The value of *function* is an 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 316.

command

Required. It specifies name of a defined command to run. The command is named and defined in a `<command>` element.

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.

timeout

Required. It specifies how many seconds before the command times out.

Example showing required attributes:

```
<run command="UpdateEnv" params="" server="" dir="/" timeout="360"/>
```

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, in the form `condition="function"`. The value of *function* is an 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 316.

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. These examples show the two forms of the <step> element.

```
<step result="FAIL"/>
```

```
<step result="pass"/>
```

Chapter 27. Rational Team Concert integration

Build Forge can be integrated with Rational Team Concert.

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

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

Note: The information below describes integration with Rational Team Concert versions 1.x and 2.x. Starting with Rational Team Concert version 3.x, you will see pre-installed Build Forge engine and build definition selections in the Rational Team Concert client. The Rational Team Concert version 3.x server is ready to process Build Forge events. The pre-installed Rational Team Concert version 3.x Build Forge plug-in works with Build Forge 7.1.1.3 and later. For more information, see the Rational Team Concert documentation.

The integration includes these components:

- **Server Extension for Rational Team Concert Server:** this component enables communication between Rational Team Concert and Build Forge. The server extension is installed on the RTC server. Installation instructions are in “Installing Rational Team Concert Server Extension.” You allocate a Build Forge user to the Rational Team Concert server.
- **Client plug-in:** this component is a plug-in for the Rational Team Concert client. It allows users to access Build Forge to view projects, run jobs, and view job results. The client plug-in must be installed on each RTC client that will access Build Forge. Installation instructions are in “Installing the client plug-in for Rational Team Concert” on page 466.
- **Adaptor:** this component is a source type adaptor to allow projects to access files in the Rational Team Concert repository. A different adaptor is used for RTC version 1.x and RTC version 2.x. See “Configuring the Rational Team Concert adaptor” on page 469.

For additional integration and troubleshooting information, see https://jazz.net/wiki/bin/view/Main/RationalBuildForge/IntegrationWithRTC#Install_the_Build_Forge_RTC_Serv.

Installing Rational Team Concert Server Extension

Before you begin

Prerequisites:

- Rational Team Concert must be version 1.x or 2.x. Note where the instructions and files differ by version in the procedure.
- The Build Forge system must be running.

About this task

Perform these steps from the Rational Team Concert Server host:

Procedure

1. Log on to the Rational Team Concert server host.
2. Navigate to the Build Forge server and, using the appropriate URL for your version of Rational Team Concert, download the server extension to a temporary location.
 - Rational Team Concert version 1.x:
`http://<bf_console_hostname>/rtc-server/BuildForgeConnectorServer.zip`
 - Rational Team Concert version 2.x:
`http://<bf_console_hostname>/rtc2-server/BuildForgeConnectorServer.zip`
 - If you are running Rational Team Concert on the same system that Build Forge is running on, you can use localhost as the host name.
 - Include the port number if the console is running on a port other than 80.
Example: `http://myhostname:11812/clients`.
3. Extract the file contents to *RTC_install/jazz/server*. The following files are added if you downloaded the extension for Rational Team Concert version 1.x:

RTC_install/jazz/server/buildforgeconnector-update-site (directory)
RTC_install/jazz/server/provision_profiles/buildforgeconnector-profile.ini

The following files are added if you downloaded the extension for Rational Team Concert version 2.x:

RTC_install/jazz/server/buildforgeconnector-update-site (directory)
RTC_install/jazz/server/conf/provision_profiles/buildforgeconnector-profile.ini

To run the RTC server in WebSphere Application Server, you must edit the *buildforgeconnector-profile.ini* file to fully qualify the path to the *buildforgeconnector-update-site* directory.

4. Restart the Rational Team Concert Server.

Results

The Rational Team Concert server consumes a Build Forge user license to communicate with the console.

Installing the client plug-in for Rational Team Concert

Install the client plug-in for Rational Team Concert from the Build Forge server.

Before you begin

Prerequisites:

- Rational Team Concert version 1.x or version 2.x - a different plug-in is required for each.
- The Build Forge Connector server extension must be installed on the Rational Team Concert server.
- The Build Forge system must be running.
- Each user should have two login IDs for the console. Build Forge limits a user to one login session. The same user name cannot be logged in through the Management Console and the RTC client at the same time. When a new session is started, any existing session is terminated. RTC client users should have two IDs:
 - UserRTC - used when they configure a build definition to run a build. This user name is used to access the Build Forge console to run the build.

- UserBF - used when they check build results. A direct login to the Build Forge Management console is required within a window in the RTC client.

About this task

The client plug-in must be installed on every Rational Team Concert client that will access Build Forge.

To install the plug-ins, perform these steps from a Rational Team Concert client.

Procedure

1. Select **Help** → **Software Updates** → **Find and Install**.
2. Select **Search for new features to install**, and then click **Next**. The system displays the **Update Sites to Visit** dialog.
3. Click **New Remote Site**. The system displays the **New Remote Site** dialog.
 - a. Enter Build Forge Connector Update Site in the name field.
 - b. Enter the update site URL in the **URL** field. The URL varies according to the version of Rational Team Concert that you are using.

Rational Team Concert version 1.x:
`http://<console_host_name>/rtc/site.xml`

Rational Team Concert version 2.x:
`http://<console_host_name>/rtc2/site.xml`
 - If you are running Rational Team Concert on the same system that Build Forge is running on, you can use localhost as the host name.
 - Include the port number if the console is running on a port other than 80. The following example works if you are using Rational Team Concert version 1.x and Build Forge is running on port 11812.
`http://myhostname:11812/rtc/site.xml`
 - c. Click **OK**.
4. In the **Update sites to visit** dialog, select the **Build Forge Connector Update Site** check box, and then click **Finish**.
5. The system displays a list of available features from the update site in the **Search Results** dialog. Select all the offered features, then click **Next**.
6. Read the license agreements, then select **I accept the terms in the license agreements**, then click **Next**.
7. Select the location where you want to install the features. To add a new location, click **Change Location**, then browse to the location you want.
8. Click **Finish**.
9. If a **Feature Verification** dialog is shown, click **Install**. The dialog appears because the plug-ins are unsigned features. The dialog is shown once for each feature installed unless you selected **Install All**.
10. You are asked to restart Rational Team Concert to make the changes take effect. Click **Yes**.
11. Restart Build Forge.

Results

When the integration is complete, you can use the plug-in to run jobs and examine job results. You must specify RationalBuildForgeConnector as the Build Engine for your builds.

Alternate Installation when SSL is enabled

Use an alternate installation method when SSL is enabled on the Build Forge system.

Before you begin

The current version of Rational Team Concert is not enabled for SSL. Therefore the plug-in cannot be installed from the Build Forge system when the Build Forge system has SSL enabled. As a workaround, make the plug-in installation files available on a non-secure web server or distribute them to users manually.

The following files are required from either `<bfinstall>/webroot/public/rtc` (for RTC version 1.x) or `<bfinstall>/webroot/public/rtc2` (for RTC version 2.x):

- features directory
- plugins directory
- site.xml file

About this task

Once you have made the `rtc` or `rtc2` directory available, users perform these steps from within their Rational Team Concert client:

Procedure

1. Select **Help** → **Software Updates** → **Find and Install**.
2. Click **Search for new features to install**, and then click **Next**. The system displays the **Update Sites to Visit** dialog.
3. Create a new site. Choose one of the following procedures.
 - Getting the files from a remote server:
 - a. Click **New Remote Site**. The system displays the **New Remote Site** dialog.
 - b. Enter “Build Forge Connector Update Site” in the name field.
 - c. Enter the location of the files:
`http://host/prism/eclipse/updateSite/site.xml`
The *host* is the host name or IP address of the web server.
The *path* is the path from the server root to where you placed the files.
 - d. Click **OK**.
 - Getting the files from the local host:
 - a. Click **New Local Site**. The system displays the **New Local Site** dialog.
 - b. Enter “Build Forge Connector Update Site” in the name field.
 - c. Enter the location of the files:
`file://path/prism/eclipse/updateSite/site.xml`
The *path* specifies the location of the files.
 - d. Click **OK**.
4. In the **Update sites to visit** dialog, select the **Build Forge Connector Update Site** check box, and then click **Finish**.
5. The system displays a list of available features in the **Search Results** dialog. Select all the offered features, and 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**, and then click **Next**.
7. Select the location where you want to install the features. To add a new location, click **Change Location**, and then browse to the location you want.
8. Click **Finish**.
9. If a **Feature Verification** dialog is shown, click **Install**. The dialog appears because the plug-ins are unsigned features. The dialog is shown once for each feature installed unless you selected **Install All**.
10. You are asked to restart Rational Team Concert to make the changes take effect. Click **Yes**.
11. Restart Build Forge.

Configuring the Rational Team Concert adaptor

About this task

An adaptor to connect Build Forge to Rational Team Concert source repository is installed automatically with Build Forge. You can configure individual Build Forge projects to access Rational Team Concert source as follows.

Procedure

1. Define a Server resource.
 - Enter the **Name** for this server resource.
 - Enter the **Host**: this is the fully qualified domain name of the host where Rational Team Server is running.
 - Enter values for other properties as appropriate.

Note: The RTC tool scm must be on the PATH for the user profile or the startup profile for the RTC server. The adaptor uses scm to access source files.
2. Set up adaptor links for each project that uses the Rational Team Concert repository. The following instructions describe how to set up an adaptor link for a project.
 - a. In the console, go to **Projects** → **Adaptor Links**.
 - b. Click **New Adaptor Link**. Set its properties in the **Details** tab, and then click **Save**.
 - State - select **Active**.
 - Name - enter the name for the adaptor link.
 - Adaptor
 - For Rational Team Concert version 1.x, choose JazzSCM
 - For Rational Team Concert version 2.x, choose JazSCMv2
 - Project - choose the project to which to apply the adaptor link.
 - Environment - choose the environment to use for this adaptor link. It must be an existing environment.
 - c. Click the adaptor link you just created.
 - d. Click **Populate Env**. This step populates the specified environment with the variables defined in the adaptor (JazzSCM.xml).
 - e. Click **Save**.
3. Update the variables provided by the adaptor. In the environment you set up and populated, edit the variables provided by the adaptor.

Four environment variables are set in the adaptor:

- **Current_Date** - sets the current date. It is used to apply timestamps. Do not change this definition.
- **Last_Run** - automatically updated by the system. Do not change this definition.
- **Directory_Path** - sets the location for source files retrieved from the repository. It is set to C:\temp by default. Change this directory to the temporary directory you want to use.

The directory is not cleaned up by default after a job runs. Delete old directories from jobs that have already run.

- **Jazz_Server** - sets the location of the Rational Team Concert server. It is set to \$BFServer by default. You must change this setting to the **Name** property of the Server resource that points to the Rational Team Concert Server.
 - a. In the console, go to **Environments**.
 - b. Select the environment that is used by the adaptor link.
 - c. Select and edit the variables you want to change.
 - d. Click **Save**.

Results

You can now run the project. Each time it runs, it connects to the Rational Team Concert repository. It updates source files that have changed at the RTC server since the last time the project ran.

Additional resources:

- Rational Team Concert version 1.x: the JazzSCM adaptor is located in `<bfinstall>/interface/JazzSCM.xml`. You can open it with a text editor or XML reader.
- Rational Team Concert version 2.x: the JazzSCMv2 adaptor is located in `<bfinstall>/interface/JazzSCMv2.xml`. You can open it with a text editor or XML reader.
- The SCM commands used by the adaptor are documented in the Rational Team Concert documentation. Use them and others when testing the connection to the server and testing commands to be used by the adaptor.

Chapter 28. Other IDE integrations

With plug-ins, you can integrate Rational Build Forge with integrated development environments (IDEs).

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.

With the provided plug-ins, developers can complete these tasks directly from their IDEs:

- 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 plug-in, 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 computer to the server running the build. The command runs before the project step. Example: You can use this command to 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 run. 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. The system replaces the first \$1 in the command with the directory name.

_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. The system replaces the first \$1 in the command with the file name.

_PRISM_DIR_POSTCMD

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. The system replaces the first \$1 in the command with the directory name.

_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[™] and Rational[®] Application Developer IDEs.

The following plug-ins are available for the Eclipse and Rational Application Developer environments:

Frequency

With the Frequency plug-in, a developer can complete these tasks:

- 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 variable 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 about using Frequency, see the online 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 online 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.

Before you begin

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

About this task

To install the plug-ins, complete this procedure from within your IDE.

Procedure

1. Select **Help** → **Software Updates** → **Find and Install**.
2. Click the **Search for new features to install** radio button, and 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 computer: `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, and then click **Finish**.
 5. The system displays a list of available plug-ins in the **Search Results** dialog. Select all the offered plug-ins, and 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, select **I accept the terms in the license agreements**, and 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.

Before you begin

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:

1. Create a directory named `prism`.
2. In `prism`, create a directory named `eclipse`.
3. 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

About this task

After you have made the prism directory available, users perform these steps from within their IDE:

Procedure

1. Select **Help** → **Software Updates** → **Find and Install**.
2. Click the **Search for new features to install** radio button, and 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, and then click **Finish**.
5. The system displays a list of available plug-ins in the **Search Results** dialog. Select all the offered plug-ins, and 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, select **I accept the terms in the license agreements**, and then click **Next**.
7. Select the location where you want to install the features. To add a new location, click **New Location**, and 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):

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.
 4. Enter a hostname or IP address in **Build Forge Services Layer Hostname**, verify or edit the other fields, and click **OK**.

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 online 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 complete these tasks:

- 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 about setting up the integration, see Chapter 27, “Rational Team Concert integration,” on page 465.

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 about setting up the integration, see Chapter 27, “Rational Team Concert integration,” on page 465.

To set up a build definition and run a build:

1. Set up a build definition.
 - a. In the Team Artifacts view, expand the project folder.
 - b. Right-click **Builds**, and then click **New Build Definition**.
 - c. In New Build Definition, select **Create a new build**, and 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, and 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, and then select **Request build**.
 - b. Specify the Build Options and Build Properties you want, and 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**, and 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 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” on page 216.

Chapter 29. WebSphere integrations

This section describes ways to integrate Build Forge[®] with WebSphere products:

- Using WebSphere Application Server rather than Apache Tomcat to run the Build Forge services and Quick Reports
- Using IBM HTTP Server (IHS) rather than Apache as the web application server

Using WebSphere Application Server instead of Apache Tomcat

You can manually configure WebSphere Application Server 6.1 or 7.0 to be the application server for the Build Forge services and Build Forge Help applications instead of using the provided Apache Tomcat application server.

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

Prerequisite: ensure WAS has the latest fixpack applied before you attempt to deploy the Build Forge services layer.

Important: When you use this configuration, you must start the Build Forge services and Build Forge Help applications in WebSphere Application Server *before* you start Build Forge. They are not started automatically.

Perform the steps in the following procedure from the WebSphere Admin Console.

1. Open the WebSphere Admin Console.

The URLs for the console are:

- `http://<was_host>:<was_port>/ibm/console`; 9060 is the default port
- `https://<was_host>:<was_port>/ibm/console`; the default port is 9443. Use this URL if WebSphere Administrative Security is enabled.

2. Configure support for your JDBC driver.

- a. Create a new variable, RBF_JDBC_DRIVER_PATH.

Create the variable in **Environment > WebSphere Variables**. Its scope should be the WAS node and server. Set the value to the directory that contains your database driver JAR files.

- b. Save the change to the master configuration.
- c. Stop and restart the WebSphere server to make the new variable available.
- d. Create a new shared library, RBF_JDBC_LIBRARY.

Create the library in **Environment > Shared Libraries**. Add the JAR file names for your JDBC device driver, using the RBF_JDBC_DRIVER_PATH that you just created. If you have more than one jar file to list, place each jar file in its own line (as is the case with DB2).

The following example is for a MySQL database driver:

```
${RBF_JDBC_DRIVER_PATH}\mysql-connector-java-5.0.5-bin.jar
```

The next example is for DB2:

```
${RBF_JDBC_DRIVER_PATH}\db2jcc.jar  
${RBF_JDBC_DRIVER_PATH}\db2jcc_license_cu.jar
```

With Unix or Linux, in these examples, use the forward slash (/) instead of backslash (\).

- e. Save the change to the master configuration.
3. Install the Build Forge application WAR file.
 - a. Open **Applications > Enterprise Applications**.
 - b. Click **Install**.
 - c. Browse to the rbf-services.war file in the Build Forge installation directory. Use /rbf-services as the context root.
 - d. Click **Next**, and clear the following check box if it is selected:
 - Create MBeans for resources
 - e. Click **Next** until you see a **Finish** button, then click **Finish**.
 - f. Click **Save** at the bottom of the installation text.
4. Install the Build Forge Help WAR file.
 - a. Open **Applications > Enterprise Applications**.
 - b. Click **Install**.
 - c. Browse to the BuildForgeHelp.war file in the Build Forge installation directory. Use /BuildForgeHelp as the context root.
 - d. Click **Next**, and clear the following check box if it is selected:
 - Create MBeans for resources
 - e. Click **Next** until you see a **Finish** button, then click **Finish**.
 - f. Click **Save** at the bottom of the installation text.
5. Set RBF_JDBC_LIBRARY as a shared library reference.
 - a. Open **Applications > Enterprise Applications**.
 - b. Click the **rbf-services.war** link.
 - c. Click **Shared library references**.
 - d. Select the **A Services Layer Login Servlet** box.
 - e. Click **Reference Shared Libraries**.
 - f. Add RBF_JDBC_LIBRARY to the list.
6. Under **Manage Modules**, select **A Services Layer Login Servlet** and locate **Class loader order** in the drop down box. Change that value to **Classes loaded with parent class loader first**.
7. Save the changes to the master configuration.
8. Check the buildforge.conf file in the <BF_INSTALL_DIR> (or <BF_INSTALL_DIR>/Platform directory on *nix) to ensure that the services_url entry in the file points to the correct URL. If you are using WAS, the value for this property in buildforge.conf should look like: https://<hostname>:9443/rbf-services

Starting the Build Forge applications in WebSphere Application Server

Start Build Forge services and the Build Forge Help on the WebSphere Application Server, as follows:

1. Choose **Applications > Enterprise Applications**.
2. Select **rbf-services.war** and **BuildForgeHelp.war**.
3. Click **Start**.

Enabling application support for Java 2 security in WebSphere Application Server

If you are running WebSphere Application Server with Java 2 Security enabled, you must configure the Build Forge services layer to use it.

Perform the following steps from the WebSphere Admin Console:

1. Open **Applications > Enterprise Applications**.
2. Select **rbf-services_war** and click **Update**.
3. Select **Replace or Add a Single File**.
4. In the **Specify the path beginning with the installed application archive file to the file to be replaced or added**, enter **META-INF/was.policy**.
5. Select **Local file system** and browse to the directory **<bfinstall>/samples/projects/was.policy**. Click **Next**.
6. Click **OK**.
7. Save the changes to the master configuration, and then stop and start the application.

Prerequisite to enable WAS for Build Forge SSL and password encryption

If you plan to enable SSL for Rational Build Forge, you must first complete these steps:

1. Set the following property on the WebSphere Application Server computer that hosts the Rational Build Forge services component.
 - Windows
-Dcom.buildforge.client.config=<bfinstall>\bfclient.conf
 - UNIX or Linux
-Dcom.buildforge.client.config=<bfinstall>/Platform/bfclient.conf
2. Restart WebSphere Application Server.

If you plan to enable password encryption for Rational Build Forge, you must first complete these steps:

1. Set the following property on the WebSphere Application Server computer that hosts the Rational Build Forge services component.
 - Windows
-Dcom.buildforge.password.encryption.file=<bfinstall>\bfpwcrypt.conf
 - UNIX or Linux
-Dcom.buildforge.password.encryption.file=<bfinstall>/Platform/bfpwcrypt.conf
2. Restart WebSphere Application Server.

Note: If you are using both SSL and password encryption you must set both of these properties on the WebSphere Application Server computer that hosts the Rational Build Forge services component.

Using IBM HTTP Server instead of Apache HTTP Server

You can configure IBM HTTP Server (IHS) for use with the Management Console instead of the Apache HTTP Server that is installed by default.

About this task

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

Prerequisite: Contact support for assistance. They help you set up your IHS with PHP compiled specially to support Build Forge. After you have modified your IHS installation you can continue to set up IHS as described in this section.

The following sections describe how to do the following tasks:

1. Modify your IBM HTTP Server configuration files to point to the Build Forge web application.
2. If you use a proxy server to access the database, modify PHP to use the proxy server.
3. If you use SSL, configure IHS to work with Build Forge through SSL.

Edit the IBM HTTP Server configuration file

Before you begin

Edit the IBM HTTP Server httpd.conf file to add information for the Build Forge web server.

Procedure

1. Locate the httpd.conf file for the IBM HTTP Server (IHS) in the conf directory of your server installation.
2. Modify the DocumentRoot setting to point to the Build Forge web application, as shown in the example. In this example, the Build Forge installation directory is /opt/buildforge.

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

3. If necessary, change the IHS port number. The default port number is 80. Make any other necessary changes to httpd.conf.

Identify Proxy Server in PHP

About this task

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

Procedure

Edit the PHP configuration file php.ini. It is located in *<php-install>/lib*, for example /usr/local/php-5.2.4.

Add the following entries:

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

Configuring SSL for IHS

About this task

In addition to the normal SSL setup for IHS, there are additional requirements for it to work with Build Forge.

Procedure

1. Include Build Forge tool directories in your PATH.
 - Windows
 - *bfinstall\openssl*
 - *bfinstall\ibmjdk\bin*
 - UNIX or Linux
 - *bfinstall/openssl*
 - *bfinstall/server/ibmjdk/bin*
2. Include Build Forge tool directories in your JAVA_HOME.
 - Windows
 - *bfinstall\ibmjdk*
 - UNIX or Linux
 - *bfinstall/server/ibmjdk*
3. Convert the Build Forge keys from PKCS12 to CMS. Use the GSKIT tool. In gsk7\bin (Windows) or bin (UNIX or Linux), run the following command (line breaks are added for clarity):

```
gsk7cmd -keydb
        -convert
        -db bfinstall\keystore\buildForgeKeyStore.p12
        -pw buildForgeKeyStore_password
        -old_format pkcs12
        -new_format cms
```
4. Store the password in a stash file. IHS uses this file to get the password during startup. Without it, IHS prompts for the password. Use the GSKIT tool. In gsk7\bin (Windows) or bin (UNIX or Linux), run the following command (line breaks are added for clarity):

```
gsk7cmd -keydb
        -stashpw
        -db bfinstall\keystore\buildForgeKeyStore.kdb
        -pw buildForgeKeyStore_password
```
5. Modify httpd.conf. Include the following entries for Windows:

```
LoadModule ibm_ssl_module modules/mod_ibm_ssl.so
Listen 0.0.0.0:443
<VirtualHost *:443>
SSLEnable
SSLClientAuth None
SSLProtocolDisable SSLv2
SSLServerCert buildforge
KeyFile bfinstall\keystore\buildForgeKeyStore.kdb
SSLStashFile bfinstall\keystore\buildForgeKeyStore.sth
ErrorLog bfinstall\Apache\logs\ssl_error.log
TransferLog bfinstall\Apache\logs\transfer.log
</VirtualHost>
```

Include the following entries for UNIX and Linux:

```
LoadModule ibm_ssl_module modules/mod_ibm_ssl.so
Listen 0.0.0.0:443
<VirtualHost *:443>
SSLEnable
SSLClientAuth None
```

```
SSLProtocolDisable SSLv2
SSLServerCert buildforge
KeyFile bfinstall/Platform/keystore/buildForgeKeyStore.kdb
SSLStashFile bfinstall/Platform/keystore/buildForgeKeyStore.sth
ErrorLog bfinstall/server/apache/logs/ssl_error.log
TransferLog bfinstall/server/apache/logs/transfer.log
</VirtualHost>
```

bfinstall is the root of the installation directory. For the steps above, on UNIX and Linux, you will often need to append */Platform* to *bfinstall* to reach the correct directory. You also need to use slash (/) instead of backslash (\) to separate directories.

Results

Consult IHS documentation on setting up SSL for more information.

Chapter 30. Working with APIs

Build Forge provides a Java client API and a Perl client API.

Client files are stored in `<bfinstall>/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 for the APIs to communicate with Build Forge. By default the ports are set to the following:

- 3966 (non-secure)
- 49150 (secure - SSL enabled)

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 to communicate with Build Forge. For more information, see “Enabling SSL for an API client (Perl or Java)” on page 119.

Creating a Build Forge user for API programs

Create a user on the Management Console for programs to use to authenticate.

About this task

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 with user credentials.

After authentication, a session ID is generated for the user session and stored in the database. If the program uses the same login as an existing user, that user session closes.

Only one thread or process can use the user credentials. If another thread or process attempts to use the same user credentials to establish a session, it causes the original session to be terminated.

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.

Java SDK 1.5 or 1.6 is required for use with the Java client API.

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.

Procedure

To download the Java API:

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.

Results

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 Java SDK to use it.

About this task

The host will act as a client to the Management Console host.

Procedure

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 provides access to an abstraction of Management Console data objects and methods.

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>.

To use the Perl Client, you must:

- Get the Perl Client package from your Management Console computer.
- 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.

Procedure

To download the Perl client API:

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.

Results

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.

About this task

The host acts as a client to the Management Console host.

Procedure

1. Install a Perl interpreter on the client host, such as ActiveState's ActivePerl version 5.8.4 or later. 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 about 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
```

After being installed, the top Perl client module is BuildForge::Services::DBO. See the PerlDoc for each module for more information.

Chapter 31. Determining the Management Console version number

To find out what version of the Management Console you are working with, place your mouse cursor over the Rational Build Forge logo in the upper-left corner of the page. The system displays the version number in a pop-up tool tip.

Chapter 32. Executable commands installed with the product

The following table describes the executable commands provided and used by Rational® Build Forge®.

On Windows, the command files are in the Build Forge installation directory, which by default is C:\Program Files\IBM\Build Forge.

On UNIX/Linux, the command files are in the *<bfinstall>*/Platform directory, where *<bfinstall>* is by default /opt/buildforge.

Note: If you are running the Management Console on z/Linux, you must specify the .pl extension to run a command.

To display the version number for any executable command, use the -v option. You must run the command from the directory where the executable commands are installed.

```
bfproject -v
```

The -v option for any command displays the command name and its version number, as shown in the following example:

```
bfproject.exe 7.0.351
```

Executable	Service?	Description
bfdbmigrate	N	Use this migration script to convert a default database from version 3.8 (MySQL) to DB2® after the schema has been converted to version 7.0. This script is intended to be used <i>after</i> using bfmigrate.exe/bfmigrate.pl.
bfproject	N	buildforge.exe issues this command to start a job.
bfengine	Y	This command starts buildforge.exe and the web server. Windows only.
bfexport	N	Use this utility to export data from the database.
bfomexport	N	Use this utility to export the BOM from the database.
bfimport	N	Use this utility to import project data into the database.
bfmigrate	N	Use this migration script to upgrade a version 3.8 database to version 7.0.
bfpurge	N	buildforge.exe issues this command to purge builds.
bfrefresh	N	The build system issues this command to update the manifests for servers.
bfsched.exe or bfsched.pl	N	buildforge.exe issues this command to check the database for scheduled jobs and start them when appropriate.

Executable	Service?	Description
bfstepcmd	N	bfproject issues this command for long-running steps, to create a separate process for them.
buildforge	N	This command manages build, purge, and schedule processes.
console_uninst	N	Use this command to uninstall the Management Console. Windows only.
bfdispatch	Y	This command starts the agent service. Windows only.
bfagent	N	Agent executable
bfpwncrypt	N	Utility for encrypting passwords

Chapter 33. 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, and 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. An agent 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 runs 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 this 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.

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 **Servers** → **Collectors** panel.

database

The database stores all the information entered into the Management Console. Also, the database stores 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 emails, and communicate with agents (running on servers).

environments

An environment is a container for a list of 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.

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 to determine the server on which it runs. 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. You can also update them manually. Use the **Refresh Server Manifest** button while viewing the server in **Servers** → *servername*.

Management Console

A component of the system that is installed on a single computer to coordinate the system. You log in to the Management Console to define and run projects 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 email 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.

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 use dynamic information. For example, a selector could specify a server that has designated properties, such as CPU type or disk size or current load. 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 selectors 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, a project or step that requires exclusive use of a resource obtains a semaphore to ensure that exclusive use.

You set a semaphore in a step by using the **.semget** command. It is released in a separate step by the **.semput** command. After you obtain the semaphore, no other step can get it. Steps that attempt to obtain the semaphore 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 computer to be available as a server in Build Forge, you must do the following:

- Install an agent on the computer (see Chapter 11, “Installing agents,” on page 143 for more information).
- Create a server resource using the Management Console.

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 to run. The selector associated with a step determines what server to use. If none is specified, the selector for the project is used. Step properties determine how the step is run 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 a step to see its step log.

threading

The manner whereby various related transactions are run concurrently.

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.

Appendix. Notices

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:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamoto-shi
Kanagawa, 242-8502 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.

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:

Intellectual Property Dept. for Rational Software
IBM Corporation
5 Technology Park Drive
Westford, MA 01886
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 is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 2010.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Windows is a registered trademark of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.