

HP Application Lifecycle Intelligence

For the HP ALM Platform

Software Version: 2.0

User Guide

Document Release Date: December 2011

Software Release Date: December 2011



Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

© Copyright 2003 - 2012 Hewlett-Packard Development Company, L.P.

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

<http://h20230.www2.hp.com/selfsolve/manuals>

This site requires that you register for an HP Passport and sign in. To register for an HP Passport ID, go to:

<http://h20229.www2.hp.com/passport-registration.html>

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HP Software Support Online web site at:

<http://www.hp.com/go/hpsoftwaresupport>

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

<http://h20229.www2.hp.com/passport-registration.html>

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Contents

User Guide.....	1
Contents.....	6
Welcome to Application Lifecycle Intelligence.....	8
How This Guide is Organized.....	10
Setting Up ALI.....	11
Prerequisites for ALI.....	11
The Installation Process.....	11
Deploying the ALI Extension.....	12
Enabling the ALI Extension.....	12
Upgrading Projects.....	13
Undeploying the ALI Extension.....	13
Perforce Migration.....	13
SCM System Integration.....	14
Supported SCM Systems.....	14
SCM Agents.....	14
Configuring Repositories.....	15
Adding Repositories.....	15
Setting External Repository Viewer.....	16
Setting Branches and Enforcement.....	17
Setting Branches.....	17
Setting Commit Patterns.....	19
Setting Change Detection.....	21
Build System Integration.....	22
Supported Build Systems.....	22
Hudson/Jenkins Installation.....	23
HP ALI Hudson/Jenkins Plugin Installation.....	23
HP ALI Hudson/Jenkins Plugin Configuration.....	24
Adding a Build Server.....	24
Adding Build Configurations.....	25
Reusing SCM Configurations from Build Configurations.....	25

Setting Build Configuration Defect Filters.....	26
Customizing ALI Project Lists.....	26
Setting Build Server Detection.....	27
Force.com Integration.....	28
Project Deployment, Testing and Report Generation.....	28
Hudson/Jenkins Force.com Configuration.....	30
Monitoring SCM Changes and Traceability.....	31
Viewing the Code Change Table.....	31
Code Change Details.....	32
Viewing Change Impact Report.....	32
Generating Project Reports.....	33
Generating Graphs.....	33
Monitoring Development Activity.....	35
Development Activity Releases.....	35
Development Activity Defects.....	35
Development Activity Requirements.....	36
Monitoring Development Activity in Defect Details.....	36
Monitoring Build Activity.....	38
Viewing Builds.....	38
Build Details.....	39
Requirements Tab.....	39
Defects Tab.....	39
Code Changes Tab.....	40
Active Developers Tab.....	40
Build Reports.....	40
Generating Build Graphs.....	40

Chapter 1

Welcome to Application Lifecycle Intelligence

Application Lifecycle Intelligence (ALI) is a set of capabilities, reports and metrics providing complete ALM traceability that enables ALM stakeholders to make informed decisions.

ALI extends the HP ALM platform model with Source Code Management (SCM) and Build Management integration and establishes linkage between ALM Entities (Releases, Requirements, Defects ...), code changes, builds, unit test results and code coverage analysis. This transforms HP ALM into the central system of record that provides information about both business and development activities.

Control changes committed to SCM

With ALI, it is possible to define SCM policies that control committed changes. The enforcement of the SCM policies helps to ensure that developers follow prescribed guidelines and best practices. Team leaders will be confident that developers implement the right features and add required metadata to the committed change sets. Policy enforcement also provides invaluable help during stabilization periods as it is easy to ensure that developers fix only severe defects or to lock the code base of a release completely.

For details, see ["Setting Branches and Enforcement" \(on page 17\)](#).

Developers, QA managers and Project Managers can easily review:

- Code changes implementing a requirement or defect.
- Code changes implemented in specific builds.
- What specifically was implemented during a specific time period, release, build or by an individual contributor.
- Which requirements and defects are affected.
- Amount of changes associated with a particular requirement .
- Line by line differences in change sets.
- Changes not associated with either requirements or defects.
- Build & Quality metrics.
- Aggregated information about code coverage, test results and amount of changes.

For details, see ["Monitoring SCM Changes and Traceability" \(on page 31\)](#), ["Monitoring Development Activity" \(on page 35\)](#) and ["Build Details" \(on page 39\)](#).

Application Lifecycle Intelligence enables this ALM traceability by integrating ALM requirements, tests and defects with popular open source and commercial SCM, Build System, Unit Testing and Code Coverage Analysis tools.

ALI 2.0 provides integration with:

- **SCM Systems**

- Subversion (SVN)
- Concurrent Versions System (CVS)
- Microsoft Team Foundation Server (TFS)
- Perforce

The integration is SCM client agnostic; developers can commit changes from their current SCM clients whether it is a command line utility or an IDE like Eclipse with the Tasktop plugin. For details, see ["Supported SCM Systems" \(on page 14\)](#).

- **Build Systems**

- Jenkins
- Hudson

ALI provides plug-ins for build servers that automatically extract build information and metrics and embed them into the ALM model. For details, see ["Build System Integration" \(on page 22\)](#).

- **Unit Testing**

- JUnit
- TestNG
- NUnit

- **Code Coverage Analysis**

- Cobertura
- NCover

Note: This guide explains how to work with Application Lifecycle Intelligence in conjunction with ALM. For more information on using ALM, refer to the *HP Application Lifecycle Management User Guide*.

How This Guide is Organized

This guide contains the following chapters:

- **Chapter 1** ["Welcome to Application Lifecycle Intelligence" \(on page 8\)](#)
An overview of Application Lifecycle Intelligence.
- **Chapter 2** ["Setting Up ALI" \(on page 11\)](#)
Installing and configuring Application Lifecycle Intelligence.
- **Chapter 3** ["SCM System Integration" \(on page 14\)](#)
Integrating Application Lifecycle Intelligence with SCM systems.
- **Chapter 4** ["Build System Integration" \(on page 22\)](#)
Integrating Application Lifecycle Intelligence with build systems.
- **Chapter 5** ["Force.com Integration" \(on page 28\)](#)
Integrating Application Lifecycle Intelligence with Force.com.
- **Chapter 6** ["Monitoring SCM Changes and Traceability" \(on page 31\)](#)
How to view change sets data.
- **Chapter 7** ["Monitoring Development Activity" \(on page 35\)](#)
How to view development activity.
- **Chapter 8** ["Build Details" \(on page 39\)](#)
How to view build data.

Chapter 2

Setting Up ALI

This chapter describes the installation requirements and setup procedures for using Application Lifecycle Intelligence (ALI).

This chapter includes:

["Prerequisites for ALI" \(on page 11\)](#)

["The Installation Process" \(on page 11\)](#)

["Deploying the ALI Extension" \(on page 12\)](#)

["Enabling the ALI Extension" \(on page 12\)](#)

["Upgrading Projects" \(on page 13\)](#)

["Undeploying the ALI Extension" \(on page 13\)](#)

If upgrading from Application Lifecycle Intelligence 1.1 to 2.0, please see ["Perforce Migration" \(on page 13\)](#).

Prerequisites for ALI

To use ALI, the following must be installed:

- ALM Edition of HP Application Lifecycle Management 11.00 SP2 or later (on server machine)

The minimum system requirements to run ALI are the same as for ALM Platform, as described in the *HP Application Lifecycle Management Installation Guide*.

ALI also requires SCM and build systems to integrate with. For details, see ["Supported SCM Systems" \(on page 14\)](#) and ["Supported Build Systems" \(on page 22\)](#).

The Installation Process

This section describes the ALI installation process.

To install ALI:

1. Ensure that ALM Platform 11.00 is installed on your server machine.

For more information on installing ALM Platform, refer to the *HP Application Lifecycle Management Installation Guide*.

2. Install *Application Lifecycle Management 11.00 SP2* on your server machine.
 - a. Visit the HP Software Support Web site at <http://www.hp.com/go/hpsupport>.
 - b. Under *Where do I find*, click *Software patches* and download the patch.

- c. Follow the on-screen installation instructions.
3. Deploy the ALI extension on your server machine. For details, see "[Deploying the ALI Extension](#)" (on page 12) for details
4. Enable the ALI extension in ALM Site Administration for every project that requires ALI.
For details, see "[Enabling the ALI Extension](#)" (on page 12)

Deploying the ALI Extension

To deploy the ALI extension:

Download the `ALI_EXTENSION.qcx` file from the add-in page.

On the ALM Platform machine, run the HP ALM Extension Deployment Wizard, select the `ALI_EXTENSION.qcx` file and follow through the wizard. The deployment tool validates extension (`.qcx`) files and updates archive (`.war`) files that can then be deployed to the application server. Once deployed, changes to the archive files take effect.

If using a JBoss application server:

- The tool automatically deploys the archive files (ALM and extension `.war` files).
- By default, the tool starts JBoss when complete.

When using another application server, deploy ALM manually after the tool is finished. The tool displays the location of the archive files (by default the `HP/ALM/deployment/qcbin.war` folder). Restart ALM after deploying these files.

For more details on running the Extension Deployment Tool, refer to the *HP Application Lifecycle Management Installation Guide* and *HP ALM Extension Deployment Tool Readme*.

Enabling the ALI Extension

After deploying ALI on the ALM Platform, the extension must be enabled. It may be enabled for either a new project or an existing one.

Before enabling the ALI extension, consider the following points:

- You cannot disable an extension for a project after you enable it.
- If the project is active, you must deactivate it before enabling an extension. After the extension has been enabled, you can reactivate the project.
- Enabling extensions for a project can take some time.

To enable the ALI extension for a project:

1. In Site Administration, click the *Site Projects* tab.
2. If relevant, create a new project in ALM in which to enable the extension. For details on creating ALM projects, refer to the *HP Application Lifecycle Management Administrator Guide*.
3. In the Projects list, select a project. In the right pane, click the *Project Extensions* tab. The Extensions list is displayed, listing extensions installed on the ALM Platform for which you have a license.
4. For the *ALI* extension, select the *Enable* checkbox. Click *Yes* to confirm

Upgrading Projects

If you worked with ALM projects on previous versions of the ALI extension, you must upgrade those projects to work with the current version.

To upgrade a project:

1. It is strongly recommended that you backup your original project before the upgrade.
2. Enter the Site Administration and click the **Site Projects** tab.
3. In the **Projects** list, select a project.
4. Click the **Expand the Maintain Project** button and select **Upgrade Project**.
5. Read the guidelines in the wizard and click **Upgrade Project**. The Upgrade Project window displays the results.

If using Perforce, you will need to reload your data. For details, see ["Perforce Migration" \(on page 13\)](#).

For more information on upgrading projects, refer to the *Upgrading Projects* section in the *HP Application Lifecycle Management Administrator Guide*.

Undeploying the ALI Extension

To undeploy ALI extensions:

On the ALM Platform machine, run the *HP ALM Extension Undeployment Wizard*, and select the extensions to undeploy.

For details on running the Extension Undeployment Wizard, refer to the *HP ALM Extension Deployment Tool Readme*.

Perforce Migration

If upgrading from Application Lifecycle Intelligence 1.1 to version 2.0, the migration of Perforce data handled by ALM requires removing the current data and reloading them from Perforce.

To migrate all data from Perforce SCM perform the following steps on each Perforce repository:

1. From the Management module, select **SCM Repositories** and choose the Perforce repository.
2. From the Repositories menu select **Cleanup** and select a date that removes the most recent commit.
3. Go to the Branches tab and for each branch:
 - a. Select branch details.
 - b. Delete the **Last Change Read** value.
4. From the Change Detection tab start the synchronization.

Chapter 3

SCM System Integration

This chapter includes:

["Supported SCM Systems" \(on page 14\)](#)

["Configuring Repositories" \(on page 15\)](#)

["Setting Branches and Enforcement" \(on page 17\)](#)

["Setting Change Detection" \(on page 21\)](#)

Supported SCM Systems

Application Lifecycle Intelligence (ALI) supports Subversion, Perforce and CVS systems natively and no additional installations are required to load change sets from these systems unless commit message policy enforcement with a push mechanism is required. If push mechanisms are required, SCM agents need to be installed on the SCM System. For details, see ["SCM Agents" \(on page 14\)](#).

TFS systems are supported via the SvnBridge library that allows you to use Subversion clients with Team Foundation Server. It converts the calls made by your Subversion client to the API supported by TFS.

Supported SCM Systems

ALI provides support for following SCM systems:

- Subversion Versions: 1.5.*, 1.6.* (Tested on 1.5.5, 1.5.6, 1.6.1, 1.6.15)
- CVS Versions: 1.11.*, 1.12.* (Tested on 1.11.22, 1.12.13)
- TFS Versions: Team Foundation Server 2010 (Tested on TFS 2010)
- Perforce Version 2010.2 (Tested on 2010.2)

The SCM agents support deployment to the following operating systems:

- Red Hat Enterprise Linux 5.x (32bit, 64bit)
- SuSE Linux Enterprise 11.x (32bit, 64bit)
- Windows 2003/2008 Server (32bit, 64bit)

Prerequisites:

MS PowerShell must be installed on Windows and enabled to run scripts. Linux/Unix agent scripts use BASH.

SCM Agents

The agent is a name for applications related to appropriate SCM systems required for PUSH mechanisms. The agent is a set of scripts or proprietary applications installed on a SCM server configured for listening on a SCM system. When a change is committed to a configured repository

and branch, the agent checks the policies and pushes the change set to the ALM server if the commit is allowed.

Instructions on how to configure SCM agents can be found in the `readme.txt` file found in the specific agent archive inside the main ALI distribution archive.

SVN Agent for Linux

```
agents\scm-integration\unix-linux\scm-agent-subversion.tgz
```

SVN Agent for Windows

```
agents\scm-integration\windows\scm-agent-subversion.zip
```

CVS_Linux

```
agents\scm-integration\unix-linux\scm-agent-cvs.tgz
```

CVS_Windows

```
agents\scm-integration\windows\scm-agent-cvs.zip
```

TFS Agents Installation Process

```
agents\scm-integration\windows\scm-agent-tfs.zip
```

Perforce Agent Linux

```
agents\scm-integration\unix-linux\scm-agent-perforce.tgz
```

Perforce Agent Windows

```
agents\scm-integration\windows\scm-agent-perforce.zip
```

For details, see ["Setting Change Detection" \(on page 21\)](#)

Configuring Repositories

To enable Application Lifecycle Intelligence (ALI) to load code changes from SCM systems and to start the automated traceability of work items (requirements/defects), code changes and defined releases, SCM repositories must be configured.

This chapter includes:

- ["Adding Repositories" \(on page 15\)](#)
- ["Setting External Repository Viewer" \(on page 16\)](#)

Adding Repositories

ALI supports Subversion, Perforce, CVS and TFS over SVN bridge. For details, see ["Supported SCM Systems" \(on page 14\)](#).

To add a repository:

1. Log in to ALM with Administrator privileges and under the Management tab, click **SCM Repositories** to open the Repositories page.
2. Click **New Repository**, and from the drop down list select a SCM type and click **OK**
3. In the new window, enter a name for the repository, a location and a username and password if

required.

Note: The location is a full path to the repository such as `http://host/svn/repo` etc..
Check with your system administrator if unsure what this value should be.

Perforce: ALI repositories are equivalent to Depots. The location of the repository should consist of a host name and port of the Perforce Server and Depot name. The format should be similar to `host:port//depot_name`.

Subversion: The repository location must point to the actual repository root. Branches are later used for specifying paths within the repository. In the case that an SVN URL has an unknown root, use the "`svn info <URL>`" command to find the root.

TFS over SVN Bridge: Fill the alias property with the direct TFS address.

CVS: Fill the alias property with whole `CVSROOT` (e.g. `:pserver:username:password@host/cvsrepo`) exactly as it is configured in your build system.

4. If any of the repository properties require modification, select a property and click the **Edit Property** button to open the **Edit Property** window and add a value. Click **OK** after entering the required value.

Note: CVS repositories require the CVS protocols `pserver` and `initial date` to start loading change sets. The CVS root should also be specified, but it is not mandatory.

5. Click **Submit**. ALI tests the connection to the repository before adding it to the list.

Change detection and commit pattern options may be set when adding the repository.

For details, see ["Setting Change Detection" \(on page 21\)](#) and ["Setting Commit Patterns" \(on page 19\)](#).

Setting External Repository Viewer

ALI provides a built-in repository viewer for viewing files' diffs and details with no additional configuration required. You can also use a preferred repository browser such as ViewVC.

The following properties are entered when configuring an SCM repository or can be changed from the details tab of an existing repository.

Template for diff links - template for a link pointing to a diff view of a given file in a repository viewing system (e.g. ViewVC). Allows you to create links from the ALI changeset table in the UI. Each file in the changeset contains a link displaying a diff view (revision in current changeset to its previous revision). The template may contain tags/variables which are expanded at runtime from the context of the currently selected changeset in the UI.

Tags/Variables that may be substituted/expanded:

`${filePath}` ... path of file within repository.

`${revision}` ... revision in current changeset.

`${fromRevision}` ... previous revision to revision in current changeset.

`${fromFilePath}` ... in case of copying/moving its source location.

Template for file links - template for a link pointing to a file view of a given file in a repository viewing system (e.g. ViewVC). Allows you to create links from the ALI changeset table in the UI.

Each file in the changeset contains a link displaying file view (file text content of a given revision in the scope of current changeset). The template may contain tags/variables which are expanded at runtime from the context of the currently selected changeset in the UI.

Tags/Variables that may be substituted/expanded:

`${filePath}` ... path of file within repository.

`${revision}` ... revision in current changeset.

Setting Branches and Enforcement

Branches are set from the **Branches** tab of any existing repository. A Branch is a separate line of a repository that exists independently of other lines.

A branch needs to be associated with a release and then change detection options can be set to make the changes visible in the Code Changes table in the form of a Change Set.

Commit Patterns may also be set in order to allow or disallow users from committing changes to the branch unless meeting specified criteria.

Policy enforcement is in place when there is an agent configured for the SCM repository.

For more information about agents, see ["SCM Agents" \(on page 14\)](#).

This chapter includes:

["Setting Branches" \(on page 17\)](#)

["Setting Change Detection" \(on page 21\)](#)

["Setting Commit Patterns" \(on page 19\)](#)

Setting Branches

To add a new branch:

1. Go to the **Branches** tab and click **add**.
2. Provide a path name for the branch and optionally a *Branch* and a *Last Change Read* property.

Note: A Branch name only has meaning for some CVS repositories. Do not fill this property for SVN or TFS over SVN bridge. The Last Change Read field is either the last revision in the case of SVN or the date-time for CVS of which the change sets are read. Leave these fields empty if reading all change sets from the given branch is required.

Note: If using Perforce, set the branch path without the Depot name. For example, if the branch is located at `//depot/HelloWorld/releases/release-1.0/...` then the path should be `/HelloWorld/releases/release-1.0`. Do not set the parameter `branch` even though the branch is named.

3. Click **Submit** to test the link and add the branch.

To associate a branch with a release:

1. Open the SCM Branch Details window by either clicking on the path of the branch or the **Details** button.
2. Click the **Add** button and check the radio button for the required release in the drop down menu

3. By default, the start date and end date is taken from the release and these can be changed by clicking in the date field and typing in a required date.

Change sets from the branch in the given time period are associated with the specified release. These change sets are visible in the Code Changes module when selecting the specified release. More releases can be specified for the branch but typically in a different time period.

To specify check-in policies for the branch where you want to enforce commits:

1. Click **Enforcement** in the left hand pane of the **SCM Branch Details** window.
2. Select from the following options:
 - **Commit message must match defined pattern** on the *Check-in Policies* tab causes commits for which the commit message doesn't match the predefined pattern to be refused by the agent.
 - **Change Set refers to a Requirement** enforces that every commit must refer to an existing requirement.
 - **Has Requirement Type** or **Has Priority** and picking the types and/or priorities from the drop down menus so commits allow only changes associated with a specific requirement type or priority.
 - **Change Set refers to a Defect** enforces that every commit must refer to an existing defect.
 - **With severity of** and selecting the severity levels required allows only changes associated with a specific severity.
 - **Add this note to the System message** sends custom system messages that informs a user when a commit is blocked with the applied note.
3. Click **OK**.

Note: The check-in policy feature requires the installation of an agent in order to function. For details, see ["SCM Agents" \(on page 14\)](#).

To specify locking policies for the branch where you want to enforce commits:

1. On the *Locking Policies* tab you can manage locking policies associated with the branch.
2. Select from the following options:
 - **Disallow commits except for the following** disallows all the commits coming to the given branch other than the following exceptions:
 - A list of users allowed to commit to the branch can be specified though the branch is locked by applying the user name(s) of SCM users who are permitted to commit.
 - A list of defects specified for which committing changes are allowed is created by clicking on the **Add defect** button and providing the id of the defect. Defects are removed from the table by clicking **Remove Defect**.
 - **Add this note to the System message** sends custom system messages that informs a user when a commit is blocked with the applied note.
3. Click **OK**.

Setting Commit Patterns

Detecting and maintaining traceability between change sets and requirements/defects is based on a commit message provided by a user (developer) while committing a change to the SCM system. ALI provides customizable mechanisms that accommodate various allowance patterns provided by the user. The purpose is to transfer requirement or defect id's within the commit message provided during the committal of a change set.

Commit Pattern settings can be accessed while setting up a new repository in the **Commit Pattern** view, or by clicking on the **Commit Pattern** tab of an existing repository.

Patterns can be defined in either *Basic* or *Advanced* mode.

Basic:

1. Add keywords for Defects or Requirements by clicking the **Add** button, entering a value and clicking **OK**. Provided keywords mark a commit as related to either a defect or a requirement.
Keywords may be deleted by selected the word and clicking the **Delete** button.
2. Add Prefix IDs to the pattern by typing in the ID number. Clicking the radio button makes them optional.
3. Click the **More Options** button add the following options:
 - **Include Default Tasktop Commit Pattern** – switched on by default. When Tasktop plugin is integrated with ALM, Tasktop generates default commit messages according to a pattern that is recognized by ALI.
 - **Case Sensitivity Commit Messages** – switched off by default. Switch on when you want to enforce case sensitivity on commit messages.
 - **Multiple Defects or Requirements separation markup** – this markup is recognized as a separator in the case of a commit message containing multiple defects or requirements.
 - **Keyword location in commit message** – keyword will be recognized only at the beginning of the commit message or anywhere within the message.
 - **User commit message separator** – separator between user commit message followed by defects or requirements keywords.

■ **Example of options in a commit message:**

fixing defect #100, #101: fix caching and enhance functionality

fixing defect = defect keyword

#100 = defect ID prefix and defect ID

“.” = multiple defects separator

“:” = user commit message separator

fix caching and enhance functionality = user commit message

4. Navigate away from the tab to commit the configuration.

Advanced:

1. Add to or change the default code pattern.
2. Test the modified code by clicking **Test Against Existing Commits**.
 - a. **Case Sensitive** enforces case sensitivity for messages.
3. Optionally test a Custom Commit Message by typing it into the field.
4. Navigate away from the tab to commit the configuration.

The **Restore Defaults** button removes any changes and replaces the default keywords.

Several basic examples of advanced configurations:

1. Example 1

a. Pattern

```
([fixing] REGEX('defects?') IDLIST(DEFECT) | [implementing] REGEX('requirements?') IDLIST(REQ)): TEXT
```

b. Example message:

```
"fixing defect #56721: something really serious was fixed"  
"defects #57893,#61432: division by zero"  
"requirement #1: domains"
```

2. Example 2

a. Pattern

```
(UNTIL(RE '((BUG))(REQ))#) (IDLIST(DEFECT lead='((BUG)?#)?' sep=',') | IDLIST(REQ lead='((REQ)?#)?' sep=',')){0,}[TEXT]
```

b. Example message:

```
"This commit fixes BUG#1,#2 and implements REQ#4,REQ#5 making the product faster (resolving BUG#7)."
```

- c. This pattern matches all inputs and extracts of any found 'BUG#' and 'REQ#' patterns. Such an open pattern may not be suitable for enforcing common policy, but it can be useful when data from legacy repositories are loaded in the "read-only" mode, e.g. for reporting purposes.

3. Example 3

a. Tasktop Pattern

```
(LISTITEM('Bug Status') - WORD IDLIST(DEFECT lead='DEF' sep="") | Incomplete - WORD IDLIST(REQ lead='REQ' sep="")): TEXT
```

b. Matches default Tasktop messages as:

```
"OPEN - task DEF10: http://host:9090/qcbin;DEFAULT;ALI_DEV-DEF10"  
"Incomplete - task REQ42: http://host:9090/qcbin;DEFAULT;ALI_DEV-REQ42"
```

Setting Change Detection

ALI detects changes made on preconfigured SCM repositories, loads information about change sets and associated files to the ALM server and automatically creates traceability between loaded code changes and work items (requirements, defects). ALI supports two detection mechanisms; the polling mechanism and the push mechanism.

When the polling mechanism is switched on, ALI periodically checks for new changes in the given SCM repository and loads them if they exist. For the polling mechanism, no configuration is required on the SCM system side so this method operates without running SCM agents. The polling mechanism corresponds to the option *Read changes from SCM* on the Change Detection tab.

The push mechanism connects SCM agents to the SCM system. The agent listens on the SCM system and when a change is committed to the preconfigured repository and branch, the agent checks the policies and pushes the change set to the ALM server if the commit is allowed. The push mechanism corresponds to the option *Receive changes transmitted by SCM agent(s)* on the Change Detection tab.

Change Detection settings can be accessed while setting up a new repository in the **Change Detection** view, or by clicking on the **Change Detection** tab of an existing repository.

There are two detection options available:

Read changes from SCM

- This option allows an interval for reading to be setup in periods of seconds as well as setting up a running schedule by clicking the **Reschedule** button and selecting a time and date.
- Immediate readings can be forced by clicking the **Synchronize** button.

Note: Synchronize process can be monitored using the **Task Manager** tool of ALM found under the Tools menu.

Receive changes transmitted by SCM Agent(s)

- If SCM agents have been setup, this option processes changes immediately. For details, see ["Supported SCM Systems" \(on page 14\)](#)

Chapter 4

Build System Integration

Builds are the key deliverables of software development. Application Lifecycle Intelligence tracks information about builds together with their relationships to other ALM entities in order to provide traceability.

The model provides information about:

- Build artifacts (binaries produced by the build process).
- Content of the build (components, packages, files ...).
- New changes included in the build (what's new since the last build – change sets that make up the build).
- Implemented work items (requirements, defects, stories that have been implemented/modified in the build).
- Relationships between Builds and Releases.
- Relationships between Builds and Tests.

Integration with build systems is a tool that allows you to measure the impact of code changes on software deliverables and build artifacts. Builds allow you to report what new code has been implemented and what the impact is on the delivered project with the delta of key metrics (e.g. a certain commit caused a 5% degradation of test results).

This chapter includes:

["Adding a Build Server" \(on page 24\)](#)

["Adding a Build Server" \(on page 24\)](#)

["Adding Build Configurations" \(on page 25\)](#)

["Reusing SCM Configurations from Build Configurations" \(on page 25\)](#)

["Setting Build Configuration Defect Filters" \(on page 26\)](#)

["Customizing ALI Project Lists" \(on page 26\)](#)

["Setting Build Server Detection" \(on page 27\)](#)

Supported Build Systems

ALI provides support for following build systems:

- Jenkins - Long-Term Support release - 1.409.2
- Hudson - Latest Production Version - 2.1.2

ALI provides plug-ins for build servers that automatically extract build information and metrics and embed them into the ALM model.

ALI supports following development metric tools:

- Unit Testing
 - JUnit – Tested on versions bundled with supported build systems.
 - TestNG - Tested on Hudson 0.8, Jenkins 0.28
 - NUnit - Tested on Hudson 0.10, Jenkins 0.14 + version of NUnit framework 2.5.10
- Code Coverage Analysis
 - Cobertura - Tested on Hudson 1.1, Jenkins 1.3
 - Ncover - Tested on Hudson 0.3, Jenkins 0.3 + version of NCover 3.4.18.6937 x86 (trial)

Hudson/Jenkins Installation

It is necessary to install:

- Hudson or Jenkins (download from <http://hudson-ci.org> or <http://jenkins-ci.org>).
 - Hudson/Jenkins plugin supporting use of SCM (SVN and CVS are supported by default. Perforce and TFS require the installation of special plugins available in the public Hudson/Jenkins plugin repository).
 - HP ALI Hudson/Jenkins plugin (`ali-hudson-plugin.hpi` located in the ALI bundle - `agents\build-integration\hudson\`). This plugin supports both Hudson and Jenkins systems.

HP ALI Hudson/Jenkins Plugin Installation

To install the HP ALI Hudson/Jenkins Plugin:

1. Go to the **Hudson/Jenkins->Plugin Manager** and open the **Advanced** tab.
2. In the **Upload Plugin** section browse to `ali-hudson-plugin.hpi` (part of the ALI bundle) and click **upload**.
 - If the project source code built by Hudson/Jenkins is stored in TFS, you have to install the `ali-hudson-tfs-plugin.hpi` as well as the base `ali-hudson-plugin.hpi`.
Caution: `ali-hudson-tfs-plugin.hpi` requires pre-installation of Hudson/Jenkins TFS plugin.
 - If the project source code building by Hudson/Jenkins is stored in Perforce, you have to install the `ali-hudson-perforce-plugin.hpi` as well as the base `ali-hudson-plugin.hpi`.
Caution: `ali-hudson-perforce-plugin.hpi` requires pre-installation of Hudson/Jenkins Perforce plugin.
3. After uploading the plugins, restart the Hudson/Jenkins server to enforce changes.

Note: The installed plugins should be listed in the **Installed** tab in the **Plugin Manager**.

Note: When the ALI plugin is installed properly, the **Ali Integration** link should be visible in the Hudson/Jenkins left side menu. Clicking this link displays capabilities provided by the plugin.

For more details on how to install and work with the Hudson/Jenkins plugins, please see the Hudson/Jenkins system documentation.

HP ALI Hudson/Jenkins Plugin Configuration

The ALI Hudson plugin has a global configuration accessible from the global Configure System and the job scope configuration accessible from a specific job. For a detailed description please go to the ALI integration plugin on the Hudson/Jenkins server, where every property is described.

To set the global configuration:

From **Manage Hudson/Jenkins – Configure System** and select the **ALI Integration** section. You can configure the following options:

- *Include the credentials in the SCM configuration* - Specify if the username and password should be included in the SCM repository descriptor. If this security model is enabled, the user must also have "extended read" permissions for the credentials to be listed.

Caution: Be very careful when enabling this option as the credentials for the SCM repository associated with build configurations are exposed on REST endpoints as plain text.

- *Update build information in HP ALM* - When selected, information about performed builds is sent to the HP ALM server immediately after a build is started and then again when it is finished.

Note: All properties specified in the global configuration can be later overrode for a particular job. These properties are required when you want to configure the pushing mechanism for the given build server, but if you want to use the push mechanism and *also* want to specify ALM properties for individual jobs, this option must be enabled.

To set a configuration for a particular job:

From the Configure link on left side menu for the given job, select the **ALI Integration** option. This option must be switched on if you want to have ALI integration enabled for the job.

- *Test sources mapping pattern* - Allows you to determine test source locations based on actual test results. For details, including an example, see the ALI integration plugin help.
- *HP ALM configuration* – Overwrite the global ALI configuration properties - Update build information in HP ALM for the given build job.
- *NCover* - Related to NCover code coverage for .NET configurations. The *NCover report XMLs* specifies the generated raw XML report files, such as `myproject/target/coverage-reports/*.xml`. Basedir of the fileset is the root workspace.
- *Force.com* – relating to Force.com integration. For details, see "[Force.com Integration](#)" (on page 28).

Adding a Build Server

To add a Build Server:

1. Log in to ALM with Administrator privileges and under the Management tab click **Build Servers** to open the Servers page.
2. Click **New Server**, and from the drop down list select a Build Server Configuration and click **OK**

3. In the New Build Server dialog box, enter a name, a location of the server, a server description and a username and password if required.

Note: The location is a full path to the server such as `http://xx.xx.xxx.xxx:xxxx`. Check with your system administrator if unsure what this value should be.

4. Select the **Change Detection** tab and optionally:
 - Select **Read changes from build server** and provide an interval or schedule run times if required.
 - Select **Receive builds transmitted by build server agent(s)** if required. See Build Server Detection for more information.
5. Click **Submit** to add the server.

Adding Build Configurations

To add a Build Configuration:

1. Go to the **Build Configurations** tab and click **add**.
2. Select a Build Configuration from the presented list and click **Ok** to open the details window.

Note: This list contains the unused configurations of the governed server.

Caution: This list contains only the configurations for Hudson/Jenkins jobs with ALI integration enabled.

3. Click the **Release** drop down menu and select an associated release.
4. Click the **Build Configuration** drop down menu and select a build type. This value allows you to filter builds in the Builds module under the Development tab.
5. Optionally:
 - Enable the configuration - If the build configuration is enabled, new builds from the build system are loaded.
 - Set the configuration as default - Default build configurations are used for computation of statistics displayed for the associated release.

Reusing SCM Configurations from Build Configurations

Existing build configurations are stored by ALM and the SCM branches and repositories can be applied to newly created configurations.

For a particular build configuration on a build system, SCM related information is typically already specified (SCM repository, branch, credentials). ALI provides the possibility to reuse this information from a build system for a particular build configuration to simplify the configuration of another SCM system within ALI.

To apply an existing SCM configuration from a particular build configuration:

1. Click on a server in the **Build Servers** page of the Management module.
2. Select the **Build Configurations** tab.
3. Select a configuration and click it to bring up the **Build Configuration Details** popup window.

4. Select the **SCM** tab to display the available repositories and branches.
5. Select the required repositories and/or branches and click the + button to apply them to the SCM configuration.

The status message next to the repository or branch location provides information about whether the repository or branch has been previously defined or not.

Note: Clicking the details button will bring up the repository or branch details popup window of the selection.

Setting Build Configuration Defect Filters

Setting of this filter reduces input set of open and new defects associated with the release shown on the Build Report Defect Trend graph for this build configuration. Do not specify defect status and target release in this filter.

To apply a defect filter to a particular build configuration:

1. Click on a server in the **Build Servers** page of the Management module.
2. Select the **Build Configurations** tab.
3. Click on a configuration name to bring up the **Build Configuration Details** window.
4. Select **Defect Filter**.
5. Click **Change Defect Filter** to open the filter defect window.
6. Add or remove a filter condition and click **Ok** to apply the condition.

To remove a defect filter from a particular build configuration:

Click **Clear Defect Filter**, confirm the deletion by clicking **Yes**.

Customizing ALI Project Lists

The following Project lists may have their behavior changed in the customize module:

- **ALI Build Category**

The Build Category from the list on the Build module tool bar contains the values of the ALI Build Category project list

- **ALI Closed Status**

The project list contains all the values with a “*Closed*” status of defects. **Closed In Build** value can only be applied to defects with this status.

- **ALI QA Status**

The QA Status field on the Build entity contains the values from the ALI QA Status project list.

- **ALI Reported Severity**

The links to code issues visible in the drop down window of the Requirement/Defect Development Activity Tab are defined by the project list. The customizable values in the customize module must contain either full set or a subset from the Severity project list.

To customize the configurations list:

1. Select **Customization** from the Tools menu of ALM, .
2. Select the **Projects Lists** object from the left menu.
3. Select the particular project list.
4. Add new items, rename or delete items from the list.
5. Click **Return** to commit the changes and return to ALM.

Setting Build Server Detection

Application Lifecycle Intelligence detects new builds on build servers, loads new status information, associated code changes and optionally, the results of unit tests, and code coverage. Traceability is automatically created between builds, code changes and work items (defects, requirements). As with SCM integration, ALI supports two detection mechanisms; the polling mechanism and the push mechanism.

Change Detection settings can be accessed while setting up a new build server in the Change Detection view, or by clicking on the Change Detection Tab of an existing build server.

Polling:

With the polling mechanism option, ALI periodically checks for new builds in the given build server and loads them if they had not already existed on ALM server.

The polling mechanism is activated with the *Read changes from build server* option on the Change Detection tab. You can schedule runs, specify a time period to check for new builds or load changes on demand with the Synchronize command.

Push:

The push mechanism works in an opposing manner to the poll mechanism. When the *Update build information in HP ALM* option is selected in the ALI integration plugin on the build server (for details, see "[HP ALI Hudson/Jenkins Plugin Configuration](#)" (on page 24)), the plugin listens on the build server. When a new build is starting, it pushes information about the build to the ALM server. When the build completes, the status and other build relevant information are updated on the ALM server.

The push mechanism is activated by the *Receive builds option transmitted by build server agent(s)* option on the Change Detection tab.

Chapter 5

Force.com Integration

This integration allows teams developing for the Force.com platform to benefit from all the features that ALI brings to standard development. Although all the source code is stored, compiled and tested in the Cloud, ALI establishes traceability between code, work items (requirements and defects), builds and build metrics (test results and coverage).

Note: Force.com integration tested on Force.com version API 22.0

The integration between Force.com and ALI requires:

- Storing Force.com source code in an SCM system (for the complete list of SCM systems supported by HP ALI see "[Supported SCM Systems](#)" (on page 14))
- Hudson or Jenkins
 - Hudson/Jenkins plugin supporting use of SCM (SVN and CVS are supported by default).
 - HP ALI Hudson plugin (`ali-bundle.zip/build/Hudson/ali-hudson-plugin.hpi`).
For details, see "[HP ALI Hudson/Jenkins Plugin Installation](#)" (on page 23).
- Configuring a build management server to deploy source code to the integration/staging environment.
- Apache Ant (download from <http://ant.apache.org/>).
- HP force-deploy-task is located in the ALI archive (`/tools/force-deploy-task/force-deploy-task-bundle.zip`, unpack content of zip file to `ant_install_dir/lib`).

Project Deployment, Testing and Report Generation

Deployment of source codes, testing and the following report generation is ensured by the special Ant task *HP force-deploy-task*. In order to function, it is necessary to create the Ant build script "build.xml" (if it does not exist) in the root folder of a Force.com project similar to the following examples:

1. The following example deploys source code to a configured Force.com environment and runs all tests. Because all tests will be run, report should contains code coverage of whole project.

```
<project name="Sample usage of force-deploy-task" default="
deployAndTestAndReport " basedir=".">
    <target name="deployAndTestAndReport">
        <taskdef name="sfdeploy"
classname="com.claimvantage.force.ant.DeployWithXmlReportTask"/>
        <delete dir="test-report-xml" quiet="true"/>
        <sfdeploy
username="username to force.com environment"
password="password to force.com environment"
serverurl="force.com server URL"
deployRoot="path to source directory"
```

```

        runalltests="true"
        reportDir=" test-report-xml " />
    </target>
</project>

```

2. The following example deploys source code to a configured Force.com environment and runs only tests that match the given pattern. In this case, ALM will not be provided full code coverage.

```

<project name="Sample usage of force-deploy-task" default="
deployAndTestAndReport " basedir=".">
    <target name="deployAndTestAndReport">
        <taskdef name="sfdeploy"
classname="com.claimvantage.force.ant.DeployWithXmlReportTask"/>
        <delete dir="test-report-xml" quiet="true"/>
        <sfdeploy
username="username to force.com environment"
password="password to force.com environment"
serverurl="force.com server URL"
deployRoot="path to source directory"
runalltests="false"
reportDir=" test-report-xml ">
        <!-- Run only tests with file names that match this pattern -->
        <batchtest>
            <fileset dir="src/classes">
                <include name="*Test.cls"/>
            </fileset>
        </batchtest>
    </target>
</project>

```

Description of the HP force-deploy-task (in the example defined as sfdeploy):

- Attribute *username* – login name to force.com environment
- Attribute *password* – password to force.com environment
- Attribute *serverurl* – URL of login page to force.com environment
- Attribute *deployRoot* – path to source code directory (where folders classes, triggers etc. are located)
- Attribute *runalltests* – (true/false) if it is true all tests are started and code coverage of project is reported, if it is false only tests specified by batchtest element are started (code coverage is not provided)
- Attribute *reportDir* – directory where all reports will be stored
- Element *batchtest* – element specifies which tests should be started (works only if runalltests=true)

- Element *fileset* – file set of tests to run
 - Attribute *dir* – directory where tests are located
 - Element *include*
 - Attribute *name* – class name pattern of tests to run

Hudson/Jenkins Force.com Configuration

1. Create a Free style job and configure the SCM and Build Triggers as needed
2. Add the build step *Invoke Ant* and specify the targets which should be started (as in the example *deployAndTestAndReport*).
3. In the *Post-build Actions* section, configure as per the following image:

The screenshot shows the 'Post-build Actions' configuration in Hudson/Jenkins. It features two main sections:

- Publish JUnit test result report:** This section is checked. The 'Test report XMLs' field contains the pattern `test-report-xml/**/TEST-*.xml`. Below this, there is a link to the 'Fileset 'includes'' setting and a checkbox for 'Retain long standard output/error' which is unchecked.
- ALI Integration:** This section is also checked. The 'Test sources mapping pattern' field contains `.*//src/classes//cls`. A tooltip below explains that this is a 'Regex pattern for locating test sources, e.g. (.*)?target/surefire-reports/.*.xml//\$1/src/test/java'. An 'Advanced...' button is located at the bottom right of this section.

4. In the test report XMLs, replace the `test-report-xml` string with your actual report directory (attribute `reportDir` in `force-deploy-task`).
5. In the test sources mapping pattern, replace `src` with your actual path to the source directory.

The above configuration is sufficient for most of cases, but in case that:

- the `force-deploy-task` is defined in the distributed Ant script which is called from the main Ant script, configure the Report directory (value of attribute `reportDir` in Ant script) in Ali Integration/Advanced
- the source code directory (which contains folders `classes`, `triggers` etc.) is not in the `src` directory which is located directly in the workspace root, configure the Project root.

Configurations examples:

Force.com

Report directory	test-report-xml
	Relative path to directory where reports are generated by HP force-deploy-task Ant task (attribute <code>reportDir</code>). Consider workspace root as basedir.
Project root	src
	Relative path to directory with source codes (consider workspace root as basedir). Default value is <code>src</code> .

Chapter 6

Monitoring SCM Changes and Traceability

After setting up the repositories, branches and enforcement rules, Application Lifecycle Intelligence (ALI) aggregates the data and presents the Change Sets in the Code Changes tab of ALM. This tab provides information about what is happening within a project from the source code point of view and maintains traceability between the source code, requirements/defects and releases.

Reporting capabilities are an important part of effective project management and ALI extends the HP ALM platform reports with additional reports covering code changes entities.

You can use project reports and graphs (summary, progress & trend) to review and compare the development activity of your teams and developers. In addition to this, ALI provides a Change Impact Report that describes the impact of implemented changes on defects and requirements.

This chapter includes:

["Viewing the Code Change Table" \(on page 31\)](#)

["Code Change Details" \(on page 32\)](#)

["Viewing Change Impact Report" \(on page 32\)](#)

["Generating Project Reports" \(on page 33\)](#)

["Generating Graphs" \(on page 33\)](#)

Viewing the Code Change Table

The **Code Changes** tab is available in the development module.

Initial display presents all code changes in selected releases and a given time period. This can be modified and filtered using the drop down menus at the top of the table.

Date/Time	Developer	Files	Lines	Message	Status Message
5/26/2011 12:49:56 PM	bill	1	3	defect #37: use standard mechanism for asynchronicity	
5/26/2011 10:48:16 AM	john	1	16	defect #40: make name/summary clickable too	Defect 40 does not es
5/26/2011 10:36:01 AM	david	1	8	requirement #1: documentation update	
5/26/2011 8:48:00 AM	luke	2	27	defect #41: add missing readme	Defect 41 does not es
5/25/2011 6:37:00 PM	bill	1	2	Incomplete - task REQ2712: Portal Semantics http://	Requirement 2,712 d
5/25/2011 6:34:20 PM	bill	1	2	Incomplete - task REQ2709: Reporting DB Schema http://	Requirement 2,709 d
5/25/2011 6:31:14 PM	bill	1	8	Incomplete - task REQ2709: Reporting DB Schema http://	Requirement 2,709 d
5/25/2011 6:08:47 PM	john	4	26	defect #37: creation time of the report, minor layout modifications	
5/25/2011 5:14:53 PM	david	3	9	requirement #1: update of documentation	
5/25/2011 5:02:42 PM	robert	3	13	Requirement #2: Added missing resource.	
5/25/2011 4:54:28 PM	robert	1	1	fixing defect #41: column selection not persisted	Defect 41 does not es
5/25/2011 4:42:50 PM	luke	6	135	defect #38: enforcement and push support more projects	
5/25/2011 4:37:35 PM	luke	1	1	requirement #3: new documentation	

The table presents Date/Time of the change, developer who made the change, number of affected files and changed lines, and the commit message provided by a developer. The status message displays an error or warning when a commit message doesn't correspond to a predefined pattern. The columns in the table may be modified and filtered by enabling the filters under the **View** menu.


Black indicates there is an associated requirement for the change set.

Green indicates there is an associated defect for the change set.

Red indicates there is neither a defect nor a requirement associated for the change set.

In addition to release and time, data associated with only requirements or defects and unassigned changes can be displayed.


Note: An unassigned change is a commit which is related to neither requirement nor a defect. This may be valuable information for a manager of a team working on unplanned changes.

Advanced filtering can be accessed by clicking on the filter button .

Enabling the Information Panel under the **View** menu opens the panel under the table with tabs for viewing associated files, requirements and defects within the change sets. Commit messages or status messages can also be viewed in this panel.

Code Change Details

To view Code Change Details:

- Double click on a Code Change or click the Code Change Details button  to open the code change in a new window.
 - In addition to the basic attributes, you can see list of associated files together with their status and have the ability to compare differences to previous versions of the file.
 - If the change set is associated with a requirement or defect, then links are provided by clicking the Requirements or Defects menu items in the left side menu.

Viewing Change Impact Report

ALI reports the impact of changes not only in terms of files & modules but also work items assigned to developers (requirements & defects). It provides aggregated information about code coverage, test results, amount of changes and who committed impacted work items.

To view the Change Impact Report:

1. Select a time period and click on **View Report**.

A report based on the time period is presented in a new window.

The report displays the changes that have occurred to Requirements and Defects during the selected period.

A graph at the top right of the report displays the effort distribution of the project in KLOC (Kilo lines of code).

2. Click **Show Unassigned Changes** to display Unassigned changes.
3. Click **Print This Report** to send the report to your printing device.
4. Click **Email This Report** to open your email client and send the URL of the report to a provided address.

Generating Project Reports

The ALI extension provides a set of predefined report templates that are added to the standard ALM report selection.

Predefined templates are:

- ALI – Code Changes Template
- ALI – Defect Overview Template
- ALI – Requirement Overview Template
- ALI – Build Template

These templates allow you to create project reports related to Code changes and Builds.

For more information about the generation of project reports see the *HP Application Lifecycle Management User Guide*.

Generating Graphs

The ALI extension enables you to generate graphs for Code changes.

You can generate the following graphs for code changes:

Graph	Description
Code changes Progress Graph	Shows how many code changes accumulated in an ALM project at specific points during a period of time. Specify the time interval displayed along the x-axis, and the requirement information by which ALM groups the data. Specify whether you want to view the count of code changes, sum of changed files or sum of changed lines related to developer.
Code changes Summary Graph	Shows how many code changes are currently in an ALM project. Specify the type of data displayed along the x-axis, and the code change information by which ALM groups the data.
Code changes Trend Graph	Shows the history of changes to specific code changes fields in an ALM project, for each time interval displayed. Specify the field for which you want to view the number of changes and the time period for which you want to view data.

To Generate a Graph:

1. Launch the Graph Wizard.

The Graph Wizard guides you through the steps involved in creating a graph and configuring its settings. You launch the graph wizard either from the Analysis View module, or while working in the Code changes module.

a. In the Analysis View module.

On the ALM sidebar, under Dashboard, select **Analysis View**, right-click a folder, and select **Graph Wizard**.

b. In the Code changes module.

Select **Analysis > Graphs > Graph Wizard**.

c. In the Graph Wizard window change the entity name to Code Changes. There are three graph types listed. Select the appropriate graph and follow the wizard instructions.**2. Create a predefined graph.**

While working in the Code changes module, generate an impromptu graph to analyze the module data.

- Select Analysis > Graphs and select one of the predefined graphs.

You can choose from :

- Code Changes - Summary Graph - 'Group by Developer'
- Code Changes - Summary Graph - Sum of 'Files' Group by Developer'
- Code Changes - Summary Graph - Sum of 'Lines' Group by Developer'
- Code Changes - Progress Graph - 'Group by Developer'
- Code Changes - Trend Graph - 'Group by Developer'

3. Create a graph in Analysis View.**a. Open the Analysis View module. On the ALM sidebar, under Dashboard, select **Analysis View**.****b. Add a folder to the analysis tree. Right-click a folder under the Private or Public root folder, and select **New Folder**.****c. Create a new graph. Right-click a folder, and select **New Graph**. Change entity to **Code Changes**, select required graph type and fill in the graph name in the **New Graph** dialog box.****d. Click on the **View Tab** in the right panel.**

For more information about the generation of graphs see the *HP Application Lifecycle Management User Guide*.

Chapter 7

Monitoring Development Activity

After setting up build servers and configurations, Application Lifecycle Intelligence (ALI) presents Development Activity as tabs in the Releases, Defects and Requirements modules of ALM. These tabs present information about what is happening within a project from the build and developer point of view.

This chapter includes:

["Development Activity Releases" \(on page 35\)](#)

["Development Activity Defects" \(on page 35\)](#)

["Development Activity Requirements" \(on page 36\)](#)

["Monitoring Development Activity in Defect Details" \(on page 36\)](#)

The links within the tabs provide detailed information and statistics for the selected build.

The table at the bottom of the tab presents changes in the build as per developer.

Click **Change** to see the activity of other configured builds.

Development Activity Releases

If there is development activity associated with a release, a green star icon will be visible on the tab. Click the tab to view the activity.

Build Status with success rates, build times, and fix times as well as Codebase Statistics in KLOC (Kilo lines of code) are presented in the tab giving the user high visibility of how the build is progressing. Click the links in the tab to view details.

Click **Change** to select a different build configuration other than the default to view.

Click **Configure Builds** to change the configuration.

For details, see ["Adding Build Configurations" \(on page 25\)](#).

The table at the bottom of the tab presents a list of developers associated with the release and the contribution level of each developer.

Development Activity Defects

If there is development activity associated with a defect, a green star icon will be visible on the tab. Click the tab to view the activity.

The tab presents Development Metrics showing unit tests and coverage and Defect Implementations for the requirement. Click the Unit Tests link to see the **Defect Unit Tests Report** and the Coverage link to see the **Defect Coverage Report**.

Click the links under Defect Implementation to see build details or specific code changes.

The table at the bottom of the tab presents Code Changes and Active Developers associated with the defect.

Click the message link in the code change to be taken to the Defect details page.

Click on the column headers to sort or filter the tables.

Development Activity Requirements

If there is development activity associated with a requirement, a green star icon will be visible on the tab. Click the tab to view the activity.

The tab presents Development Metrics and Requirement Implementations for the requirement. Click the link Unit Test link to see the **Requirements Unit Tests Report** and the Coverage link to see the **Requirement Coverage Report**.

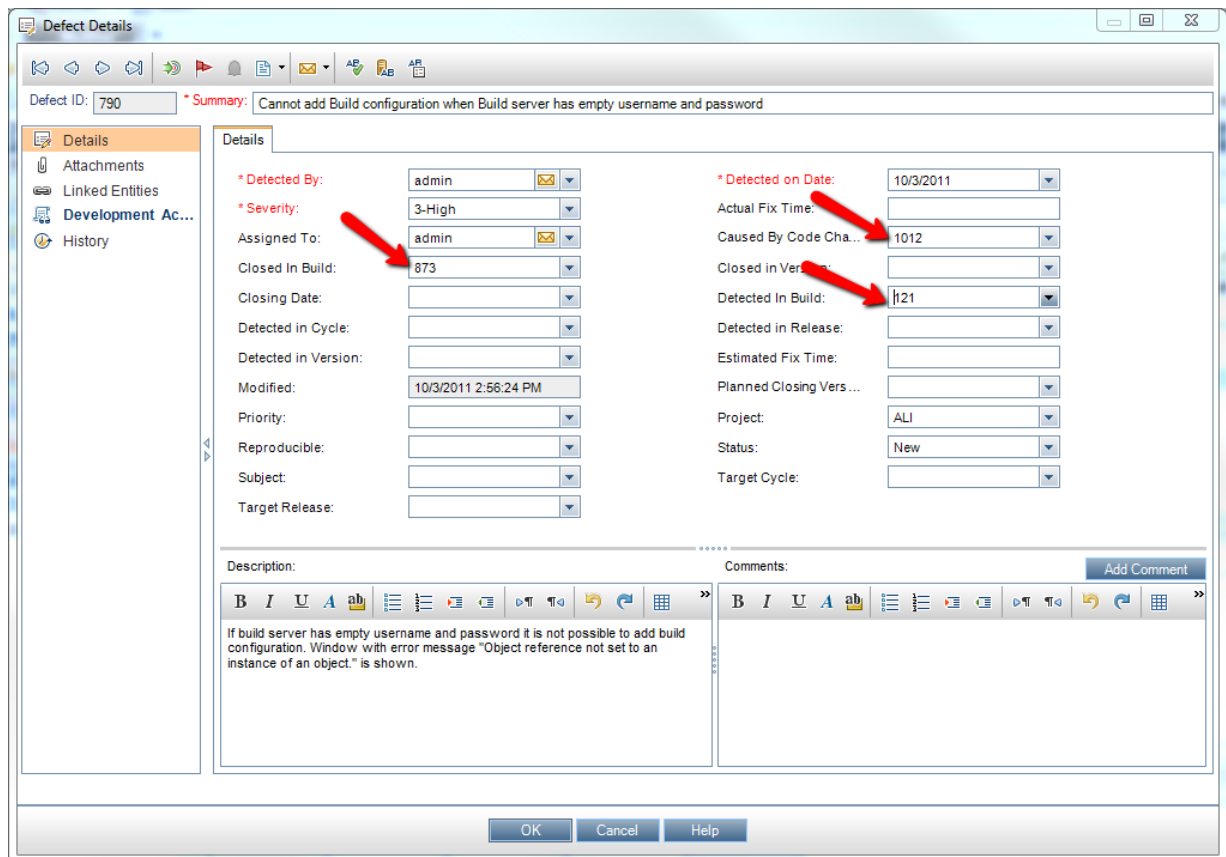
The table at the bottom of the tab presents the specific Code Changes and Active Developers associated with the requirement.

Click the message link in the code change to be taken to the Code Change details page.

Click on the column headers to sort or filter the tables.

Monitoring Development Activity in Defect Details

Application Lifecycle Intelligence introduces three new values to the Defect Details page.



Detected in Build – states on which build the given defect has been detected.

Closed in Build - states on which build the given defect has been closed.

Caused by Code change – states which change set caused this defect.

If a static code analysis tool supporting integration with HP ALM Platform has been configured, these values will be automatically filled. If not, they can be filled manually using the drop down list.

Data from these fields populate the defects tab the Build Details page and are represented in the Build Summary Report. For details, see ["Defects Tab" \(on page 39\)](#) and ["Build Reports" \(on page 40\)](#).

Chapter 8

Monitoring Build Activity

After setting up build systems, Application Lifecycle Intelligence monitors build results together with related code and implemented working items.

In the Builds module, you can see the status and results of your Fast, Nightly or Integration builds and take action in the case of reported errors. Depending on build system configuration, you can monitor the results of unit tests, the number of successful or failed tests, code coverage, source code and defect statistics.

This chapter includes:

["Viewing Builds" \(on page 38\)](#)

["Build Details" \(on page 39\)](#)

["Build Reports" \(on page 40\)](#)

["Generating Build Graphs" \(on page 40\)](#)

Additionally, specific project reports and graphs for Builds are available. For details, see ["Build Reports" \(on page 40\)](#) and ["Generating Build Graphs" \(on page 40\)](#)

Viewing Builds

When logging on to a project that has the ALI extension enabled, the **Builds** tab is available.

The initial display presents all builds in a selected release(es) and a given time period. This can be modified and filtered using the drop down menus at the top of the table.


Date/Time	Build ID	Number	Revision	Build...	Build Status	Label	QA Status	Unit Test
[ThisWeek]								
10/5/2011 4:55:18 PM	196	361		Test				
10/5/2011 3:15:07 PM	191	360	117168	Test	Success			100%
10/5/2011 2:25:16 PM	186	359	117161	Test	Warning			99%
10/5/2011 1:32:07 PM	178	358	117154	Test	Success			100%
10/5/2011 12:45:17 PM	175	357	117152	Test	Success			100%
10/5/2011 11:39:27 AM	173	356	117142	Test	Success			100%
10/5/2011 10:45:17 AM	172	355	117140	Test	Warning			78%
10/5/2011 9:18:10 AM	169	354	117137	Test	Warning			99%
10/5/2011 8:30:17 AM	167	353	117136	Test	Warning			77%
10/4/2011 7:00:17 PM	164	352	117135	Test	Warning			99%
10/4/2011 5:44:18 PM	162	351	117134	Test	Warning			99%
10/4/2011 4:48:29 PM	160	350	117130	Test	Success			100%
10/4/2011 4:00:18 PM	154	349	117127	Test	Warning			78%

The table presents Date/Time of the build, and all other information recorded for the build. The columns in the table may be modified and filtered by enabling the filters under the **View** menu.

Green indicates a successful build.

Yellow indicates a problem with the build and a warning.


Red indicates the build failed.

Advanced filtering can be accessed by clicking on the filter button .

Enabling the Information Panel under the **View** menu opens the panel under the table with tabs for viewing requirements, defects, code changes, distribution and status messages of the build.

Build Details

To view Build Details:

Double click on a Build or click the Build Details button  to open the build details in a new window.

In addition to basic information about the build, selecting a tab provides specific information about the build.

Available Tabs:

- ["Requirements Tab" \(on page 39\)](#)
- ["Defects Tab" \(on page 39\)](#)
- ["Code Changes Tab" \(on page 40\)](#)
- ["Active Developers Tab" \(on page 40\)](#)

Select a status from the QA Status drop down list and click **OK** to apply that status to the build.

Click **View Report** to generate a Build Change Report. For details, see ["Build Reports" \(on page 40\)](#).

Click **Build System** to generate a view of the entire Build System.

Requirements Tab

Provides a list of requirements related to the build.

Click the column header to sort the list.

Click on the requirement name to be taken to the requirement details page in ALM.

Defects Tab

Provides a list of defects associated with a build.

Detected Defects, Closed Defects and Delivered Fixes can be viewed by selecting the associated tab. A green star icon indicates a new entry in a list.

Click the column header to sort the list.

Click the link in the defect summary to be taken to the defect detail page in ALM.

Code Changes Tab

Provides a list of code changes associated with the build.

Click the column header to sort the list.

Click on the code change message to be taken to the code change details page in ALM.

Active Developers Tab

Presents a list of developers associated with the build along with level of contribution.

Click the column header to sort the list.

Build Reports

The **Build Summary Report** provides an overview view of builds from a given time period and build category. The graph combines information from SCM activity and open defects from the selected release.

To view a Build Summary Report:

1. Select Builds from the Development module of Application Lifecycle Intelligence.
2. Click the **View Report** button to generate the summary report in a new window.

The **Specific Build Report** displays the requirements, delivered defects, new defects and closed defects associated with that build.

To view a Specific Build Report:

1. Click on the link for that build in the summary report, or open the build detail page and click the **View Report** button.
2. Click the **Show Unassigned Changes** link to display unassigned changes.

The **Build Change Report** provides an overview of requirements affected by the selected build and defects that have been altered, closed or created by the selected build.

To view a Build Change Report:

1. From the **Builds** page of the **Development** module, select a build and open the **Build Details** page.
2. Click **View Report** to generate the change report in a new window.

Generating Build Graphs

The ALI extension enables you to generate graphs for Builds.

You can generate the following graphs:

Graph	Description
Builds Summary	Shows how many builds are currently in an ALM project.

Graph	Description
Graph	Specify the type of data displayed along the x-axis, and the build information by which ALM groups the data.
Builds Trend Graph	Shows the history of builds to specific build fields in an ALM project, for each time interval displayed. Specify the field for which you want to view the number of builds and the time period for which you want to view data.

To Generate a Graph:

1. Launch the Graph Wizard

The Graph Wizard guides you through the steps involved in creating a graph and configuring its settings. You launch the graph wizard either from the Analysis View module, or while working in the Builds module.

- In the Analysis View module.

On the ALM sidebar, under Dashboard, select **Analysis View**, right-click a folder, and select **Graph Wizard**.

- In the Builds module.

Select **Analysis > Graphs > Graph Wizard**.

- In the Graph Wizard window change the entity name to Builds. There are two graph types listed. Select the appropriate graph and follow the wizard instructions.

2. Create a predefined graph

While working in the Builds module, generate an impromptu graph to analyze the module data.

- Select Analysis > Graphs and select one of the predefined graphs.

You can choose from :

- Builds - Summary Graph - by 'Build Category'.
- Builds - Summary Graph - by 'Build Category' Group by 'Build Status'.
- Builds - Summary Graph - by 'QA Status'.
- Builds - Trend Graph - Group by 'Build Category'.

3. Create a graph in Analysis View.

- a. Open the Analysis View module. On the ALM sidebar, under Dashboard, select **Analysis View**.
- b. Add a folder to the analysis tree. Right-click a folder under the Private or Public root folder, and select **New Folder**.
- c. Create a new graph. Right-click a folder, and select **New Graph**. Change entity to **Builds**, select required graph type and fill in the graph name in the **New Graph** dialog box.
- d. Click on the **View Tab** in the right panel.

For more information about the generation of graphs see the *HP Application Lifecycle Management User Guide*.

Additionally, a specific project report for Builds is available. For details, see "[Generating Project Reports](#)" (on page 33).