HP Application Lifecycle Intelligence

Software Version: 2.9
For HP ALM 12.50

User Guide

Document Release Date: September 2015
Software Release Date: September 2015

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 - 2015 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

HP Software Solutions Now accesses the HPSW Solution and Integration Portal Web site. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this Web site is **http://h20230.www2.hp.com/sc/solutions/index.jsp**

Contents

Chapter 1: ALI introduction

This chapter contains a high-level look at ALI.

This chapter includes:

ALI overview

Application Lifecycle Intelligence (ALI) tracks development activities and links them to Application Lifecycle Management (ALM) entities. ALI integrates your Source Code Management (SCM) and Build Management tools with Application Lifecycle Management(ALM) and links activities such as code changes, builds, unit test results, and code coverage analyses from your SCM and Build Management tools to ALM entities such as Releases, Requirements, Defects, and Tests.

ALI also enables you to create and enforce SCM policy compliance. For example, you can allow a check-in only if the feature is scheduled for a particular release, or allow a check-in during stabilization only if the change fixes a severe defect. You can require metadata before allowing a check-in, or lock a code-base for a release.

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*.

Usage examples:

- Review code changes linked to a requirement or defect, or first included in a build.
- Check what was implemented during a time period, for a release or build, or by an individual contributor.
- Inspect line by line differences in code changes.
- Identify changes not associated with a requirement or defect.
- Review build and quality metrics.
- See which tests are based on a build.

ALI workflow

This section describes the overall ALI workflow.

1. Set up and configure ALI.
   a. Enable the ALI extension in ALM.

      See "Enable the ALI extension" on page 11.

   b. If you used an earlier version of ALI, follow the steps in "Migrate from previous versions" on page 11.

   c. Configure the synchronization with SCM and build systems.

      See "Configure scheduled synchronization" on page 12.

   d. Set up and configure the connections with your build systems.

      See "Setting up build system integration" on page 15.

   e. Set up and configure the connections with your SCM systems.

      See "Setting up source control management system integration" on page 27.

   f. If you develop for the Force.com platform, set up Force.com integration.

      See "Setting up Force.com integration" on page 45.

2. Track changes in the source code.

   See "Managing SCM changes and traceability" on page 50.

3. Track the impact of development activity on releases, requirements, and defects.

   See "Monitoring development activity" on page 59.

4. Track the quality of your builds.

   See "Monitoring build activity" on page 55.

Chapter 2: Before you begin

This chapter provides details of the tasks you must complete before you can begin setting up and configuring ALI.

This chapter includes:

Enable the ALI extension

In ALM Site Administration, enable the ALI extension for every project in which you want to use ALI.

For more information on enabling extensions in an ALM project, refer to the *HP Application Lifecycle Management Administrator Guide*.

Migrate from previous versions

This section includes:

- "Migrate from ALI 2.0 or earlier"
- "Migrate Perforce Data from ALI 1.1"
- "Upgrade work items"

To identify the ALI version you are working with, select **Help > About** from the main ALM menu.

Migrate from ALI 2.0 or earlier

1. In the Code Changes module, add the **Message** column. After upgrading ALI from a version earlier than 2.0, this column is no longer displayed unless you add it.
2. If you use Perforce for source code management, follow the steps under "Migrate Perforce Data from ALI 1.1" below.

Migrate Perforce Data from ALI 1.1

If you are upgrading from Application Lifecycle Intelligence version 1.1, remove and reload the Perforce data handled by ALM as follows:

1. In 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.

Upgrade work items

ALI provides scripts that you must run to upgrade the SYSTEM_FIELD project table. The scripts are in the ALI bundle in folder **resources\work-items-update-scripts**. Use your database console to run the scripts.

For each project with the ALI extension enabled, run **check-existence-of-has-changeset-linkage-fields.sql** to verify whether the **has-changeset-linkage** fields exist. If the script result is 0, add the fields by running **add-has-changeset-linkage-fields.sql**.

Configure scheduled synchronization

ALI detects changes made on preconfigured SCM repositories and build servers, loads information about code changes and associated file builds and build artifacts to the ALM server, and automatically creates traceability between loaded code changes, builds, and work items (requirements, defects).

At regular intervals, the ALI scheduler runs discovery on the ALI-enabled projects and synchronizes each project. If synchronization of a project ran more recently than the predefined synchronization interval, the synchronization is not performed.

To configure the behavior of the ALI scheduler log in to Site Administration and click the **Site Configuration** tab. Add or set the following ALM Site Administration parameters. If the parameters do not exist, the default values are used.

Caution: Changing these values can affect server performance.

- **ALI_MAX_PROJECT_SYNC_JOB.** The maximum number of synchronization jobs that can run at the same time. The default value is 5.
- **ALI_PROJECT_SYNC_DISCOVERY_INTERVAL.** The interval between checks to determine if any project requires synchronization. The default value is 10 minutes.
- **ALI_MIN_PROJECT_SYNC_PERIOD.** The minimum time interval between synchronizations for a project. The actual run time depends on the number of synchronized projects and synchronization jobs. The default value is 60 minutes.

For example:

1. ProjectA was synchronized at 11:30.

2. Changes were made to ProjectA at 11:45.

3. The scheduler runs a discovery check at 12:00. Result: Synchronization is not performed for ProjectA.

4. ProjectA is going to be synchronized at 12:30 if there is a free synchronization job available.

For more information on setting ALM configuration parameters, refer to the Setting ALM Configuration Parameters section in the *HP Application Lifecycle Management Administrator Guide*.

Configure TFS integration support

For full support of Microsoft Team Foundation Server (TFS) systems, you must install the HP ALI TFS Services on the TFS Server.

To install the services, run the HP ALI TFS Services installer on the TFS server. The installer is included in the ALI bundle. The installer sets up the environment with all necessary validation. HP ALI TFS Installer is valid for all versions of TFS supported by ALI.

For details on supported TFS versions, see "Supported SCM systems" on page 62 and "Supported build systems" on page 62.

This section includes:

HP ALI TFS Services

HP ALI TFS Services include the following:

- **Line Count Service:** Counts changed lines within commits.
- **Build Service:** Provides build information.

- **ALI Agent services:** Services used by the ALI agent for validation of commit messages and push code changes into ALM.

Upgrading from an Earlier ALI Version

If you were working with native TFS integration in HP ALM 11.52 Patch 01 or earlier, it is strongly recommended to uninstall/remove the previous version of HP ALI TFS Services. This is not required if you were previously working with TFS over SvnBridge.

To remove a previous version of HP ALI TFS Services:

1. Uninstall the previous `CheckinEventService` using `BisSubscribe /unsubscribe`.
2. Remove from IIS any IIS Application, IIS Web Site, or IIS Application Pool related to the previous version of the ALI TFS Services.
3. Delete the **AliTfsServicesSite** folder, located under C:\inetpub on the TFS server. This folder is mapped to the previous ALI TFS Services IIS Application.

Moving from TFS over SvnBridge to Native TFS integration

TFS over SvnBridge is no longer supported. To start using the built-in TFS integration:

1. Remove all the previous SCM repositories of the **TFS over SvnBridge** type that are configured in ALM.
2. Add a new SCM repository of the **TFS** type for each Project Collection.
3. Add an SCM branch for each project that you had previously configured in ALI.

Chapter 3: Setting up build system integration

This chapter includes:

Build system setup workflow

Builds are the key deliverables of software development. ALI tracks information about builds together with their relationships to other ALM entities.

Integration with build systems allows you to measure the impact of code changes on software deliverables.You can see reports about what new code has been implemented and what the impact is on the delivered project with the delta of key metrics. For example, a certain commit caused a 5% degradation of test results.

To set up build system integration:

1. Make sure you are working with a supported build system. For details, see "Supported build systems" on page 62.
2. Review the requirements for your build server, and perform any necessary configuration.

   For details, see:

   - "Hudson/Jenkins integration" on the next page
   - "TFS build server integration" on page 20
3. Add a build server to ALI. See "Add a build server" on page 20.
4. Add build configurations. See "Add build configurations" on page 21.

   You can also reuse existing SCM configurations from build configurations that are defined on the build server. See "Reuse SCM configurations from build configurations" on page 22.
5. Set a defect filter to limit the defects displayed on the Build Report Defect Trend graph. See "Set build configuration defect filters" on page 23.
6. Configure build change detection. See "Configure build change detection" on page 23.
7. Customize ALI-specific project lists using ALM Project Customization. See "Customize ALI project lists" on page 24.
8. Tag defects as code issues. See "Track code issues" on page 24.

Hudson/Jenkins integration

If you are working with a Hudson/Jenkins build server, install the HP ALI Hudson/Jenkins plugin, `ali-hudson-plugin.hpi`. The plugin is located in the ALI bundle at `agents\build-integration\hudson\`.

For Perforce, TFS, or Git:

Install the Hudson/Jenkins plugins and any applicable additional HP ALI Hudson/Jenkins plugins.

The Hudson/Jenkins plugins that support the use of Perforce, TFS, and Git SCMs can be downloaded from http://hudson-ci.org or http://jenkins-ci.org. The plugins are available in the public Hudson/Jenkins plugin repository. Support is as follows:

- SVN is supported by default.
- CVS is supported by default in Jenkins.
- CVS is supported by default in Hudson versions 2.x.

For CVS in Hudson 3.x:

Install the Hudson CVS plugin as the prerequisite for the HP ALI Hudson plugin. Hudson removed the CVS plugin which was preinstalled until version 3.0 by default. The CVS plugin can be downloaded from http://hudson-ci.org. The plugin is available in the public Hudson plugin repository.

The additional HP ALI Hudson/Jenkins plug-ins are located in the ALI bundle. Install any applicable to your specific SCM.

For details on how to install the ALI plugins for Hudson/Jenkins, see "HP ALI Hudson/Jenkins plugin deployment" below below.

HP ALI Hudson/Jenkins plugin deployment

To install the HP ALI Hudson/Jenkins Plugin:

1. In Hudson/Jenkins, open the Plugin Manager and click the **Advanced** tab.
2. In the **Upload Plugin** section, browse to `ali-hudson-plugin.hpi` or `ali-jenkins-plugin.hpi`, located in the ALI bundle, and click **upload**.
    - **TFS:** If the project source code built by Hudson/Jenkins is stored in TFS, first install the Hudson/Jenkins TFS plug-in, downloaded from Hudson/Jenkins and then install the `ali-`

`hudson-tfs-plugin.hpi/ali-jenkins-tfs-plugin` and the base plug-in, `ali-hudson-plugin.hpi/ali-jenkins-plugin.hpi`.

> **Note:** When you define a TFS repository in ALI for source control management, the format of the repository location includes the name of the project collection. For details, see "Add a TFS repository" on page 33.
>
> Hudson/Jenkins does not require the name of the project collection, but it must be defined on the Hudson/Jenkins server in order to work successfully with ALI. On the Hudson/Jenkins server, make sure the path to the TFS server includes the project collection name in **Source Code Management > Server URL**.

- **Perforce:** If the project source code built by Hudson/Jenkins is stored in Perforce, first install the Hudson/Jenkins Perforce plugin, downloaded from Hudson/Jenkins and then install the `ali-hudson-perforce-plugin.hpi/ali-jenkins-perforce-plugin.hpi` and the base `ali-hudson-plugin.hpi/ali-jenkins-plugin.hpi`.
- **Git:** If the project source code built by Hudson/Jenkins is stored in Git, first install the Hudson/Jenkins Git plugin, downloaded from Hudson/Jenkins and then, install the `ali-hudson-git-plugin.hpi/ali-jenkins-git-plugin.hpi` and the base `ali-hudson-plugin.hpi/ali-jenkins-plugin.hpi`.

  > **Note:** The ALI Git plugin does not support a branch name with wildcards. On the Hudson/Jenkins server, make sure there is no wildcard defined in **Source Code Management > Branches to build**.

3. After uploading the plugins, restart the Hudson/Jenkins server to enforce changes.
4. Confirm that the installed plugins are listed in the **Installed** tab in the Plugin Manager.
5. Confirm that the **ALI Integration** link is 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, see the Hudson/Jenkins system documentation.

HP ALI Hudson/Jenkins plugin configuration

Access the ALI Hudson plugin global configuration from the global Configure System. Access the job-specific configuration from the job.

All properties specified in the global configuration can be overridden for a particular job.

For detailed descriptions of the properties , see the ALI integration plugin on the Hudson/Jenkins server.

To set the global configuration:

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

- **Include the credentials in the SCM configuration** - Specify if the user name 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: If this option is enabled, the credentials for the SCM repository associated with build configurations are exposed on REST endpoints as plain text.

- **Update build information in HP ALM** - If this option is enabled, the ALM server is notified when a build is started and when it is finished.

  If you want to use the push mechanism and also want to specify ALM properties for individual jobs, this option must be enabled

- **Hudson/Jenkins URL** - Specify the URL of the Hudson or Jenkins server.

  Note: If there are problems with the Build System detail links, ensure that the correct Hudson/Jenkins server URL is entered.

To set a configuration for a particular job:

From the Configure link on left side menu for the job, select the **ALI Integration** option to enable ALI integration for the job.

- **Test sources mapping pattern** - Set test source locations based on actual test results. For details, see the ALI integration plugin help.

- **HP ALM configuration** – Overwrite the global ALI configuration properties. Update build information in ALM for the build job.
- **NCover** - NCover code coverage for .NET configurations. The *NCover report XMLs* specifies the generated raw XML report files, such as `myproject/target/coverage-reports/*.xml`. The basedir of the fileset is the root workspace.
- **Force.com** – For details, see "Setting up Force.com integration" on page 45.

TFS build server integration

Note: For full support of TFS as a build server, the HP ALI TFS Services must be installed on the TFS 2012 Server. For details on configuration and limitations, see "Configure TFS integration support" on page 13.

TFS Build Features and Limitations

TFS build server support exposes data on all the defined build definitions, their builds, code changes, test results, and code coverage per build. In addition, information on all the defined SCM repositories per build definition is exposed. The build statistics are exposed without any limitations except test and code coverage results.

To access the test results and code coverage results required by ALI for the build definition, the definition must meet the following criteria:

- The build definition and build must have a “drop folder” defined. The build must also have a log folder.
- The build must clean the workspace at the beginning of the build. All code changes must be checked out.
- Results of the test and code coverage frameworks that were used must be stored as attachments in the *.trx and *.coverage files, respectively.
- The TFS build server configuration does not support the push mechanism. All builds are stored in HP ALM using the polling mechanism.

Add 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 type and click **OK**.
3. In the New Build Server dialog box, fill in the details.

   Note:

   - The location is a full path to the server such as http://xx.xx.xxx.xxx:xxxx/yyyy. Get the location from your system administrator. The /yyyy segment is optional. It is for the context on which the server is running (such as on a Jenkins server).
   - The build server location is the URL of the service which is part of HP ALI TFS Services which you download and install on the TFS server. The service installer is part of ALI bundle. As the TFS build server location, use the URL in the following format for each project collection:
     http://tfsServer:tfsPort/tfsali/Service/BuildService.svc/ali/ProjectCollectionName.

     Where:

     - tfsServer – address of the TFS server
     - tfsPort – port of the TFS server
     - tfsali – name of the application on the IIS Site defined during installation of HP ALI TFS Services
     - ProjectCollectionName – name of project collection
4. Select the **Change Detection** tab and optionally:
   - Select **Read changes from build server** if required. Changes will be read periodically from your SCM repository.
   - Select **Receive builds transmitted by build server agent(s)** if required. See "Configure build change detection" on page 23 for more information.

Add build configurations

To add a Build Configuration:

1. In the Build Configurations tab, click **New Build Configuration**.
2. Select a build configuration from the list and click **OK** to open the details window.

   The list contains the unused configurations of the governed server.

   The list contains only the configurations for Hudson/Jenkins jobs with ALI integration enabled.
3. Click the **Release** drop down list and select an associated release.
4. Click the **Build Configuration** drop down list 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.

Reuse SCM configurations from build configurations

Existing build configurations and their SCM repository configuration settings stored in the build server can be applied to new build configurations in HP ALM.

To apply an existing SCM build configuration:

1. In the **Build Servers** page of the Management module, select a server.
2. Select the **Build Configurations** tab.
3. Select a configuration and click it.
4. In the **Build Configuration Details** window, 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. Clicking the details button displays the repository or branch details pop-up window of the selection.

   The status message next to the repository or branch location shows whether the repository or branch is already defined.

Set build configuration defect filters

Setting this filter reduces the 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 build configuration:

1. In the **Build Servers** page of the Management module, select a server.
2. Select the **Build Configurations** tab.
3. Click a configuration name to open 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** and click **Yes** to confirm.

Configure build change detection

When ALI detects new builds on build servers, it 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.

Set Change Detection settings in the Change Detection view if you are defining a new build server . For an existing server definition,use the Change Detection Tab.

Polling:

Polling means that ALI periodically checks for new builds in the build server and loads them on the ALM server.

To activate the polling mechanism, set the *Read changes from build server* option on the Change Detection tab. To load changes on demand, use the **Synchronize** command. To configure scheduling for automatic load changes, see "Configure scheduled synchronization" on page 12.

Push:

To start the plugin listening on the build server, set the *Update build information in HP ALM* option in the ALI integration plugin on the build server. See "HP ALI Hudson/Jenkins plugin configuration" on page 19. When a new build is starting, the plugin pushes information about the build to the ALM server. When the build completes, the status and other data are updated on the ALM server.

To activate the push mechanism, set the *Receive builds option transmitted by build server agent(s)* option on the Change Detection tab.

Customize ALI project lists

In ALM Project Customization, in the Project Lists page, you can customize the following default ALI project lists:

ALM Project List	Description
ALI Build Category	The values available in the **Build Category** field.
ALI Closed Status	When a defect's **Status** field is set to one of the values defined in this list, ALI considers the defect closed. You can set a value for **Closed In Build** only if the defect status a member of the Closed Status list. The Closed Status values are a subset of the Status project list.
ALI Code Issue Definition	The values that mark a defect as a **code issue**. For more details on code issues, see "Track code issues" below.
ALI QA Status	The possible values for a build's **QA Status**.
ALI Reported Severity	The values available for links to code issues, displayed in the Development Activity tab in the Requirements or Defects modules, and in Build Details. The values must be the same values or a subset of the values of the **Priority** project list.

For more details on customizing project lists, refer to the *HP Application Lifecycle Management Administrator Guide*.

Track code issues

If a problem is identified in your source code, you can track it in the defects, requirements, and builds that are associated with the problematic code.

When you create a new defect to fix the problematic code, ALI can tag the defect as a **code issue**, based on criteria that you set. After you configure code issue tracking in your project, you can view links to code issue defects from the related Requirement, Defect, and Build Details dialog boxes.

To configure code issue tracking in your ALM project:

1. **Prerequisite:** Customize the default ALI project lists. For details, see "Customize ALI project lists" on the previous page.

2. In Project Customization, in the Project Entities page, create a user-defined field to use for code issue tracking. Supported field types are **String** or **Lookup List**.

   For more details on creating user-defined fields, refer to the *HP Application Lifecycle Management User Guide*.

3. In the Project Lists page, in the **ALI Code Issue Definition** project list:

   a. Add the value of the user-defined field's **Name** field as the only list item.

   b. Select a value from the user-defined field you created in Step 2 to use to tag the defect as a code issue. Add this value as a sub-item.

Example:

Create a user-defined field. For example, **Defect Category**. This field might have values such as **Bug**, **Issue**, **Enhancement**, **Fortify**.

In the **ALI Code Issue Definition** project list, add the user-defined field's **Name** field as the only list item. For example, `BG_USER_05`.

Add the value of the user-defined field that you want to use to tag the defect as a code issue as a sub-item in the list. For example, `Fortify`.

To track code issues:

1. **Prerequisite:** To identify a defect as a code issue of a requirement, defect, or build, the defect must meet the following conditions:

   - The defect is not closed. A defect is closed if the defect **Status** is one of the values defined in the **ALI Closed Status** project list.

   - The defect's **Priority** is one of the values defined in the **ALI Reported Severity** project list.

2. Tag the defect as a code issue: In the Defect Details dialog box, fill in the user-defined field that was created above for code issue tracking.

3. Connect the defect to the defect, requirement, or build associated with the problematic code change:

    a. In the Defect Details dialog box, select the appropriate value from the ALI **Caused by Code Change** field, which lists code changes and commit messages.

    b. In the Defect Details dialog box, select a value in the **Detected in Build** field.

Example:

1. You commit a code change related to a specific defect, requirement, or build. For example, a code change you made when fixing Defect 124.
2. A reviewer, or the third-party tool you are using for code analysis, recognizes that the code change causes a problem. You create a new defect, making sure to set the following two fields:

    a. In your code issue user-defined field, **Defect Category** in this example, select the appropriate value. In this example, you would select **Fortify**.

    b. In the **Caused by Code Change** field, locate and select the code change that caused this issue. For example, `Defect #124 : Fixing authentication issue`.

Results:

The new, code issue defect is listed in the associated Build, Requirement, or Defect Details dialog boxes.

Chapter 4: Setting up source control management system integration

This chapter includes:

SCM setup workflow

Before ALI can report on code changes and link them to ALM entities, you must set up source control management system (SCM) integration.

To set up SCM integration:

1. Make sure you are working with a supported SCM system. For details, see ""Supported SCM systems" on page 62.
2. Review prerequisites.

   See "SCM integration prerequisites and limitations" on the next page.
3. Add repositories.

   See "Add SCM repositories" on page 30.
4. [Optional] Install the agents for your SCM.

   See "SCM agents" on the next page.
5. [Optional] Set an external repository viewer.

   See "Set an external repository viewer" on page 35.
6. [Optional] Require a particular format for check-in messages. See "Set commit patterns" on page 37.
7. Add branches and associate them with a release.

   See "Add branches and associate with release" on page 41.
8. [Optional] Control commits at the level of an SCM branch by setting check-in policies and locking for a branch. See "Set branch check-in policies" on page 42.
9. Configure detection of changes in source.

   See "Configure code change detection" on page 43.

SCM integration prerequisites and limitations

Application Lifecycle Intelligence (ALI) supports Subversion, Perforce, GIT, and CVS systems natively. However, if check-in policies and/or the push mechanism is required, SCM agents need to be installed on the SCM System. For details, see "SCM agents" below.

TFS systems are supported natively. To support the push mechanism, install the TFS agent web services. Furthermore, without the TFS agent web services, all code changes are displayed with “0” in the changed lines field.

Prerequisites:

MS PowerShell 2.0 or later must be installed on Windows and enabled to run scripts. Linux/Unix agent scripts use BASH. The BASH version must be 4.0 or later to support pushing into multiple repositories.

Limitations:

Git: ALI does not support the Git functionality of changing the history of commits. If commit history is changed in a Git repository, you must reload the entire commit history tracked by ALI for all affected branches. To reload the commit history, remove these branches and add them again.

SCM agents

An HP ALI agent is a set of scripts or proprietary applications installed on an SCM server. The ALI agent listens to activity on the SCM server, reports the activity to ALI, and supports the integration with ALI. For example, when a change is committed to a configured repository and branch, the agent checks whether the commit is allowed according to your policies, and pushes the code change to the ALM server if the commit is allowed.

For details on configuring SCM agents, see the `readme.txt` file, located in the specific agent archive inside the main ALI distribution archive.

For more details on working with SCM agents, see the following sections:

- "Configure code change detection" on page 43
- "Set branch check-in policies" on page 42

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 Agent for Linux

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

CVS Agent for Windows

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

TFS Services for Windows

`agents\scm-integration\windows\ali-tfs-services.zip`

Perforce Agent for Linux

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

Perforce Agent for Windows

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

Git Agent for Linux

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

Git Agent for Windows

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

Add SCM repositories

Configure SCM repositories to enable loading code changes from SCM systems and to enable automatic traceability of:

- work items (requirements/defects)
- code changes
- defined releases

This section includes:

Add a Subversion repository

For the list of supported repositories, see "Supported SCM systems" on page 62.

To add a repository:

1. Log in to ALM with Administrator privileges. On the ALM sidebar, under Management, select **SCM Repositories** to open the Repositories page.
2. Click **New Repository**. From the drop down list, select an SCM type and click **OK**.
3. In the **New SCM Repository** window, fill in the fields.

   The location is the full path to the repository. For example, `http://host/svn/repo`. Get the location from your system administrator.

   The repository location points to the actual repository root. Branches are later used for specifying paths within the repository. If an SVN URL has an unknown root, use the "`svn info <URL>`" command to find the root.

To edit a repository:

1. Select a repository in the left pane of the Repository page.
2. Edit the fields in the right pane, using the tabs to access different groups of settings.
3. If a repository property requires modification, select it and click the **Edit Property** button. In the **Edit Property** window, add a value and click **OK**. The properties are in bottom area of the details tab.

Add a CVS repository

For the list of supported repositories, see "Supported SCM systems" on page 62.

To add a repository:

1. Log in to ALM with Administrator privileges. On the ALM sidebar, under Management, select **SCM Repositories** to open the Repositories page.
2. Click **New Repository**. From the drop down list, select an SCM type and click **OK**.
3. In the **New SCM Repository** window, fill in the fields.

The location is the full hostname of the CVS server. Get the hostname from your system administrator.

In the alias property, enter the full `CVSROOT` exactly as it is configured in your build system. For example, `pserver:username:password@host/cvsrepo`. The CVSRoot property is mandatory for the pserver protocol.

4. CVS repositories require the CVS protocols *pserver* and *initial date* to start loading code changes. The CVS root should also be specified, but it is not mandatory.

To edit a repository:

1. Select a repository in the left pane of the Repository page.
2. Edit the fields in the right pane, using the tabs to access different groups of settings.
3. If a repository property requires modification, select it and click the **Edit Property** button. In the **Edit Property** window, add a value and click **OK**. The properties are in bottom area of the details tab.

Add a Perforce repository

For the list of supported repositories, see "Supported SCM systems" on page 62.

To add a repository:

1. Log in to ALM with Administrator privileges. On the ALM sidebar, under Management, select **SCM Repositories** to open the Repositories page.
2. Click **New Repository**. From the drop down list, select an SCM type and click **OK**.
3. In the **New SCM Repository** window, fill in the fields.

   ALI repositories are equivalent to Perforce Depots. The location of the repository is the host name and port of the Perforce Server, and the Depot name. The format is `host:port//depot_name`.

To edit a repository:

1. Select a repository in the left pane of the Repository page.
2. Edit the fields in the right pane, using the tabs to access different groups of settings.

3. If a repository property requires modification, select it and click the **Edit Property** button. In the **Edit Property** window, add a value and click **OK**. The properties are in bottom area of the details tab.

Add a TFS repository

For the list of supported repositories, see "Supported SCM systems" on page 62.

To add a repository:

1. Log in to ALM with Administrator privileges. On the ALM sidebar, under Management, select **SCM Repositories** to open the Repositories page.

2. Click **New Repository**. From the drop down list, select an SCM type and click **OK**.

3. In the **New SCM Repository** window, fill in the fields.

   ALI repositories are equivalent to TFS Project Collections. The location of the repository is the host name and port of the TFS Server, and the path to the Project Collection. The format is `http://host:port/tfs/path_to_project_collection`.

   If the line count service URL is different than the default value `http://tfsServer:tfsPort/tfsali/Service/LineCountService.svc`, then you must enter the line count service URL. If it is not set, the branch is created with warnings and all loaded code changes are stored with 0 changed lines. The line count service is part of the ALI TFS SCM agents which you download and install on the TFS Server. The HP ALI TFS Services are part of the ALI bundle.

   Note: . The previous service, CountLinesService.svc, is not supported as of ALI version 2.0.0.139876.

To edit a repository:

1. Select a repository in the left pane of the Repository page.

2. Edit the fields in the right pane, using the tabs to access different groups of settings.

3. If a repository property requires modification, select it and click the **Edit Property** button. In the **Edit Property** window, add a value and click **OK**. The properties are in bottom area of the details tab.

Add a Git repository

For the list of supported repositories, see "Supported SCM systems" on page 62.

To add a repository:

1. Log in to ALM with Administrator privileges. On the ALM sidebar, under Management, select **SCM Repositories** to open the Repositories page.
2. Click **New Repository**. From the drop down list, select an SCM type and click **OK**.
3. In the **New SCM Repository** window, fill in the fields.

   The repository location is the entire Git URL. For example: `https://github.com/hp/ali.git`. Do not include the user name in the URL: `ssh://~~git@~~server1.abc.com/home/git/gitrepo.git`. Enter the user name in the **Username** field.

   Supported protocols include http(s), Git, and ssh.

   For ssh authentication using a private key, use the Security Key section.

   Note: If you are facing connection issues to github.com, make sure the proxy is set in ALM.

   To set the ALM proxy, add the following line into the wrapper.conf file, located in the ALM application deployment folder (C:\ProgramData\HP\ALM\):

   ```
   wrapper.java.additional.[nextFreeNumber]=-Dhttp.proxyHost=[proxyServerUrl] -
   Dhttp.proxyPort=[proxyPort] -Dhttps.proxyHost=[httpsProxyServerUrl] -
   Dhttps.proxyPort=[httpsProxyPort]
   ```

   Where:

   - [`nextFreeNumber`] is next unused number of parameter of the wrapper.conf
   - [`proxyServerUrl`] is URL of the proxy server
   - [`proxyPort`] is a port number of HTTP
   - [`httpsProxyServerUrl`] is URL of the HTTPS proxy server (optional parameter)
   - [`httpsProxyPort`] is a port number of HTTPS proxy (optional parameter)

4. If you configured the GitHub repository, select the "GitHub view diff/file link templates" property to use the native GitHub file view and the diff view web tools.

To edit a repository:

1. Select a repository in the left pane of the Repository page.
2. Edit the fields in the right pane, using the tabs to access different groups of settings.
3. If a repository property requires modification, select it and click the **Edit Property** button. In the **Edit Property** window, add a value and click **OK**. The properties are in bottom area of the details tab.

Set an external repository viewer

ALI provides a built-in repository viewer for viewing file diffs and details. You can also use an external repository viewer, such as ViewVC.

If you use an external viewer, 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 code changes table in the UI. Each file in the code change contains a link displaying a diff view (revision in current code change to its previous revision). The template can contain tags/variables which are expanded at run time from the context of the currently selected code change in the UI.

Tags/Variables that may be substituted/expanded:

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

`${revision}` ... revision in current code change.

`${fromRevision}` ... previous revision to revision in current code change.

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

Note: The following address enables you to view the diff of two files stored in:

TFS

```
http://tfshost:tfsport/tfs/_COLLECTION_/_
versionControl/changesets#opath=${fromFilePath}&oversion=${fromRevision}
&mpath=${filePath}&mversion=${revision}&_a=compare
```

Where:

- tfshost: address of the TFS server
- tfsport: port where TFS server is running
- _COLLECTION_: name of the collection where the requested file's project is a member
- opath: full path to the original source file (including '$' symbol and project name) e.g.$/ali-replica/alik/pom.xml
- ocs: TFS original code change ID
- oversion: TFS original code change ID
- mpath: full path to the modified source file (including '$' symbol and project name) e.g.$/ali-replica/alik/pom.xml
- mcs: TFS modified code change ID
- mversion: TFS modified code change ID

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

Tags/Variables that may be substituted/expanded:

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

`${revision}` ... revision in current code change.

Note: The following address enables you to view the source of files stored in TFS:

TFS

```
http://tfshost:tfsport/tfs/_COLLECTION_/_
versionControl/changesets#cs=${revision}&path=${filePath}&version=${revision}&_
a=contents
```

Where:

- tfshost: address of the TFS server
- tfsport: port where TFS server is running
- _COLLECTION_ : name of the collection where the requested file's project is a member
- path: full path to the source file (including '$' symbol and project name) e.g. $/alireplica/alik/pom.xml
- cs: TFS code change ID
- version: TFS code change ID

Set commit patterns

Detecting and maintaining traceability between SCM changes and ALM requirements and defects is based on the commit message entered in the SCM system. ALI enables you to define patterns for the commit message that include requirement or defect IDs.

To set commit patterns:

1. Choose the **Commit Pattern** view while setting up a new repository, or open the **Commit Pattern** tab of an existing repository.
2. Select either **Basic** or **Advanced** to choose how to define the pattern.
3. If you use **Basic**, add the keywords, ID prefix characters and other options as described immediately below.
4. If you use **Advanced**, edit the default pattern as described below under "Advanced pattern definition" on the next page.

Basic pattern definition:

1. Add keywords for Defects or Requirements by clicking the **Add** button, entering a keyword or phrase, and clicking **OK**. The keywords mark a commit as related to either a defect or a requirement.

   To associate code changes with alternative identifiers, such as IDs from an external tracking tool, see "Associate code changes with alternative identifiers" on page 40.

2. Add an ID Prefix characters to the pattern by typing in the characters. Select **Optional** if the prefix is not required.

3. Click the **More Options** button to add the following options:

   - **Include Default Tasktop Commit Pattern**. If selected, when the Taskop plugin is integrated with ALM, Tasktop generates default commit messages according to a pattern that is recognized by ALI.

   - **Case Sensitivity Commit Messages**. If selected, commit messages patterns are case sensitive.

   - **Multiple Defects or Requirements separation markup**. Type in the separator between multiple defects or requirements in a commit message.

   - **Keyword location in commit message**. Select whether keywords are recognized only at the beginning of the commit message or anywhere within the message.

   - **User commit message separator**. Type a character to separate the defects or requirements specification from a free-text user commit message.

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

Example of options in a commit message: The message: `fixing defect #100, #101: fix caching and enhance functionality`
Elements of the message:

- *fixing defect* - the defect keyword
- *#100* - defect ID prefix and defect ID
- *","* - multiple defects separator
- *":"* - user commit message separator
- *fix caching and enhance functionality* - free-text user commit message

Advanced pattern definition

1. Add to or change the default code pattern.

2. Test the modified code by clicking **Test Against Existing Commits**.

3. **Case Sensitive** enforces case sensitivity for messages.
4. You can test a message by typing it into the **Custom Commit Message Test** box.
5. Navigate away from the tab to commit the configuration.

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

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 such as:

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

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

Associate code changes with alternative identifiers

By default, you can link code changes to ALM requirements or defects based on their ALM entity IDs. If you maintain another set of identifiers for your requirements or defects, you can associate code changes with these alternative identifiers.

For example, if you synchronize your projects with an external defect tracking tool, you can reference the identifiers generated by that tool in your commit messages. ALM can then use the alternative identifier to link the code change to an ALM requirement or defect.

To associate code changes with alternative identifiers:

1. Prerequisites:

   a. The alternative IDs must be stored in an ALM user-defined field. The supported field types are **Number** and **String**.

   b. The alternative ID must be unique. The code change is linked to the first matching ALM entity.

2. In the SCM Repositories module, select a repository and click the Commit Pattern tab. Make sure the **Basic** view is selected.

3. In the Defects or Requirements section, click **Add**.

4. In the **Add keyword** dialog box:

   a. In the **Keyword** field, enter a keyword or phrase to indicate that the commit message is using an alternative identifier. For example, `Resolve Jira issue`. Developers must include this

keyword or phrase in their commit messages.

b. In the **Field** list, select the user-defined field that stores the alternative identifiers.

When a commit message contains the keyword or phrase defined here, ALM looks at the user-defined field to locate the entity you want to associate with the code change.

Add branches and associate with release

This section includes:

- **To add a new branch**
- **To associate a branch with a release**

To add a new branch:

1. On the ALM sidebar, under Management, select **SCM Repositories**, and open the Branches tab.

   Click **Add** .

2. Enter a path for the branch, and optionally, enter a **Branch** and a **Last Change Read** property if required for the repository type, according to the following guidelines:

   - Do not define the **Branch** property for SVN, TFS, or Perforce.
   - The **Last Change Read** field value may be one of the following:
     - For a CVS repository, the Last Change Read field should contain date/time.
     - For all other repositories, the Last Change Read field should contain the revision number.

     To read all change sets from the given branch, leave this field empty.
   - **CVS**:
     Branch name only has meaning for some CVS repositories.
   - **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 to `/HelloWorld/releases/release-1.0`. Do not set the parameter branch even though the branch is named.

- **TFS:**
  Use the branch path to the project path without '$'. For example, if the project is located at `$/TestApp`, then branch path should be `/TestApp`. The branch path must contain only the name of the project. Paths containing subfolders are not supported.

- **Git:**
  Set the branch path to “/”. The branch should be set to the real Git branch name. Use only the simple branch name in the format "master", and not "refs/head/master".

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 **Add** and select the release in the drop down menu.

3. By default, the **Start Date** and **End Date** are taken from the release. They can be changed by clicking in the date field and typing in a required date.

   Code Changes from the branch in the given time period are associated with the specified release. These sets are visible in the Code Changes module when selecting the specified release.

   More than one release can be associated with a branch. Generally, releases associated with a branch have different time periods.

Set branch check-in policies

You can restrict check-ins on a branch according to several different properties of the check-in. You can use locking for more specific permissions and restrictions.

The check-in policy feature requires the installation of an agent. For details, see "SCM agents" on page 29.

Note: These features are not supported for a Git repository. Settings are hidden.

To specify check-in policies for a branch:

1. In the left pane of the **SCM Branch Details** window, click **Enforcement**.

2. Select from the following options on the **Check-in Policies** tab:

- **Commit message must match defined pattern**. This is the pattern commit messages must match for the check-in to be allowed by the agent.
- **Code Change refers to a Requirement**. If selected, a commit must refer to a requirement.
  - **Has Requirement Type**. If you select this option, then select the requirement types from the drop-down list, Only check-ins associated with requirements of the selected types are allowed by the agent.
  - **Has Priority**. If you select this option, then select priorities from the drop-down list, only check-ins associated with requirements of the selected priorities are allowed by the agent.
- **Code Change refers to a Defect**. If selected, a commit must refer to a defect.
  - **With Severity of**. If you select this option, then select severities from the drop-down list, only check-ins associated with defects of the selected severities are allowed by the agent.
- **Add this note to the system message when a Commit is blocked**. Type a custom system message to send to a user when a commit is blocked. ALI blocks a user from committing changes according to the options specified in the Check-in Policies tab.

To specify locking policies for a branch:

On the **Locking Policies** tab you can restrict check-ins on a branch using the following options:

- **Disallow commits except for the following**. Select to disallow commits on the given branch. You can specify the following exceptions:
  - **Allow Users**. A list of users allowed to commit to the branch.Type in the user names of SCM users who are allowed to commit.
  - **Allow Defects**. A list of defects. If a commit is associated with these defects, it is allowed. Click **Add defect** and type a defect ID. To remove a defect from the list, select the row and click **Remove Defect**.
- **Add this note to the system message when a Commit is blocked**. Type a custom system message to send to a user when a commit is blocked.

Configure code change detection

ALI detects changes made to SCM repositories and associates the code changes with work items, such as requirements and defects. ALI supports two change detection mechanisms: poll and push.

When polling is enabled, ALI periodically checks for new changes in the given SCM repository. Polling operates without SCM agents.

When the push mechanism is enabled, an agent on the SCM system detects changes. When a change is committed to the repository and branch, the agent checks the policies. If the commit is allowed, that agent pushes the code change data to ALM.

To set change detection:

- While setting up a new repository, use the **Change Detection** view in the New SCM Repository dialog box.
- For an existing repository, select the repository in the SCM Repositories module, and click the **Change Detection** tab.

Select the detection options:

Option	Comment
Read changes from SCM	Select this option to enable polling. To configure scheduling for automatic load changes, see "Configure scheduled synchronization" on page 12. Immediate readings can be forced by clicking the **Synchronize** button. The synchronize process can be monitored using the **Task Manager** tool of ALM found under the Tools menu.
Receive changes transmitted by SCM Agent(s)	Select this option to enable pushing changes. If SCM agents have been set up, changes begin to be processed immediately when this option is selected. For details, see "Supported SCM systems" on page 62

Chapter 5: Setting up Force.com integration

Teams developing for the Force.com platform can benefit from features that ALI brings to standard development. Although 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).

This chapter includes:

Prerequisites

Integration between Force.com and ALI requires:

- Make sure you are working with a supported version. For details, see "Supported Force.com versions" on page 63.
- Force.com source code must be stored in an SCM system. For details, see "Supported SCM systems" on page 62.
- Hudson or Jenkins plugins must be installed.
  - Hudson/Jenkins plugin supporting the use of SCM.
  - HP ALI Hudson plugin//HP ALI Jenkins plugin (`ali-hudson-plugin.hpi/ali-jenkins-plugin.hpi` from `ali-bundle`).
  - HP ALI Hudson/Jenkins Force.com plugins (`ali-hudson-salesforce-plugin.hpi/ali-jenkins-salesforce-plugin.hpi` from `ali-bundle`)

  See "HP ALI Hudson/Jenkins plugin deployment" on page 17.
- A build management server to deploy source code to the integration/staging environment must be configured.
- Apache Ant must be installed. (Download from http://ant.apache.org/).
- The HP force-deploy-task must be installed. The task is located in the ALI archive (`/tools/force-deploy-task/force-deploy-task-bundle.zip`. Unpack the zip file to *<ant_install_dir>*`/lib`).

Set up project deployment, testing, and report generation

Deployment of source codes, testing, and report generation is run by an Ant task. The required class for the deploy task is installed as part of the HP force-deploy-task. See "Prerequisites" above.

Define the task in an Ant build script *<Force.com project root>*`\build.xml`. Add the task to an existing `build.xml` or create the file.

Example of running all tests:

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

```
<project name="Sample usage of 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>
```

Example of running tests that match filter:

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.

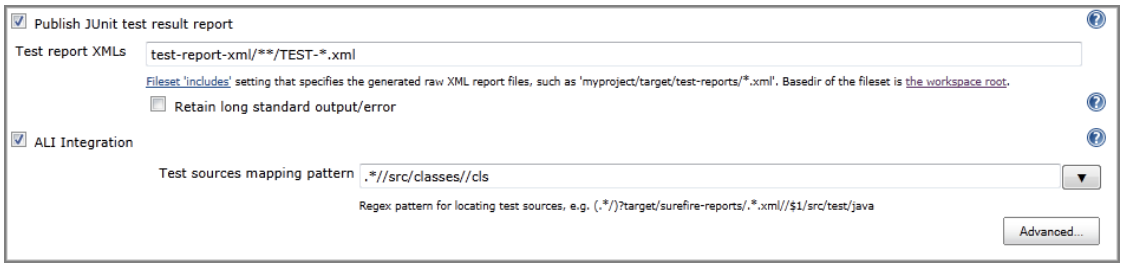
```
<project name="Sample usage of deploy task"
    default=" deployAndTestAndReport " basedir=".">
    <target name="deployAndTestAndReport">
        <taskdef name="sfdeploy" classname=
            "com.claimvantage.force.ant.DeployWithXmlReportTask"/>
        <delete dir="test-report-xml" quiet="true"/>
        <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. Applies only if
               runalltests is false.-->
        <batchtest>
            <fileset dir="src/classes">
                <include name="*Test.cls"/>
            </fileset>
        </batchtest>
    </target>
</project>
```

Attributes of deploy task (`sfdeploy` in the examples above)

Element	Attribute	Description
sfdeploy		The deploy definition
sfdeploy	username	Log-on name to Force.com environment
sfdeploy	password	Password to Force.com environment
sfdeploy	serverurl	URL of login page to force.com environment
sfdeploy	deployRoot	Path to source code directory that contains folders, classes, triggers and so on.
sfdeploy	runalltests	If `true`, all tests are started and code coverage of project is reported. If `false`, only tests specified by `batchtest` element are started. Code coverage is not reported.
sfdeploy	reportDir	Directory for reports
batchtest		Tests to be started if `runalltests="false"`.
batchtest/fileset	dir	Location of batchtest test files
batchtest/include	name	Class name pattern of batchtest tests

Configure Hudson/Jenkins - Force.com

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 deploy targets. See the example *target name="deployAndTestAndReport"* in topic "Set up project deployment, testing, and report generation" on page 46
3. In the *Post-build Actions* section, configure like this:



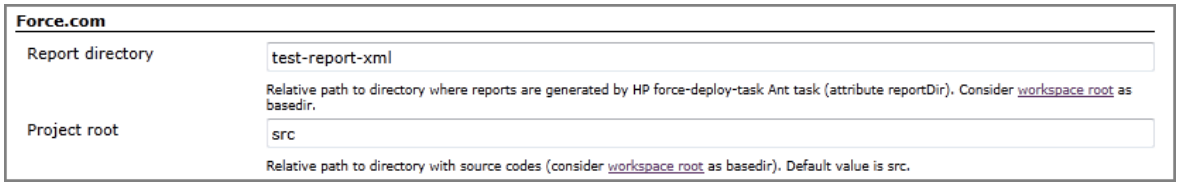
4. In the test report XMLs, replace the `test-report-xml` string with your actual report directory. This is the same as the value of the `reportDir` attribute in the Ant deploy task.

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.

Configuration example:



Chapter 6: Managing SCM changes and traceability

After the repositories and branches are set up, Application Lifecycle Intelligence (ALI) aggregates the data and presents them in the Code Changes module of ALM. This module provides information about what is happening to the source code in a project and maintains traceability between the source code, requirements/defects, and releases.

This chapter includes:

SCM changes and traceability workflow

To manage and monitor SCM changes:

1. Require a particular format for the check-in message. See "Set commit patterns" on page 37.
2. Control commits at the level of an SCM branch by setting check-in policies and locking for a branch. See "Set branch check-in policies" on page 42.
3. View code changes. See "View code changes" below.
4. View the impact of code changes on defects and requirements. See "View the impact of code changes" on the next page.
5. Review and compare the development activity of your teams and developers.

   See:

   - "Generate project reports" on the next page.
   - "Generate graphs" on page 53.

View code changes

To access the **Code Changes** module select Development on the ALM sidebar and click Code Changes.

Each record in the Code Changes grid presents data on one commit action, whether a single file or multiple files were committed to the SCM.

The color of the record indicates the type of work item that the code change is linked to.

Color	Description
Blue	linked to a requirement
Green	linked to a defect
Red	unassigned (not linked to any work item)

The Message column contains the commit message entered by the developer. The Status Message displays an error or warning when the commit message does not correspond to a predefined pattern.

Enabling the Information Panel under the **View** menu opens the panel under the table with tabs for viewing associated files and work items. An asterisk (*) on one of these tabs indicates that there is data in that view pane. The tabs for view panes that display simple text are not marked with an asterisk if there is text.

Code change details

The Code Changes details view presents additional details.

To view Code Change Details:

To open the Code Change Details window, click on the **Revision** field of a Code Change or click the **Code Change Details** button  .

Clicking **View Changes** in the **Diff to Previous** column displays the differences from the previous version of the file.

View the impact of code changes

The Change Impact Report aggregates code change data by requirement and defect. The report shows the size of the related changes and the developers who checked in the code. A graph at the top right of the report shows the distribution of effort by requirements, defects, and unassigned code changes. The effort is expressed in KLOC (Kilo lines of code).

To view the Change Impact Report:

In the Code Changes module, select a time period and a release from the toolbar or grid, and click **View Report**. All other filter values are ignored.

At the bottom of the Change Impact Report, the following options are available:

- **Show Unassigned Changes**. Displays changes that are not associated with a defect or requirement.
- **Print This Report**.
- **Email This Report**. Creates an email in your email client that contains the URL of the report.

Generate 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 enable 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*.

Generate graphs

You can generate the following graphs:

Graph	Description
Code changes Progress Graph	Shows how many code changes accumulated in an ALM project at specific points during a period of time.
Code changes Summary Graph	Shows how many code changes are currently in an ALM project.
Code changes Trend Graph	Shows the history of changes to specific code changes fields in an ALM project, for each time interval displayed.

To generate a graph, do one of the following:

- **Launch the Graph Wizard.**

  In the Code Changes module, select **Analysis** > **Graphs** > **Graph Wizard**.

  In the Graph Wizard window, select the appropriate graph and follow the wizard instructions.

- **Create a predefined graph in the Code Changes module.**

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

- **Create a graph in Analysis View.**

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

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

Chapter 7: Monitoring build activity

After you set up the build systems integration, ALI 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 on reported errors. Depending on the 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.

Project reports and graphs for Builds are available. For details, see "View build reports" on the next page and "Generate build graphs" on page 57

This chapter includes:

View builds

To access the **Code Builds** module select Development on the ALM sidebar and click Builds.

The Build Status field displays each status according to the following colors:

Color	Description
Green	Successful build
Yellow	Warning. Indicates a problem with the build
Red	Build failed

To view details of a build, click on a build number, or select a build and click the **Build Details** button  .

View build reports

Build Summary Report

The **Build Summary Report** provides an overview 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 the Build Summary Report:

Click the **View Report** button to generate the summary report.

Specific Build Report

The **Specific Build Report** displays the requirements, tests, delivered defects, new defects, and closed defects associated with the 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 **Show Unassigned Changes** to display code changes in the build that are not associated with a requirement or defect.

Build Change Report

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

To view a Build Change Report:

1. Select a build and click **Build Details**.
2. On the **Build Details** page, click **View Report** to generate the change report.

Generate build graphs

You can generate the following types of graphs:

Graph	Description
Builds Summary Graph	Shows the number of builds in each of the categories you specify as the X-axis.
Builds Trend Graph	Shows the history of builds by fields you specify, for each time interval displayed.

To generate a graph, do one of the following:

- **Use the Graph Wizard.**

  To launch the wizard either:

  - On the ALM sidebar, under Dashboard, select **Analysis View**, right-click a folder, and select **Graph Wizard**. In the Graph Wizard window change the entity name to Builds.

  Or:

  - In the Builds module, select **Analysis** > **Graphs** > **Graph Wizard**.

- **Generate a predefined graph.**

  Select **Analysis > Graphs** in the Builds module and select one of the predefined graphs.

- **Generate a graph in Analysis View.**

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

For more information about generating graphs, refer to the *HP Application Lifecycle Management User Guide*.

Additionally, a specific project report for Builds is available. For details, see "Generate project reports" on page 52.

View build-test traceability

Build-test traceability enables you to create and track the associations between specific builds, and specific test sets and runs.

Before you run tests, you define a test set. You can then define the build you want to use for the test set. This means that all the tests are going to be run on the selected build. On the basis of this association, you can review results of test sets and test runs for particular builds.

You can set and view associations between builds and tests in the following locations:

Location	Build-test traceability
Builds module > Build Report	In the Build Details dialog box, click **View Report** to view the Build Change Report.
Builds module > Tests tab	In the Builds module window, or in the Build Details dialog box, click the **Tests** tab.
Test Lab module > Test Set Details dialog box	The **Test Build** field displays the build number associated with the selected test set. Select a build to associated with the test set.
Test Lab module > Manual Runner dialog box	The **Test Build** field displays the build number associated with the selected run.
Test Lab module > Run Details dialog box	The **Test Build** field displays the build number associated with the selected run.

For more details on the Test Lab module, refer to the *HP Application Lifecycle Management User Guide*.

Chapter 8: Monitoring development activity

After build servers and configurations are set up, ALI displays the Development Activity tab in the Releases, Defects, and Requirements modules of ALM. This view pane displays detailed information and statistics for the selected ALM entity.

If there is development activity associated with a release, defect, or requirement, a green star icon is visible on the Development Activity tab.

Monitor Development Activity in the Releases module

In the **Development Activity** tab:

- Click **Change** to see the activity of other builds, if there is more than one related build.
- Click **Coverage** to see the Coverage Report.
- Click **Unit Tests** to see the Unit Tests Report.

The success rate in the Build Status area is the percentage of builds that completed with status Success or Warning. Click **Configure Builds** to change the configuration. For details, see "Add build configurations" on page 21.

Monitor Development Activity in the Requirements module

In the **Development Activity** tab:

- Click **Change** to see the activity of other builds, if there is more than one related build.
- Click **Coverage** to see the Coverage Report.
- Click **Unit Tests** to see the Unit Tests Report.
- Click the message link in the code change to view the Code Change details page.

Monitor Development Activity in the Defects module

To view Code Changes and Active Developers associated with a defect, do one of the following:

- Click the **Development Activity** tab in the table below the grid.
- Select **Development Activity** in a **Defect Details** window.

To view details of a code change, click the Message link in **Code Changes** .

ALI adds three values to the Defect Details page:

- **Detected in Build** – the build the defect was detected on.
- **Closed in Build** - the build the defect was closed on.
- **Caused by Code change** – the code change that caused this defect.

If a static code analysis tool supporting integration with For HP ALM 12.50 has been configured, these values are automatically set. If not, they can be set using the drop down list.

Data from these fields can be viewed in the Defects section on the Build Details page. The data is also included in the Build Summary Report. For details, see "View build reports" on page 56.

Appendix A: Supported systems

This appendix details the systems supported by ALI.

This appendix includes:

Supported SCM systems

ALI supports the following SCM systems:

System	Versions
Subversion	1.7.x, 1.8.x (Tested on 1.7.1, 1.7.3, 1.8.3)
Concurrent Versions System (CVS)	1.11.x, 1.12.x (Tested on 1.11.23, 1.12.13)
Microsoft Team Foundation Server (TFS)	2012, 2013 (Tested on TFS 2012, 2013)
Perforce	2013.1, 2014.2 (Tested on 2013.1, 2014.2)
Git	1.8.x, 2.1.x (Tested on 1.8.5, 2.1.2, Linux only)

The SCM agents support deployment to the following operating systems:

Red Hat Enterprise Linux 6.x (32bit, 64bit)
SuSE Linux Enterprise 11.x (32bit, 64bit)
Windows 2008 R2 Server (64bit)
Windows 2012 Server (64bit)
Windows 2012 R2 Server (64bit)

Prerequisites:

MS PowerShell 2.0 or later must be installed on Windows and enabled to run scripts. Linux/Unix agent scripts use BASH. The BASH version must be 4.0 or later to support pushing into multiple repositories.

Limitations:

Git: ALI does not support the Git functionality of changing the history of commits. If commit history is changed in a Git repository, you must reload the entire commit history tracked by ALI for all affected branches. To reload the commit history, remove these branches and add them again.

Supported build systems

ALI supports these build systems:

System	Versions
Jenkins - Long-Term Support releases	1.532.1, 1.565.3 (Tested on 1.532.1, 1.565.3)
Hudson - Production Versions	3.1.0, 3.2.1 (Tested on 3.1.0, 3.2.1)
Microsoft Team Foundation Server (TFS)	2012, 2013 (Tested on TFS 2012, 2013)

ALI provides plug-ins for build servers. The plug-ins automatically extract build information and metrics and save them to ALM.

ALI supports these development metric tools:

Tool	Versions
Unit Testing	• JUnit – Tested on versions bundled with supported build systems. • TestNG - Tested on Hudson 0.8, Jenkins 0.28, Jenkins 0.32 • NUnit - Tested on Hudson 0.10, Jenkins 0.14 + version of NUnit framework 2.5.10, TFS 2012, 2013 • Visual Studio Managed Unit Testing Framework (MSTest - part of VS 2012, 2013)
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) • Visual Studio 2012, 2013 Code Coverage

Supported Force.com versions

Force.com integration was tested on Force.com API version 22.0.