General

Helpful Links

This sections contains a list of useful websites and references. Some of the links require an HP Passport account.

ALM Supported Environments

Integrations Matrix ALM 12.00

HP Software Support Online

HP ALM Resources Site

ALM Functionality by Edition

HP Application Lifecycle Management is also available in several editions which provide subsets of ALM functionality—HP ALM Essentials Edition, HP Quality Center Enterprise Edition, and HP ALM Performance Center Edition.

HP ALM Edition	Description
HP ALM	A unified platform for managing and automating processes, activities, and assets for building, testing, delivering, and maintaining applications. It includes modules for requirements, test, defect, and development management, and overall release and project planning. HP ALM helps organizations drive consistent processes, shared best-practices, and asset sharing across projects.
HP ALM Essentials Edition	Provides a subset of HP ALM product functionality, and is designed to help smaller teams get up and running quickly. It provides core functionality for requirements management, test management, and defect management.
HP Quality Center Enterprise Edition	Resides on the same unified platform as HP ALM. It delivers core functionality for quality management. It supports building a quality center of excellence through tight integrations with HP Unified Functional Testing, HP Business Process Testing, and HP Sprinter.
HP ALMPerformance Center Edition	Functionality for the complete management, scheduling, running, and monitoring of performance test scripts. It resides on the same platform as HP ALM and integrates directly with HP ALM and HP LoadRunner.

The following table indicates the availability of ALM functionality according to edition. Further information on each function is provided below.

Functionality	HP ALM Edition	HP ALM Essentials Edition	HP Quality Center Enterprise Edition	HP ALM Performance Center Edition
"Licensing" on page 4	✓	✓	✓	✓
"Release Management" on page 4	✓	✓	✓	✓
"Project Planning and Tracking (PPT) Releases" on page 5	✓	✗	✗	✗

Functionality	HP ALM Edition	HP ALM Essentials Edition	HP Quality Center Enterprise Edition	HP ALM Performance Center Edition
"Multiple Requirement Types" on page 5	✓	✓	✓	✓
"Requirement to Requirement Traceability" on page 5	✓	✓	✓	✓
"Risk-Based Quality Management" on page 5	✓	✗	✓	✓
"Business Models Module" on page 5	✓	✗	✓	✗
"Test Authoring and Execution" on page 5	✓	✓	✓	✓ (partial)
"Test Resources" on page 5	✓	✓	✓	✓
"Test Configurations" on page 5	✓	✓	✓	✓
"HP Sprinter" on page 5	✓	✗	✓	✗
"Lab Management" on page 5	✓	✗	✗	✓
"Automatic Provisioning of Cloud Test Hosts" on page 6	✗	✗	✗	✓

Functionality	HP ALM Edition	HP ALM Essentials Edition	HP Quality Center Enterprise Edition	HP ALM Performance Center Edition
"Versioning" on page 6	✓	✗	✓	✓
"Baselining" on page 6	✓	✗	✓	✓
"Sharing Requirements and Tests" on page 6	✓	✗	✗	✓
"Sharing Defects" on page 6	✓	✗	✗	✗
"Cross Project Customization" on page 6	✓	✓	✗	✓
"Cross Project Reporting" on page 6	✓	✗	✗	✓
"Export Data to Excel" on page 6	✓	✓	✓	✓
"Business Process Testing" on page 6	✓	✗	✓	✗
"Extensions" on page 7	✓	✓ (partial)	✓ (partial)	✓ (partial)
"Upgrading Editions" on page 7	--	✓	✓	✓

Licensing

ALM licensing is determined according to your purchase agreement. The site administrator can manage and monitor the license usage from Site Administration. For more details, see the *HP Application Lifecycle Management Administrator Guide.*

Release Management

You organize and track your upcoming releases in the Releases module. Performance Center

Edition does not support release management. If you are working with Performance Center Edition, fields and commands related to cycles and releases are not available. For example, Target Cycle and Target Release are not available.

Project Planning and Tracking (PPT) Releases

ALM project planning and tracking (PPT) functionality enables quality assurance managers to track application readiness by defining goals for activities of an application release. PPT is available for ALM Edition only.

Multiple Requirement Types

You can assign each requirement in the Requirements module to a default ALM requirement type. In addition, you can customize the default types and create your own requirement types.

Requirement to Requirement Traceability

Requirements traceability defines a relationship between two or more requirements, assisting you when analyzing the impact of a change proposed in a specific requirement.

Risk-Based Quality Management

The risk-based quality management feature enables you to calculate at which level to test each requirement, based on the nature of the requirement and the resources you have available. In the ALM Essentials Edition, the Risk tab is unavailable in the Requirements module. In addition, any risk related fields are unavailable.

Business Models Module

The Business Models module enables you to import business process models from standard modeling tools, and test their quality in ALM. The Business Models module is not available in ALM Essentials Edition or Performance Center Edition.

Test Authoring and Execution

You can build test plans and design tests based on your project requirements, and then execute those tests to diagnose and resolve problems. ALM Performance Center Edition supports performance testing only.

Test Resources

The Test Resources module enables you to manage resources used by your tests.

Test Configurations

Test configurations enable testing of various use-cases, each time with a different set of data.

HP Sprinter

HP Sprinter provides enhanced manual testing functionality and a variety of tools to assist in the manual testing process. Sprinter is available for ALM Edition and Quality Center Enterprise Edition.

Lab Management

Lab Management allows users to manage the lab resources and systems they use for functional and performance testing in ALM. Lab Management is available for functional and performance testing with ALM Edition and for performance testing with Performance Center Edition.

For more details, refer to the *HP ALM Lab Management Guide*.

Automatic Provisioning of Cloud Test Hosts

Cloud provisioning is currently available only for provisioning of load generators.

For more details, refer to the *HP ALM Lab Management Guide*.

Versioning

Version control enables you to create and manage ALM entities while maintaining previous versions of those entities. ALM Essentials Edition does not support single entity versioning of your ALM projects.

Baselining

You can create a baseline to store a snapshot of multiple entities in your project, at a specific point in time. ALM Essentials Edition does not support baselining.

Sharing Requirements and Tests

The Libraries module enables you to create and compare baselines of requirements, tests, test resources, and business components. You can also reuse an existing set of entities by importing, synchronizing and comparing libraries across multiple projects.

Quality Center Enterprise Edition: You can use the Libraries module to create and compare baselines in a project. Quality Center Enterprise Edition does not include importing, synchronizing and comparing libraries across multiple projects.

Sharing Defects

You can share and synchronize defects across multiple ALM projects using the *HP ALM Synchronizer*. Defect sharing is available for ALM Edition only.

For more details, see the *HP ALM Synchronizer User Guide*, available from the HP Application Lifecycle Management Adds-ins page.

Cross Project Customization

Cross project customization enables you to work with template projects to standardize policies and procedures across projects in your organization. Cross project customization is not available for Quality Center Enterprise Edition.

For more details, see the *HP Application Lifecycle Management Administrator Guide*.

Cross Project Reporting

When you create graphs in ALM, you can combine and compare data from multiple projects. Cross project reporting is unavailable for some entities. Cross project reporting is not available for Quality Center Enterprise Edition.

Export Data to Excel

All editions enable exporting ALM data to Excel for reporting. Exporting functionality is unavailable for some entities.

Business Process Testing

Business Process Testing enables non-technical subject matter experts to build and work with business components in a script-free environment and to create application-quality business process tests. Business Process Testing is not available for ALM Essentials Edition or Performance Center Edition.

Each user with the proper permissions who logs into an ALM server with a Business Process Testing license uses up both a Business Process Testing license and an ALM license.

> **Note:** You can run test sets containing business process tests in the Test Lab module and you can also edit automated components in Unified Functional Testing, even if no Business Process Testing license is available in ALM.

For more information on Business Process Testing, refer to the *HP Business Process Testing User Guide*.

Extensions

ALM extensions provide added functionality to ALM. Various extensions are available, depending on the edition with which you are working. If you have a license for an ALM extension, you can utilize the added functionality by enabling the extension on a per project basis. For more details on enabling extensions, refer to the *HP Application Lifecycle Management Administrator Guide*.

To view the list of extensions available with ALM 12.00 or to download documentation for the extensions, access the HP ALM Add-ins Page. You can access the Add-ins page in ALM from **Help > Add-ins**.

Upgrading Editions

You can upgrade your current edition to another edition. For example, you can upgrade from HP ALM Essentials Edition 12.00 to HP ALM Edition 12.00.

For more information on upgrading, refer to the *HP Application Lifecycle Management Installation and Upgrade Guide*.

System Requirements

ALM Server-Side System Requirements

This section includes the server-side system requirements and configurations for installing HP ALM 12.00 on Windows and Linux.

ALM Server-Side Hardware Requirements

The following table includes hardware requirements for installing ALM on a server machine.

CPU	**Windows:** Quad Core AMD64 processor or equivalent x86-compatible processor **Linux:** Quad Core AMD 64 bit or equivalent x86 compatible processor
Memory (RAM)	**Minimum:** 8 GB
Free disk Space	**Minimum:** 8 GB

Recommended ALM Server-Side Environments

The following table includes recommended configurations for each operating system.

Operating System	Database Server	Web Server
Microsoft Windows Server 2008 R2 Enterprise SP1 64 Bit	Microsoft SQL Server 2008 R2 SP2	Microsoft IIS 7.5
Red Hat Enterprise Linux 6.4 64 Bit	Oracle 11.2.0.3	Apache 2.2

Supported ALM Server-Side Configurations

The following applications are supported for installing ALM on a server machine. You may use any combination of these applications. To ensure best performance and quick support resolutions, we recommend using the recommended environments described in the section "System Requirements " above.

<table>
<tr><td>Operating Systems</td><td>

Microsoft Windows Server 2008 R2 Enterprise SP1 64 Bit
Microsoft Windows Server 2012 Standard 64 Bit
Red Hat Enterprise Linux 6.3 64 Bit
Red Hat Enterprise Linux 6.4 64 Bit
Oracle Enterprise Linux 6.3 64 Bit
Oracle Enterprise Linux 6.4 64 Bit
SUSE Linux 11.0

Note:

It is strongly recommended that ALM be deployed only on 64 bit systems in Windows production environments.
Localized editions of ALM are supported only on Windows operating systems.
Oracle Enterprise Linux versions are supported provided they are compatible with the supported versions of Red Hat Linux.

</td></tr>
<tr><td>Database Servers</td><td>

Oracle 10.2.0.5
Oracle 11.2.0.3
Microsoft SQL Server 2008 R2 SP2
Microsoft SQL Server 2012 SP1

Note:

All database servers, unless otherwise stated, have been validated on 64 bit configurations.
HP ALM is certified to work with Transparent Data Encryption (TDE) for Microsoft and Oracle databases. Implementation of TDE does have an impact on system performance, for details reach out to the database vendor providing encryption.

</td></tr>
<tr><td>Applications Servers</td><td>The Application server functionality is now built in to the ALM platform and there is no need to install a third party application server (WebLogic, WebSphere, and JBoss) from version 11.50 and later. The HP ALM application server uses JDK7 (1.7) and supports 64 bit JVM.</td></tr>
</table>

Web Servers	• Apache 2.2 • Microsoft IIS 7.5 • Microsoft IIS 8.0 **Note:** • By default, application servers include HTTP servers. If you need additional functionality a web server can be added. • HP will certify the major and minor releases of Apache (e.g. Apache 2.2) while all minor-minor releases of Apache (e.g. version 2.2.x) will not undergo specific certifications as it is expected that minor-minor software releases from Apache will maintain full compatibility.
Virtual Environments	VMWare ESX/ESXi Server 5.0 and later **Note:** HP ALM is certified to work with VMWare ESX/ESXi. Due to the rapidly evolving architectures provided by virtualization vendors as long as the above stated vendor guarantees full compatibility of the virtualized environment to the HP ALM approved system requirements for physical hardware, HP ALM will function as designed.
Full Disk Encryption	Full disk encryption (FDE) is supported for all system components, including database, server, repository server, and client machines. Implementation of FDE, does have an impact on system performance. For details, contact the vendor that provides the encryption.

ALM Client System Requirements

This section describes the client system requirements for installing ALM on client machines.

CPU	Dual Core 1.6 Ghz (or higher) or equivalent compatible
Memory (RAM)	**Minimum:** 2 GB
Free Disk Space	**Minimum:** 2 GB

Recommended ALM Client Environments

The following section includes recommended configurations for each operating system.

Operating System	Browsers	Office Suites
Microsoft Windows 7 SP1 32 Bit	Microsoft Internet Explorer 10	Microsoft Office 2010 SP2 32 Bit

ALM Desktop Client Supported Environments

This section describes the ALM desktop client supported environments.

Prerequisites	• Microsoft .NET Framework 4.0 / Microsoft .NET Framework 4.5 • Microsoft Office 2010 / Microsoft Office 2013
Operating System	• Microsoft Windows 7 SP1 32 Bit • Microsoft Windows 7 SP1 64 Bit • Microsoft Windows 8 32 Bit (Requires Microsoft Fix 449677. For more details, see http://support.microsoft.com/kb/2870007. • Microsoft Windows 8 64 Bit (Requires Microsoft Fix 449677. For more details, see http://support.microsoft.com/kb/2870007. • Microsoft Windows Server 2008 R2 SP1 64 Bit **Note:** • If you are integrating Quality Center with other HP testing tools, you must modify the DCOM permissions on your client machine. • DCOM is not required for running Functional test sets (server-side execution). • Functional test sets are available only in ALM Edition. • HP ALM Client on Windows 64 bit runs in WOW64 mode. • You can work with the HP ALM desktop client using a remote desktop.

Browsers	• Microsoft Internet Explorer 8 • Microsoft Internet Explorer 9 • Microsoft Internet Explorer 10 **Note:** For customers who have restrictions on plugins in their browsers, such as ActiveX, HP ALM can be loaded in the HP ALM Explorer Add-In. For more information on downloading and installing the Add-In, see the *HP Application Lifecycle Management Installation and Upgrade Guide.*
Office Suites	• Microsoft Office 2010 SP2 32 Bit • Microsoft Office 2013 32 Bit **Note:** Only 32 bit versions of Microsoft Office are supported.
Virtual Environments	• Citrix XenApp 6.0 or higher • Microsoft Terminal Services • Remote Desktop Services **Note:** HP ALM is certified to work with Citrix XenApp, Microsoft Terminal Services and Remote Desktop Services. Due to the rapidly evolving architectures provided by Virtualization vendors as long as the above stated vendor guarantees full compatibility of the virtualized environment to the HP ALM approved system requirements for physical hardware, then HP ALM will function as designed.
Other Configuration Settings	• **Screen Resolution (Minimum):** 1024x768 • **DPI Setting:** 100%

ALM Web Client Supported Environments

This section describes the ALM Web client supported environments.

Operating System	• Microsoft Windows 7 SP1 32 Bit • Microsoft Windows 7 SP1 64 Bit • Microsoft Windows 8 32 Bit • Microsoft Windows 8 64 Bit • Red Hat Enterprise Linux 6.3 64 Bit • Red Hat Enterprise Linux 6.4 64 Bit • Oracle Enterprise Linux 6.3 64 Bit • Oracle Enterprise Linux 6.4 64 Bit • Ubuntu 12.04 LTS • Apple Mac OS X (On Macbook Pro only)
Browsers	• Microsoft Internet Explorer 10 (On Microsoft Windows only) • Google Chrome 31 (On Microsoft Windows and Apple Mac OS only) • Mozilla Firefox 26 (On Microsoft Windows, Linux OS only) • Apple Safari 6.1 (On Mac OS only)
Virtual Environments	• Citrix XenApp 6.0 or higher • Microsoft Terminal Services • Remote Desktop Services **Note:** HP ALM is certified to work with Citrix XenApp, Microsoft Terminal Services and Remote Desktop Services. Due to the rapidly evolving architectures provided by Virtualization vendors as long as the above stated vendor guarantees full compatibility of the virtualized environment to the HP ALM approved system requirements for physical hardware, then HP ALM will function as designed

Other Configuration Settings	• **Screen Resolution:** 1024x768 (minimum); 1600x1050 (maximum) • **DPI Setting:** 100%,125%

HP ALM Lab Service for Remote Test Execution System Requirements

This section describes the system requirements for installing HP ALM Lab Service for remote test execution.

Processor	**CPU Type:** Intel Core, Pentium, AMD or compatible **Speed:** 2 GHz or higher recommended, 1 GHz minimum
Memory (RAM)	**Minimum:** 1GB **Note:** Memory needs depend on the number of add-ins.
Disk Space	**Minimum:** 1GB **Note:** You must also have an additional 120 MB of free space on the system disk.
Operating System	• Windows Server 2008 (SP2) 32 or 64 bit • Windows Server 2008 R2 (SP1) 64 bit • Windows 7 (SP1) 32 or 64 bit • Windows 8 32 or 64 bit • Windows Server 2012 64 bit
Operating Systems Supporting Auto Login	• Windows 7 (SP1) 32 or 64 bit • Windows Server 2008 R2 (SP1) 64 bit

HP ALM Performance Center System Requirements

This section describes the system requirements for installing ALM for Performance Center, and Performance Center components.

Performance Center Server Configurations

This section describes the system requirements for installing a Performance Center Server.

Processor	**CPU Type:**: Intel Core, Pentium, Xeon, AMD or compatible **Speed:** 2 GHz or higher recommended, 1 GHz minimum.
Memory (RAM)	**Minimum:** 4GB
Available Hard Disk Space	**Minimum:** 5 GB
Screen Resolution	**Minimum:** 1024 x 768
Operating System	• Windows Server 2008 R2 (SP1) 64 bit (Recommended) • Windows Server 2012 64 bit
Browser	Microsoft Internet Explorer 8, 9, or 10
Web Server	IIS 7.5 or 8.0

Recommended ALM and Performance Center Configurations

The following table describes the recommended configuration for Performance Center server components. They should be used along with the recommended ALM server configurations above.

Performance Center Server Operating System	Windows 2008 R2 (SP1) 64 bit
Host Operating System	Windows 2008 R2 (SP1) 64 bit

Host and Windows Standalone Load Generator Configurations

This section describes the system requirements for installing a Performance Center Host and a standalone Load Generator on Windows.

Processor	**CPU Type:** Intel Core, Pentium, Xeon, AMD or compatible **Speed:**1.6 GHz minimum. 2 GHz or higher recommended **Note for Pentium Processors:** Intel Hyper-Threading technology is not supported. Hyper-Threading can be disabled in the BIOS. For more information, see: http://www.intel.com/support/processors/pentium4/sb/CS-017371.htm
Memory (RAM)	**Recommended:** 4 GB or higher **Minimum:** 2GB

Available Hard Disk Space	**Minimum:** 40 GB
Screen Resolution	**Minimum:** 1024 x 768
Operating System	• Windows Server 2008 R2 64-bit • Windows 7 SP1 32 and 64-bit • Windows 8 64-bit • Windows Server 2012 64 bit
Browser	Microsoft Internet Explorer 8, 9 or 10

Linux Standalone Load Generator Configurations

This section describes the supported operating systems and CPU types for installing a standalone Load Generator on Linux systems.

Processor	**CPU Type:** Intel Core, Pentium, Xeon, AMD or compatible
Operating System	• Red Hat Linux Enterprise Linux 5.8 or 6.0, 32 bit • Red Hat Linux Enterprise Linux 5.8 or 6.4, 64 bit • Ubuntu Server 10.04LTS, 32 or 64 bit • Ubuntu Server 12.04LTS, 32 or 64 bit • Oracle Enterprise Linux 5.0 RH, 32 or 64 bit • Oracle Enterprise Linux 6.0 RH, 6.0UEK, 32 bit • Oracle Enterprise Linux 6.0 RH, 6.4UEK, 64 bit • Amazon Linux AMI 2012.03 or later, 32 or 64 bit **Note:** Oracle Enterprise Linux versions are supported as long as they are compatible with the supported versions of Red Hat Linux.

Standalone VuGen and Standalone Analysis Configurations

This section describes the system requirements for installing Standalone VuGen and Standalone Analysis.

Processor	**CPU Type:** Intel Core, Pentium, Xeon, AMD or compatible **Speed:**2 GHz or higher recommended, 1.6 GHz minimum.
Memory (RAM)	**Recommended:** 4 GB or higher **Minimum:** 2GB
Available Hard Disk Space	**Minimum:**40 GB
Screen Resolution	**Minimum:** 1024 x 768
Operating System	• Windows 7 (SP1) 32 and 64-bit • Windows Server 2008 R2 (SP1) 64 bit • Windows 8 64 bit • Windows Server 2012 64 bit (On Standalone Analysis only)
Browser	Microsoft Internet Explorer 8, 9, or 10

MI Listener System Requirements

This following table describes the system requirements for the MI Listener.

Processor	**CPU Type:** Intel Core, Pentium, Xeon, AMD or compatible **Speed:** 2 GHz minimum, 4 GHz or higher recommended
Memory (RAM)	**Minimum:** 2GB **Note:**Memory needs depend on the number of Load Generators and Controllers connected to the MI Listener.
Operating System	• Windows Server 2008 R2 (SP1) 64 bit • Windows 7 (SP1) 32 and 64-bit • Windows 8 64-bit • Windows Server 2012 64 bit

Administration

Default ALM Parameters

You can set the following default site configuration parameters:

Parameter	Description
ADD_NEW_ USERS_FROM_ PROJECT **(formerly CUSTOM_ ENABLE_ USER_ ADMIN)**	If this parameter is set to **N**, you can add new ALM users from Site Administration (**Site Users** tab) only. If this parameter is set to **Y** (default), new ALM users can also be added from Project Customization. In the Project Users page, click **Add User**. The Add User to Project dialog box opens. If this parameter is set to **Y**, a **New** button is available for adding new ALM users.
ATTACH_ MAX_SIZE	The maximum size (in kilobytes) of an attachment that can be sent with email from ALM. If the attachment size is greater than the specified value, the email is sent without the attachment. By default, the maximum email attachment size is 3,000 KB.
ATTACH_ TOTAL_ MAX_SIZE	The maximum size (in kilobytes) of all attachments that can be sent with email from ALM, taking into account the value of "ATTACH_ MAX_SIZE" above. If the size of all the attachments is greater than the specified value, the email is sent without the attachments. By default, the maximum email attachment size is 10,000 KB.
AUTO_MAIL_ WITH_ ATTACHMENT **(formerly SAQ_MAIL_ WITH_ ATTACHMENT)**	If this parameter is set to **Y** (default), defect email is sent with attachments. This applies only if you select **Send mail automatically** in the **Site Projects** tab. **Note:** The former parameter name is supported for purposes of backward compatibility.
AUTO_MAIL_ WITH_ HISTORY **(formerly SAQ_MAIL_ WITH_ HISTORY)**	If this parameter is set to **Y** (default), defect email is sent with the history. This applies only if you select **Send mail automatically** in the **Site Projects** tab. **Note:** The former parameter name is supported for purposes of backward compatibility.

Parameter	Description
BASE_ REPOSITORY_ PATH	The base repository path. The ALM and Site Administration repositories are sub-folders of this repository. If you change this parameter value, new projects you create are stored in this location. After you change the value of this parameter, you must restart all servers in the cluster. The initial repository path is set during ALM server configuration.
COMMUNICATION_ SECURITY_ PASSPHRASE	Communication between HP ALM and other HP BTO applications is enabled after authentication by a Single Sign-On (SSO) token through REST API. This parameter contains the passphrase that ALM uses to encrypt the SSO token. The initial parameter value is the SSO communication security passphrase that was entered during the ALM server configuration. Changing the COMMUNICATION_SECURITY_PASSPHRASE parameter on ALM requires that the equivalent value on other servers will be updated as well, for example, on Performance Center server and Host machines.
CREATE_ HTTP_ SESSION	You can use this parameter if you are working with load balancing over a cluster of application servers. If the parameter is set to **Y**, ALM creates an HTTP session. This causes the load balancer to operate in sticky mode, meaning that after a request sent by a client is directed to a particular node in the cluster, all subsequent requests sent by that client are directed to the same node. By default, this parameter is set to **N**.
DISABLE_ VERBOSE_ ERROR_ MESSAGES	This parameter is a security feature that controls the level of detail that error messages display. If the parameter is set to **N**, the user can view system details connected to the error. By default, this parameter is set to **Y**.
EVENT_ LOG_PURGE_ PERIOD_ DAYS	**Performance Center:** The time interval in days that deletable events remain in the EVENT_LOG database table. By default, the value is set to 60. If you set the value to **-1**, the events period is unlimited.
LDAP_ SEARCH_ USER_ CRITERIA	A comma-separated list of ALM user properties to be used as LDAP search criteria, if the Domain Authentication property does not contain the user's distinguished name (DN). The order of the properties defines their priority if multiple results are found. The following are the possible values: **username**, **email**, **fullname**, **phone**, **description**.

Parameter	Description
LIBRARY_ FUSE	The parameter value indicates the base number for calculating the maximum number of entities for a library, in order to maintain optimal performance. By default, the value is 2500. Calculation is performed as follows: • Maximum number of tests in a library = LIBRARY_FUSE * 1 (2500 by default) • Maximum number of resources in a library = LIBRARY_FUSE * 0.25 (625 by default) • Maximum number of business components in a library = LIBRARY_FUSE * 0.25 (625 by default) A validation of this value is performed when you create baselines, import libraries, or synchronize libraries. **Business Process Testing:** We recommend you set this parameter to a high value, such as 300000. For information on a related parameter, see "REQUIREMENTS_ LIBRARY_FUSE" on page 42.
LICENSE_ ARCHIVE_ PERIOD	The time interval in days during which license usage is archived. License usage information before this period is removed from the archive. By default, the value is set to **365** days. If you set the value to **-1**, the license archive period is unlimited.
LOCK_ TIMEOUT	The maximum number of hours that ALM objects can remain locked. After this time, the lock is removed. By default, the value is set to **10** hours.
MAIL_ FORMAT	The format ALM uses to send email. By default, the format is set to **HTML**. To instruct ALM to send email as plain text, change the value to **Text**.
MAIL_ INTERVAL	The time interval in minutes for sending a defect email according to your mail configuration settings. By default, the value is set to 10 minutes. Note that this applies only if you select **Send mail automatically** in the **Site Projects** tab.
MAIL_ MESSAGE_ CHARSET	The character set used by ALM to send email to users. By default, the value is set to **UTF-8**.
MAIL_ PROTOCOL	Displays the mail service used to send email messages to users. To configure the mail protocol, use the **Settings** button.

Parameter	Description
MAIL_ SERVER_ HOST	Displays the server name used by the SMTP mail service. To configure the server name, use the **Settings** button. For information on a related parameter, see "MAIL_SERVER_PORT" on page 36.
MAIL_SHOW_ SITE_NAME	Indicates whether the site name is displayed in the subject of the mail. This parameter can be either a project parameter or a site parameter. If the parameter is defined in both site and project tables, the project value is considered. The default value is **N**.
REPORT_ QUERY_ RECORDS_ LIMIT	The maximum number of records that can be retrieved from the database for an Excel report. If you set the value to **-1**, the number is unlimited.
REPORT_ QUERY_ TIMEOUT	The maximum length of time in seconds that ALM waits for an SQL query for an Excel report to be executed. If the query takes longer than this time to be executed, the query is canceled.
RESTRICT_ SERVER_ FOLDERS	This parameter enables you to access restricted-access server directories using the OTA **ExtendedStorage.ServerPath** property. If this parameter does not exist, or is set to **Y**, you can only use the **ExtendedStorage.ServerPath** property to access the following directories: • the Site Administration (**SA**) directory • the root directory for a project • the **attach** subdirectory for a project • the **baseline** subdirectory for a project • the **checkouts** subdirectory for a project • the **components** subdirectory for a project • the **hist** subdirectory for a project • the **resources** subdirectory for a project • the **StyleSheets** subdirectory for a project • the **tests** subdirectory for a project If this parameter is set to **N**, you can access all server directories using the **ExtendedStorage.ServerPath** property.

Parameter	Description
SITE_ANALYSIS	If this parameter is set to **Y** (default), you can track ALM license usage over time from the **Site Analysis** tab. If this parameter is set to **N**, the Site Analysis tab is unavailable.
SUPPORT_TESTSET_END	If this parameter is set to **Y** (default), UFT closes automatically when a test set is finished running.
UPLOAD_RESULTS_AFTER_TEST_RUN	Test results are, by default, saved in ALM. When running an automated test from within ALM you can avoid saving large test results. Large test results might be produced when runnning UFT GUI tests, UFT API tests, or automated business process tests. **ALWAYS**. Default. Always save test results. **NEVER**. Never save test results. **ON_FAILURE**. Save test results if run failed.
WAIT_BEFORE_DISCONNECT	The time interval in minutes that the ALM client can be inactive before it is disconnected from the ALM server machine. Disconnecting the client enables the license to be used by another ALM user. By default, the value is set to **600** minutes. For performance reasons, it is recommended to set a value of at least 60 minutes. If you set the value to **-1**, ALM is not disconnected, regardless of how long the client is inactive. For more information, see "AUTO_LOGOUT_ON_SERVER_DISCONNECT" on page 24.

Optional ALM Parameters

You can add the following optional site configuration parameters:

Parameter (A – Z)	Description
ALLOW_MULTIPLE_VALUES	This parameter determines whether the **Allow Multiple Values** check box is visible in the Project Entities page in Project Customization. If this parameter is set to **N**, then the **Allow Multiple Values** check box is unavailable. If this parameter does not exist or is set to "Y", then the **Allow Multiple Values** check box is available.

Parameter (A – Z)	Description
ALLOW_UPDATE_ USER_ PROPERTIES_ FROM_ CUSTOMIZATION	User details are set in Site Administration. A project administrator cannot change details of project users in Project Customization. If this parameter is set to **Y**, the project administrator is able to change the details of project users in Project Customization. This option may cause a security risk, as it allows the project administrator to replace a user's e-mail address with his own. By doing so, the project administrator can then use the Forgot Password link to reset and change a user's password. If this parameter does not exist, or is set to **N**, only the user can change his details in Project Customization.
ALLOW_WEBUI_ ACCESS	This parameter determines whether to allow access to Web UI client. Access is allowed only to users who belong to at least one group with **Allow Web UI Project Access** permission. If this parameter is set to **Y** or is not defined, ALM can be accessed through a web-based client user interface. If this parameter is set to **N**, ALM can only be accessed through a desktop-based client user interface. The default is **Y**.
ASYNC_ MAIL_ENABLED	If this parameter is set to **Y** (default), emails are sent asynchronously. This means that the email is queued to be sent, and you can continue working. If an email is undeliverable: • An email notification is sent to you, if the mail server is up. • A warning is added to the Site Administration log. If this parameter is set to **N,** emails are sent synchronously—meaning, emails are sent immediately and you continue working only when the mail is sent successfully.

Parameter (A – Z)	Description
AUTO_LOGOUT_ ON_SERVER_ DISCONNECT	The ALM server can disconnect an ALM client session. This occurs if: • The site administrator disconnects the session. • The session is automatically disconnected, according to the inactivity time interval setting. For more information on setting a timeout, see "WAIT_BEFORE_DISCONNECT" on page 22. The ALM client machine displays a message, informing the user that the session has been disconnected. If this parameter is set to **Y**, the client machine also automatically performs logout actions and returns the user to the ALM Login window. This ensures that the user does not continue to work in a session which is no longer connected to the server. If this parameter is set to **N**, no logout action is performed on disconnect. **Note:** If this parameter is set to **Y** in external authentication mode, the client machine returns the user to the ALM Connect to project window.
AUTO_MAIL_ SUBJECT_ FORMAT **(formerly SAQFORMAT**	This parameter enables you to customize the subject line of defect email sent automatically to users. For example, you can define a subject line such as `Defect no. 4321 has changed` by providing the value `Defect no. ?BG_BUG_ID has changed` , where **Defect no.** and **has changed** are strings, and **BG_ BUG_ID** is an ALM field name. **Note:** The former parameter name is supported for purposes of backward compatibility.
AUTO_ MAIL_USER_ NOTIFICATION	This parameter enables you to prevent ALM sending automatic email notification to project administrators when users are assigned or removed from a project in Site Administration. If this parameter is set to **N**, then automatic notification is not sent to project administrators. If this parameter does not exist, is empty, or is set to **Y**, then automatic notification is sent.
BACKWARD_ SUPPORT_ ALL_DOMAINS_ PROJECTS	This parameter enables the use of DomainsList and ProjectsList properties for the purposes of backward compatibility. If this parameter is set to **Y**, then the DomainsList and ProjectsList properties are supported. If the parameter does not exist or is empty, the default value is **N**, and these properties are not supported.

Parameter (A – Z)	Description
BV_EXCEL_ REPORT_ MAX_ROWS	This parameter defines the maximum number of rows that can be included in a Business View Excel Report. The default value is **100000**.
CLEAN_ ORPHAN_ ANALYSIS_ DATA_ JOB_SLEEP_ INTERVAL	This parameter defines how often orphan analysis data is cleaned up off the file repository. The data is cleaned up by the Orphan Analysis Data File Cleanup job. Define a value in minutes where the minimum is **10080** (one week). The default value is **43200** (one month).
COPY_ CHANGES_ USER_FIELDS **(formerly COPY_PASTE_ CHANGES_ OWNER)**	This parameter enables you to specify that the user who copies a record is listed in the specified User List fields of the copy. The value of this parameter is a comma-separated list of User List fields. For example, set the value of the parameter to BG_DETECTED_BY. Assume defect 10 is detected by user Cecil_qc, and user Shelly_qc copies defect 10. ALM creates a copy of the defect with Shelly_qc as the user who detected the defect, not Cecil_qc.
CUSTOM_HELP_ MENU_LINK	This parameter enables you to add a custom entry to the Help menu that links to a URL address. For example, if you want to allow users local access to ALM product movies, you can save the movies on the server, and create a link to a movies index page. Use the following syntax to enter a parameter value: <link alias>;<URL>, where the values of both <link alias> and <URL> are surrounded by quotation marks, and separated by a semicolon. For example, set the value of the parameter to "MyBusiness - Online Help Page";"http://mybusiness/ALMHelp". The above example adds the MyBusiness - Online Help Page entry to the Help menu. Clicking the entry opens a custom Web page located at http://mybusiness/ALMHelp.

Parameter (A – Z)	Description
CUSTOM_ PREREQUISITES_ PAGE_URL	This parameter enables you to handle missing prerequisites during the deployment phase of starting the ALM client. The value of this parameter is either: • A valid URL to a page that contains links for downloading alternate prerequisites. • **NO_URL** or blank. The deployment phase opens the default URL. The default value is blank. **Note:** Setting a separate URL for each prerequisite is not supported. The page must contain information for all prerequisites.
DASHBOARD_ PAGE_ITEM_ LIMIT	By default, dashboard pages can include up to four graphs. This parameter enables you to set a different number of maximum graphs that dashboard pages can include. Increasing the number of graphs may reduce the system's performance.
DEBUG_ CLOSURE_ LOG_DOM_PROJ	This parameter enables the creation of a server log for debugging issues related to closure. This can be helpful when investigating missing or redundant results of closure queries, or suspected performance related issues. **Caution:** This parameter should not be enabled unless debugging is required by support for specific reasons. Delete this parameter if not in use for a specific debugging purpose. The closure log can be enabled per project, per domain, or per site. Use one of the following formats for the parameter value: • **Per project**: <domain name>;<project name> For example, `DEFAULT;project1` For more than one project in a domain, separate the projects with a semi-colon. For example, for two projects named project1 and project2 in the DEFAULT domain, use: `DEFAULT;project1; DEFAULT;project2` • **Per domain:** <domain name>;`___DEBUG_ALWAYS___` to indicate all projects in the specified domain For example, for the DEFAULT domain, use: `DEFAULT;___DEBUG_ ALWAYS___` For more than one domain, separate the domains with a semi-colon.

Parameter (A – Z)	Description
	• **Per site:** `___DEBUG_ALWAYS___`**;**`___DEBUG_ALWAYS___` to indicate all domains, and all projects in each domain **Note:** The `DEBUG_ALWAYS` value uses three underscores as a prefix and suffix. The log files are saved to the location for server log files, as specified in the Site Administration **Server** tab > **Client Log File Settings**. The log files are not purged automatically. You must delete the closure log files manually when the closure log is no longer required.
DISABLE_ ASYNC_ CUSTOMIZATION_ LOAD	This parameter affects customization loading by OTA. If this parameter is set to **N**, customization is loaded asynchronously. If this parameter is set to **Y**, customization is loaded synchronously. The default is **N**.
DISABLE_ COMMAND_ INTERFACE	If this parameter is set to **Y** (default), only users belonging to the TDAdmin group can use the OTA **Command** object. If it is set to **N**, any user can use it. If it is set to **ALL**, no users can use it.
DISABLE_ CONSOLE_ DEBUG_INFO	This parameter enables you to access the ALM debug info console page (access to it is disabled by default). The console, if enabled, is accessible using **http://<qcserver>/qcbin/tdservlet?method=debuginfo** or **http://<qcserver>/qcbin/debug**. If this parameter exists and is set to **N**, the debug info console page can be accessed.
DISABLE_ DEFAULT_ VALUES	If this parameter is set to **Y**, default values for certain entities (such as defects, tests, and test configurations) cannot be set per user per project.

Parameter (A – Z)	Description
DISABLE_ EXTENDED_ STORAGE	This parameter controls user access to the OTA **ExtendedStorage** object. This is a security feature that can be used to limit access to the file system of the project. If this parameter is set to **Y** (default), the ExtendedStorage object cannot be accessed from TDConnection. Users can access the object from a specific entity for read-only, but no changes can be made. If it is set to **N**, the ExtendedStorage object can be accessed by all users, from a specific entity or from TDConnection. **Note:** This parameter can impact how certain add-ins run, such as the HP Screen Recorder. For details, see the documentation for the individual add-ins, available from the HP Application Lifecycle Management Add-ins page (**Help > Add-ins**) or the HP Application Lifecycle Management Tools page (**Help > ALM Tools**).
DISABLE_GET_ CHILDREN_ LISTS_WITH_ VERSIONING	If this parameter is set to **Y**, this parameter disables certain performance enhancements that improve the speed at which various modules display data (such as entity trees). This parameter is relevant for version controlled projects. The default is **N**.
DISABLE_HTTP_ COMPRESSION	By default, the data transferred from the ALM server to clients is compressed to improve performance. If this parameter exists and is set to **Y**, the data compression is disabled.
DISABLE_ PASSWORD_ OTA_ ENCRYPTION	By default, the OTA **TDConnection.Password** property is encrypted. If this parameter exists and is set to **Y**, encryption for this property is disabled. **Note:** Setting this parameter has no effect on password encryption during transport to the server machine.
DISPLAY_ LAST_USER_ INFO	This parameter enables you to add additional security to the client ALM Login window. By default, ALM displays the last user login information (user name, domain and project). If this parameter is set to **N**, the last user login information is not saved on the client machine and is not displayed in the ALM Login window. To activate this parameter, you must log in to ALM, log out, and log in again. If this parameter is set to **Y** or does not exist, the last user information is displayed.

Parameter (A – Z)	Description
DOWNLOAD_ REST_ ATTACHMENTS	This parameter determines whether attachments are downloaded automatically in REST. If the value is **Y**, attachments are downloaded only after the browser prompts the user to open or save the attachment. If the value is **N**, the browser downloads the attachment automatically The default is **Y**.
EI_DELETE_ INTERVAL	This parameter defines the interval, in hours, between jobs that delete execution items from the database. If an execution item was marked as **Deleted** for more than 10 hours, it is deleted by the job. The default is 10.
ENABLE_ COLUMN_ VISIBILITY_ TRACKING	This parameter defines whether fields can be made invisible in the GUI. The available options are: • **Y**. Fields can be made invisible in the GUI. • **N**. Fields cannot be made invisible in the GUI By default, the parameter is set to **N**.
ENABLE_ CREATE_ DOCGEN	This parameter enables or disables the Document Generator menu item in the masthead menu. The available options are: • **Y**. Document Generator is enabled.. • **N**. Document Generator is disabled. By default, the parameter is set to **N**.
ENABLE_ CREATE_ DOCGEN_ FAVORITE	This parameter defines whether new project documents can be added to your favorites list. The available options are: • **Y**. Project documents can be added to the favorites list. • **N**. Project documents cannot be added to the favorites list. By default, the parameter is set to **N**.

Parameter (A – Z)	Description
ENABLE_ CREATE_ LEGACY_ EXCEL_ REPORT	This parameter defines whether Microsoft Excel reports can be created in the Analysis View module. The available options are: • **Y**. Reports can be created. • **N**. Reports cannot be created. The default setting is **N**.
ENABLE_ CREATE_ STANDARD_ REPORT	This parameter defines whether standard reports can be created in the Analysis View module. The available options are: • **Y**. Reports can be created. • **N**. Reports cannot be created. Only existing reports are available. The default setting is **N**.
ENABLE_ ENTITY_ SELECTION_ TREE_REQ_ MOVE_TO	By default, the MoveTo event is not triggered when navigating up/down in a tree. However, in Quality Center version 10.00, the MoveTo event was triggered when navigating up/down in a Requirements tree. For backward compatibility, this parameter enables you to define whether the MoveTo event is triggered in when navigating up/down in a Requirements tree. The available options are: • **Y**. The MoveTo event is triggered. • **N**. The MoveTo event is not triggered. The default setting is **N**.
ENABLE_JMX_ CONSOLE	By default, JMX Console is disabled. This parameter allows you to enabled JMX Console for debugging purposes. The available options are: • **Y**. JMX Console is enabled. • **N**. JMX Console is disabled. The default setting is **N**.

Parameter (A – Z)	Description
ENABLE_ OUTPUT_ SANITIZATION	This parameter determines whether REST output sanitization is available. The available options are: • **Y**. REST output sanitization is enabled. • **N**. REST output sanitization is disabled. The default setting is **Y**.
ENABLE_ PERFORMANCE_ MONITOR_BIRT_ REPORTS	By default, QC Sense reports are disabled. This parameter allows you to generate QC Sense reports for debugging purposes, The available options are: • **Y**. QC Sense reports are enabled. • **N**. QC Sense reports are disabled. The default setting is **N**.
ENABLE_XSRF_ VALIDATION	This parameter enables or disables the XSRF token validation sent from the client on request The available options are: • **Y**. The XSRF token validation is enabled. • **N**. The XSRF token validation is disabled. The default setting is **Y**.
ENTITY_LINK_ HOST	This parameter enables you to set the mail server host name used in the link to an entity when ALM mails an entity. By default, ALM uses the default host name specified during installation. For details, see MAIL_ SERVER_PORT.
ENTITY_LINK_ PORT	This parameter enables you to set the mail server port number used in the link to an entity when ALM mails an entity. By default, ALM uses the default port number specified during installation.
ENTITY_LINK_ SCHEMA	This parameter enables you to set the schema used in the link to an entity when ALM mails an entity.

Parameter (A – Z)	Description
ENTITY_TABLE_NAMES_FOR_FIXING_UPDATE_PERMISSIONS_INCONSISTENCY	This parameter determines which database tables are checked by the verifier to find potential permission inconsistencies that are then fixed by the repairer. Permission inconsistencies occur when a user group has permission to modify specific fields of an entity, but does not have permission to modify the entity itself.
EXTENDED_MEMO_FIELDS	This parameter extends the maximum number of memo-type user-defined fields per entity to 15 instead of five. The default value is **N**. To extend the number of memo-type fields, set the parameter to **Y**.
FAST_RECONNECT_MODE	This parameter defines options for reconnecting after a user session expires. This parameter is not valid for external authentication, since the user must always be certified when reconnecting. Values include: **0.** Disables the reconnect option that bypasses reload of customization if no major change has been made. Users must manually log out and log in again when the session expires. **100** (default)**.** Password authentication is required. The user must enter a password to reconnect and continue working in ALM. **200.** The user does not need to enter password information to reconnect to ALM. User authentication is performed using the current password. If the user's password has been changed since the last log in, the user cannot reconnect. The user must log out and log in again using the new password. **Note:** If the user has been removed from the ALM Users list, the user cannot reconnect.
FAVORITES_DEPTH	Defines the number of most recently used favorite views displayed on the **Favorites** menu. By default, ALM displays the four most recently used views on the menu. To hide the list of recently used views completely, set the parameter to **0**.
FETCH_LIMIT	To optimize performance, the maximum number of records retrieved and displayed in ALM grids is limited. This parameter enables you to change the default limit. If this parameter does not exist, the maximum number of records displayed is limited to 500. If the value of this parameter is set to **0**, all results are displayed. For information on a related parameter, see "GROUP_FETCH_LIMIT" on page 34.

Parameter (A – Z)	Description
FILE_ EXTENSION_ BLACK_LIST_ DOWNLOAD	The value is the semicolon delimited string with the file extensions that are not allowed to be downloaded via open attachments, REST API, or FTP Explorer. **Note:** • Imported or restored projects are not blocked even if their repository contains files of blocked file types. • Projects upgraded from previous versions still contain files of blocked file types that already exist in the repository. • During optimized repository migration, files of blocked file types are not blocked.
FILE_ EXTENSION_ BLACK_LIST_ UPLOAD	The value is the semicolon delimited string with the file extensions that are not allowed to be uploaded via open attachments, extended storage, REST API, or FTP Explorer. **Note:** • Imported or restored projects are not blocked even if their repository contains files of blocked file types. • Projects upgraded from previous versions still contain files of blocked file types that already exist in the repository. • During optimized repository migration, files of blocked file types are not blocked.
FORCE_LOGIN_ SSL_MODE	If this parameter is set to **Y**, only the login process is sent over SSL (HTTPS). All other communication is sent without SSL (using HTTP). **Note:** ALM must be configured to work with SSL. For information on a related parameter, see "LOGIN_SSL_PORT" on page 36.
FROM_EMAIL_ ADDRESS	If a user clicks the **Forgot Password** link in the ALM Login window, an email notification is sent to the user with a link to specify a new password. This parameter enables you to change the email address in the email **From** field.

Parameter (A – Z)	Description
FTP_PORT	The port number of the FTP service that enables you to browse ALM project repositories. . Recommended values are **21** or **2121**. **Note:** • Defining the parameter starts the FTP service on the specified port. • Reconfiguring the parameter restarts the FTP service. • Deleting the parameter stops the FTP service.
GET_COVERAGE_FROM_BL_FOR_PINNED_TESTSET	This parameter enables you to get coverage from the baseline (and not from current view) when selecting tests for a requirement to be added to a pinned test set. The default value is **N**.
GRAPH_RESULT_LIFESPAN	When a graph is shared via the **Share Analysis Item** command in the Analysis View module, the graph retrieves data from a cache. By default, the cache information is updated every 60 minutes. This parameter enables you to configure the frequency, in minutes, at which the cache is updated. The minimum value you can assign to the parameter is 5 minutes. The maximum value is 60 minutes.
GROUP_FETCH_LIMIT	To optimize performance, the maximum number of records retrieved and displayed per group when a **group by** filter is applied to ALM grids is limited. This parameter enables you to change the default limit. If this parameter does not exist, the maximum number of records displayed per group is limited to 100. If the value of this parameter is set to **0**, all results are displayed for each group. For information on a related parameter, see "FETCH_LIMIT" on page 32.

Parameter (A – Z)	Description
INHERIT_ MODULE_ ACCESS_ TO_VIEWS	Some modules are logically connected to views in other modules. For example, the Business Models module is related to business linkage in the Requirements module and the Test Plan module. Project managers can hide access to modules for certain users/groups. This parameter indicates whether such related views should be hidden if their "parent" modules are hidden. **Note:** This parameter affects only views related to the Business Models module. Default: N (meaning, related views are not hidden)
INPUT_ VALIDATION_ LOGPATH	This parameter contains the input validation framework log folder location. If the parameter is not defined, the input validation error log is created in **repository/log/InputValidation**.
INPUT_ VALIDATION_ MODE	This parameter controls the input validation framework mode. The available options are: • **OFF**. No input validation is performed. • **LOG**. Perform hidden validation. Do not throw exceptions but log all errors. • **ON**. Full input validation is performed. The default setting is **ON**.
LAB_MAX_DB_ HANDLERS	This parameter enables you to define the maximum number of connections to the database allowed for LAB_PROJECT. The default value is **100**.

Parameter (A – Z)	Description
LDAP_IMPORT_ATTRIBUTE_MASK	This parameter enables you to define a regular expression that can be used to distinguish between different values for an LDAP attribute when importing users from an LDAP directory. When importing users, ALM chooses a value for the attribute that matches the regular expression. The parameter should be of the format: **<LDAP attribute name> = <regular expression>**, where **<LDAP attribute name>** is the name of the LDAP attribute whose value you want to choose, and **<regular expression>** is a regular expression. This regular expression should conform to the standard Java syntax for regular expressions. For example, a parameter value `uid=^\D\w+$` would choose values for the LDAP attribute `uid` consisting of a non-digit followed by any number of word characters (letters, numerals or the underscore character).
LDAP_RESULT_SIZE_LIMIT	The maximum number of results that LDAP returns for a filtered query. The default value is **1000**. A value less than the recommended minimum value of **100** can slow LDAP imports and searches. A value greater than the recommended maximum value of **10000** can cause the server to run out of memory.
LDAP_TIMEOUT **(formerly DIRECTORY_TIME_LIMIT_CONSTRAINT)**	The length of time, in milliseconds, that ALM waits before canceling an LDAP operation. The time limit on LDAP operations prevents a situation where LDAP encounters a problem and causes ALM to wait indefinitely. The default timeout value is 10 minutes.
LOGIN_SSL_PORT	If the FORCE_LOGIN_SSL_MODE parameter exists and is set to **Y**, this parameter enables you to configure the port used for SSL login. By default, the value is 443. For information on a related parameter, see "FORCE_LOGIN_SSL_MODE" on page 33.
MAIL_SERVER_PORT	The SMTP server port used by ALM to send mail. By default, the value is set to 25. For information on a related parameter, see "MAIL_SERVER_HOST" on page 21.

Parameter (A – Z)	Description
MAX_ CONCURRENT_ REPORTS	This parameter defines the maximum number of project reports that can be generated concurrently. Define any value where the minimum is **1**. Where the maximum number has been reached, and you choose to generate an additional report, the report generation only begins once one of the original reports finishes generating. For example, if the maximum number is **5**, and you try generate an additional report, the sixth report only begins generating once one of the original five concludes generating. The default maximum is **3**.
MAX_GRAPH_ RESULT_DATA_ TABLE_VOLUME	This parameter enables you to change the maximum size of the graph results, as calculated by the number of cells in the graph's Grid View. The size of a cell is 8 bytes. The default maximum is **100** megabytes.
MAX_KPIS_PER_ MILESTONE	This parameter determines the maximum number of KPIs that can be defined per milestone. The default number is 30.
MAX_ MILESTONES_ PER_RELEASE	This parameter determines the maximum number of milestones that can be defined per release. The default number is 20.
MAX_PROJECTS_ IN_EXCEL_ REPORT	This parameter determine the maximum number of projects that can be selected fora Business Views Excel report.
MAX_QUERY_ LENGTH	This parameter enables you to change the maximum query length (number of letters). It includes the parameters and is valid for Oracle and for SQL. The default value is 1000000. Some reports are too large for the default limit query size to handle. In such cases, this parameter should be set to 100000000.
MAX_SCOPE_ ITEMS_PER_ RELEASE	This parameter determines the maximum number of scope items that can be defined per release. The default number is 20.
MAX_STEPS_IN_ PROJECT	This parameter limits the maximum number of steps that can be fetched in each request, to protect the server from an Out Of Memory error.

Parameter (A – Z)	Description
MAX_TEST_ INSTANCES	This parameter enables you to limit the number of test instances handled by the server per call. The default value is 1000000. If there is a call with more test instances than this value, an error message will appear and the calls will be rejected causing the operation to fail.
MAX_TESTS_IN_ TEST_SET	This parameter enables you to limit the number of tests in a test set for copy/paste or delete functions. The default value is 20000. If there is a test set with more tests than this value, an error message will appear and the functions will not be performed.
MAX_ THRESHOLD_ VALUES_ PER_KPI	This parameter determines the maximum number of threshold values that can be defined per KPI. The default number is 12. .
NLS_SEARCH_ LOCALE	The language used by the **Find Similar Defects** command to tokenize the defect summary. This parameter is needed only if the default locale on the server does not match the language in which the defect summary is written, in terms of whether spaces are used to separate words. The value should be a string value that matches a language code listed in ISO 639 (http://www.w3.org/WAI/ER/IG/ert/iso639.htm). For example, if the default locale is English and the text is in Japanese, which does not use spaces to separate words, set `NLS_SEARCH_ LOCALE=ja`. If this parameter is not defined or is invalid, the default locale of the server is used.
ORACLE_RAC_ SUPPORT	This parameter must be set to **Y** to enable RAC support on Oracle database servers.
ORPHAN_ ANALYSIS_ DATA_ FILE_INACTIVE_ PERIOD	The parameter defines how old an analysis data file must be in order to be considered an orphan. The file is considered to be an orphan if it's last modified property is older than the time frame you define as part of this parameter. Define a value in milliseconds where the minimum is **3600000** (one hour). The default value is **86400000** (one day).

Parameter (A – Z)	Description
OUTPUT_ SANITIZATION_ BY_CLIENT_TYPE	This parameter determines whether output sanitization is invoked by client type. The format is <client type name>=true/false;<client type name>=true/false;... The default value for any client type that is not defined scpecifically in this site parameter is true.
PASSWORD_ RESET_DISABLE	This parameter determines whether ALM users can reset their passwords using the **Forgot Password** link in the ALM Login window. If this parameter is not defined or if the parameter is set to 'N', users can reset their passwords using the **Forgot Password** link. If LDAP authentication is enabled, you must set this parameter to 'Y'.
PASSWORD_ RESET_ ELAPSED_ TIME	If a user clicks the **Forgot Password** link in the ALM Login window, by default 24 hours have to elapse before the same user can make another password reset request. This parameter allows you to change the length of time, in minutes, that has to elapse before a user can make another password reset request.
PASSWORD_ RESET_SERVER	If a user clicks the **Forgot Password link** in the ALM Login window, an email notification is sent to the user with a link to specify a new password. This parameter allows you to override the default URL, or parts of the URL, that is embedded in the reset link. Use one of the following syntaxes: • **<server>:<port>.** Overrides both the default server and port. • **<server>.** Overrides the default server. • **<port>.** Overrides the default port.
PASSWORD_ RESET_VALID_ PERIOD	If a user clicks the Forgot Password link in the ALM Login window, an email notification is sent to the user with a link to specify a new password. By default, the link is valid for 24 hours. This parameter allows you to change the length of time, in minutes, that the link is valid.
PPT_KPI_ FAILURES_ PERCENTAGE_ PER_RELEASE_ FUSE	By default, if 10% or more of the KPI calculations within the release fail, ALM aborts project planning and tracking calculations on a release and skips to the next release in a project. The default number is 60.

Parameter (A – Z)	Description
PPT_RECENTLY_USED_PROJECTS_THRESHOLD_MINUTES	This parameter sets a threshold for controlling which projects are included in PPT calculations. PPT calculations are not performed for projects to which no one has logged in for the number of minutes specified The default is 10080 minutes (7 days).
PROJECT_SELECTION_MAX_PROJECTS	By default, cross-project graphs can include up to six projects. This parameter enables you to set a different number of maximum projects that cross-project graphs can include. Increasing the number of projects may reduce the system's performance.
QC_SENSE_AUTHORIZATION_DISABLED	This parameter allows you to disable the authorization check for users that want access to QC Sense reports. The default value is **N** (meaning, the authorization check is enabled).
QC_SENSE_REPORTS_USERS	By default, only site administrator users have access to QC Sense reports. This parameter enables you to grant permissions to additional users who are not site administrators. Enter user names using the following syntax: <user1>;<user2>;<user3>
REPLACE_TITLE	This parameter enables you to change the names of ALM modules across all your projects. Rename one or more modules by entering the following parameter value: <original title1 [singular]>;<new title1 [singular]>; <original title1 [plural]>;<new title1 [plural]>; <original title2 [singular]>;<new title2 [singular]>;... For example, if you want to change the name of the Defects module to **Bugs**, and the Requirements module to **Goals**, enter the following: Defect;Bug;Defects;Bugs;Requirement;Goal; Requirements;Goals Renaming the Releases module does not change the module name in the following locations: • The **Releases** command in the Releases module menu bar. • The **New Release Folder** menu command and dialog box. • The **New Release** menu command and dialog box.

Parameter (A – Z)	Description
REPORT_MAX_ ALLOWED_SIZE	This parameter defines the estimated maximum number of pages that can be included in a project report. The default value is **1000**. **Note:** While a report it being generated, ALM can only estimate the number of pages. Therefore it cannot be guaranteed that the final generated report will not exceed this number of pages. To leave the maximum number of pages as unlimited, set the parameter to **-1**. However, this is not recommended as it may negatively impact on your server's performance.
REPORT_ RESULTS_ LIFESPAN	When a project report is shared via the **Share Analysis Item** command in the Analysis View module, the report retrieves data from a cache. By default, the cache information is updated every 60 minutes. This parameter enables you to configure the frequency, in minutes, at which the cache is updated. The minimum value you can assign to the parameter is 5 minutes. The maximum value is 60 minutes.
REPOSITORY_ GC_ DELAY_ CANDIDATE_ TIME	This parameter enables you to create a delay between the time of the cleaning processes of each project repository and the time the files without references are actually deleted. Set a value in days between **0** and **28**. If the parameter does not exist, ALM delays the deletion of obsolete files in the file system for seven days.
REPOSITORY_ GC_ JOB_PRIORITY	The parameter determines the speed at which the repository cleanup process is performed. Set a value between **0** (fastest) and **10** (slowest). If the parameter does not exist, the speed is set to **3**.
REPOSITORY_ GC_ PROJECT_ CLEANUP_ INTERVAL	This parameter defines the time interval in days between cleaning processes of each project repository. Set a value in days between **1** and **28**. If the parameter does not exist, ALM scans project repositories once every seven days.

Parameter (A – Z)	Description
REPOSITORY_MIGRATION_JOB_PRIORITY	The parameter determines the speed at which files are copied from the old to the new project repository. Set a value between **0** (fastest) and **10** (slowest). If the parameter does not exist, the speed is set to **3**.
REQUIREMENT_REVIEWED_FIELD_AUTOMATIC_UPDATE	If this parameter is set to **Y** (default), then any change to a requirement field automatically sets the **Reviewed (RQ_REQ_REVIEWED)** field to **Not Reviewed**. If it is set to **N**, then a change to a requirement field does not affect the value of the Reviewed field.
REQUIREMENTS_LIBRARY_FUSE	This parameter indicates the maximum number of requirements for a library in order to maintain optimal performance. The default value for this parameter is 3500. A validation of this value is performed when you create baselines, import libraries, or synchronize libraries. For information on a related parameter, see "LIBRARY_FUSE" on page 20.
REST_API_DEFAULT_PAGE_SIZE	The number of entities per page returned by default with each GET operation on a collection using the REST API (unless the API consumer specifies a different page size for retrieval). The default is 100 entities.
REST_API_HTTP_CACHE_ENABLED	This parameter enables REST API support for server side HTTP caching. If enabled, the server supports caching using an ETag for the following resources: • customization/entities • customization/relations • customization/used-lists • customization/users The default is **Y**.
REST_API_MAX_BULK_SIZE	The maximum number of entities allowed for a single bulk operation. A bulk operation over REST means to be able to POST, PUT, or DELETE a collection of entities of the same type. The default is 2000 entities.

Parameter (A – Z)	Description
REST_API_MAX_ENTITY_TREE_SIZE	REST API provides a tree sub-resource for hierarchical entities. It retrieves an entity collection sorted as an expanded tree. This parameter limits the maximum collection size retrieved in a single request. **Note:** The tree sub-resource does not support paging, and therefore the **REST_API_MAX_PAGE_SIZE** parameter is not relevant. The default is 100 entities. **Note:** Higher values affect performance.
REST_API_MAX_PAGE_SIZE	The maximum number of entities per page that can be returned with a single GET operation on a collection using the REST API. The default is 5000 entities.
REST_SESSION_MAX_IDLE_TIME	This parameter sets the REST API session maximum idle time (in minutes). The idle time is the amount of time a REST API session token remains valid if there is no activity in the session. After this period, the session (including the licenses and locks that it holds) expire. On the next call, the REST API recreates a new session. The default value is 60 minutes.
SA_MAX_DB_HANDLERS	This parameter enables you to define the maximum number of connections to the database allowed for the Site Administration schema. The default value is **50**. The value must be 50 or higher.
SD_RUN_AMOUNT	The amount of runs taken into account when calculating the suggested duration of a particular test set or build verification suite execution. The default value for this parameter is 10. The recommended threshold value is 100. A higher value could potentially affect performance.
SECURED_QC_URL	When ALM generates email, it includes a link to ALM in the email. If this parameter is set to **Y**, the ALM URL uses an SSL connection (starting with `https:`). If it is set to **N** (default), SSL is not used.

Parameter (A – Z)	Description
SEND_ EXCEPTION_ DEFAULT_TO	This parameter defines the default email address for sending an error report in ALM. For information on a related parameter, see "SEND_EXCEPTION_ ENABLED" below. For more information on sending error details, see the *HP Application Lifecycle Management User Guide*.
SEND_ EXCEPTION_ ENABLED	If this parameter is set to **Y**, the **Send Error Details** option is available on the **Help** menu in ALM, enabling you to send details of an error as attachments to an email. For information on a related parameter, see "SEND_EXCEPTION_ DEFAULT_TO" above.
SKIP_CLIENT_ PREREQUISITES_ CHECK	This parameter enables you to bypass the prerequisites check performed in the deployment phase of starting the ALM client. The default value is **N**. **Tip:** For similar functionality in MSI Generator, check the **Skip Prerequisites Check** checkbox.
SMTP_SSL_ SUPPORT	This parameter enables an SSL tunnel to the SMTP server. The default value is **N**.
SQL_QUERY_ VALIDATION_ BLACK_LIST	By default, ALM checks that SQL queries for an Excel reports do not include any of the following commands: **INSERT**, **DELETE**, **UPDATE**, **DROP**, **CREATE**, **COMMIT**, **ROLLBACK**, **ALTER**, **EXEC**, **EXECUTE**, **MERGE**, **GRANT**, **REVOKE**, **SET**, **INTO**, or **TRUNCATE**. This ensures that you do not inadvertently modify or delete records in the project database. You can modify which commands are on this list by adding this parameter. The parameter's value must be a comma-separated list of SQL commands that ALM should verify are not included in SQL queries for an Excel report. Note that this verification is not performed if the SQL_QUERY_ VALIDATION_ENABLED parameter exists and is set to **N**. For details, see "SQL_QUERY_VALIDATION_ENABLED" on the next page.

Parameter (A – Z)	Description
SQL_QUERY_VALIDATION_ENABLED	By default, ALM checks SQL queries in Excel reports to ensure that they are valid and do not alter the project database. If this parameter is set to **N**, this validation is not performed. If this parameter does not exist, is empty, or is set to **Y**, this validation is performed. For information on a related parameter, see "SQL_QUERY_VALIDATION_BLACK_LIST" on the previous page.
SSO_EXPIRATION_TIME	Expiration time of the LWSSO token (the authentication token of the REST API) in minutes. After this period of inactivity, the REST API consumer is required to re-authenticate. The default value is 60 minutes.
STATIC_CONTENT_MAX_CACHED_FILE_SIZE	The maximum file size in MB that can be cached in the Jetty static content cache. You can reduce the cache file size, or cancel the cache by setting the parameter value to **0**.
SUPPORT_TESTSET_END	If this parameter is set to "Y", the Automatic Runner dialog box can signal a remote agent when a test set execution starts and ends. The events are passed using the Remote Agent's **Set_Value** method.
SUSPEND_REPOSITORY_GC	This parameter relates to the project repository cleaning process.. The parameter stops the cleanup process on the entire site. Use this only in special circumstances. For example, to check if the cleanup process is impacting system performance. Define this parameter and set its value to **Y** to temporarily stop the project repository cleaning process. To restart the cleaning process, set the parameter to **N**.
SUSPEND_REPOSITORY_MIGRATION	This parameter relates to the project repository migration process. The parameter stops the migration process on the entire site. Use this only in special circumstances. For example, to back up projects, or to check if the migration is affecting system performance. Define this parameter and set its value to **Y** to temporarily stop the migration of project files. To restart the migration, set the parameter to **N**.
TEXT_ENCODING_BY_CLIENT_TYPE	This parameter determines whether text encoding of output sanitization is invoked by client type. The format is <client type name>=true/false;<client type name>=true/false;... The default value for any client type that is not defined scpecifically in this site parameter is false.

Parameter (A – Z)	Description
TEXT_SEARCH_ TIMEOUT	The length of time, in seconds, that ALM waits before canceling the operation of enabling and rebuilding the text search indexes. This operation is activated by clicking the **Enable/Rebuild Text Search** button in the Site Projects tab of the Site Administration. The default timeout value is 20 minutes.
UNIX_SERVER	If this parameter is set to **Y**, it enables direct file access from a testing tool on a Windows machine to a UNIX based repository. You must then add a new parameter for each directory on the UNIX server machine you want to be able to access externally and specify the corresponding Windows path, as follows: • **Parameter name** is `FOLDER_MAPPING_`*n* where *n* is an identifying number. For example, `FOLDER_MAPPING_1` • **Parameter value** is in the format *UNIXpath->Windowspath* For example, `/opt/Mercury/repository/qc/->\\netapp\qc\repository\` **Note:** This parameter applies to HP LoadRunner.
UPGRADE_ EXCEPTION_FILE	This parameter defines the location of the global exception file to be used when upgrading projects. This file defines exceptions for the ALM database user schema. By default, the **SchemaExceptions.xml** file is saved in the directory.
UPLOAD_ ATTACH_ MAX_SIZE	**Note:** This parameter does not affect uploads of attachments in integration tools. The default value is blank (meaning, all attachments are uploaded regardless of size).
VERIFY_REPORT_ FOLDER	This parameter determines where the verification report is saved when the project verification process completes. By default, the output is saved in **<ALM repository path>\sa\DomsInfo \MaintenanceData\out** on your ALM server machine.

Workflow

Reference for ALM Events

This section contains an alphabetical reference of the ALM event functions and subroutines. It includes the event name, description, syntax, type (Function or Sub), the value returned by a function, and the entities for which the event procedure is available.

The following event functions are available:

Function Name	When the Function is Triggered
"ActionCanExecute" on page 49	before performing a user action
"Attachment_CanDelete" on page 51	before deleting an attachment
"Attachment_CanOpen" on page 51	before opening an attachment
"Attachment_CanPost" on page 52	before updating an attachment
"CanAddTests" on page 53	before adding tests to a test set
"CanCustomize" on page 53	before opening Customization window
"CanDelete" on page 53	before deleting an object from the server
"CanLogin" on page 56	before a user logs in to the project
"CanLogout" on page 57	before a user logs out of the project
"CanPost" on page 57	before posting an object to the server
"CanRemoveTests" on page 59	before removing tests from a test set
"CanAddComponentsToTest" on page 52	before adding business components to a test of type Flow or Business-Process
"CanAddFlowsToTest" on page 52	before adding flows to a test of type Business-Process
"CanRemoveComponentsFromTest" on page 58	before removing business components from a test of type Flow or Business-Process
"CanRemoveFlowsFromTest" on page 59	before removing flows from a test of type Business-Process
"CanDeleteGroupsFromTest" on page 56	before deleting groups from a test of type Flow or Business-Process
"CanReImportModels" on page 58	before importing business models
"DefaultRes" on page 59	before resetting project defaults

Function Name	When the Function is Triggered
"FieldCanChange" on page 61	before changing a field value
"GetDetailsPageName" on page 64	before displaying Defect Details dialog box
"GetNewBugPageName" on page 64	before displaying Add Defect dialog box (for backward compatibility)
"GetNewReqPageName" on page 65	before displaying New Requirement dialog box (for backward compatibility)
"GetReqDetailsPageName" on page 65	before displaying Requirement Details dialog box (for backward compatibility)

The following event subroutines are available:

Subroutine Name	When the Subroutine is Triggered
"AddComponentToTest" on the next page	a component has been added to a test of type Flow or Business-Process
"AfterPost" on page 50	an object has been posted to the server
"Attachment_New" on page 52	an attachment is added
"DialogBox" on page 60	a dialog box is opened or closed
"EnterModule" on page 60	user switches modules
"ExitModule" on page 60	user exits a module
"FieldChange" on page 62	a field value changes
"MoveTo" on page 66	user changes focus
"MoveToComponentFolder" on page 68	user moves to the specified component folder in the business component tree (for backward compatibility)
"MoveToFolder" on page 68	user clicks a folder in the test sets tree (for backward compatibility)
"MoveToSubject" on page 68	user clicks a subject in the test plan tree (for backward compatibility)
"New" on page 69	an object is added
"RemoveComponentFromTest" on page 70	user removes a component from a test of type Flow or Business-Process
"RunTests" on page 70	user clicks **Run** in the Test Lab module (provided that Sprinter is not installed and none of the tests is automated)

Subroutine Name	When the Subroutine is Triggered
"RunTests_Sprinter" on page 70	user clicks **Run** in the Test Lab module (provided that Sprinter is installed and at least one test is automated)
"RunTestSet" on page 71	user clicks **RunTest Set** in the Test Lab module
"RunTestsManually" on page 71	user clicks **Run** > **Run Manually** in the Test Lab module

ActionCanExecute

This event is triggered before ALM performs an action that has been initiated by the user, to check whether the action can be executed.

You can add code to this event procedure to perform actions when the user has initiated a particular action, or to prevent the action from being executed in specific cases. For example, see "Example: Controlling User Permissions" on page 96.

Syntax	**ActionCanExecute(ActionName)** where **ActionName** is the action that the user has initiated. Actions are in the format **context.action**. **Note:** The previous format for this event is supported for purposes of backward compatibility. We recommend you use ActionCanExecute instead. User-defined actions start with the prefix `UserDefinedActions`.
Type	Function
Returns	True or False
Availability	`ActionCanExecute` (all modules)

Tip: To obtain the name of an action, see the sample code on "Action Object" on page 73.

AddComponentToTest

This event is triggered when the user adds a component to a test of type Flow or Business-Process in the Test Script tab.

Version Control: Changing components checked in or checked out by another user, using the AddComponentToTest event, is not supported.

Syntax	**AddComponentToTest**
Type	Sub
Availability	`AddComponentToTest`

AfterPost

This event is triggered after an object has been posted to the server.

Project fields should not be changed after they have been posted, because then the new value is not stored in the database.

Syntax	**<entity>_AfterPost**
Type	Sub
Availability	• `AnalysisItem_AfterPost` • `AnalysisItemFolder_AfterPost` • `Baseline_AfterPost` • `Bug_AfterPost` • `BusinessModel_AfterPost` • `BusinessModelFolder_AfterPost` • `BusinessModelPath_AfterPost` • `Component_AfterPost` • `ComponentFolder_AfterPost` • `Cycle_AfterPost` • `DashboardFolder_AfterPost` • `DashboardPage_AfterPost` • `Library_AfterPost` • `LibraryFolder_AfterPost` • `Release_AfterPost` • `ReleaseFolder_AfterPost` • `Req_AfterPost` • `Resource_AfterPost` • `ResourceFolder_AfterPost` • `Run_AfterPost` • `Step_AfterPost`

	• `Test_AfterPost` • `TestConfiguration_AfterPost` • `TestFolder_AfterPost` • `TestSet_AfterPost` • `TestSetFolder_AfterPost`

Attachment_CanDelete

This event is triggered before ALM deletes an attachment from the server, to check whether that attachment can be deleted.

Syntax	**Attachment_CanDelete(Attachment)** where **Attachment** is the **IAttachment** interface. For more information, refer to the *HP ALM Open Test Architecture API Reference*.
Type	Function
Returns	True or False
Availability	`Attachment_CanDelete` (all modules)

Attachment_CanOpen

This event is triggered before ALM opens an attachment from the server, to check whether the attachment can be opened.

Syntax	**Attachment_CanOpen(Attachment)** where **Attachment** is the **IAttachment** interface. For more information, refer to the *HP ALM Open Test Architecture API Reference*.
Type	Function
Returns	True or False
Availability	`Attachment_CanOpen` (all modules)

Attachment_CanPost

This event is triggered before ALM updates an existing attachment on the server, to check whether the attachment can be updated.

Syntax	**Attachment_CanPost(Attachment)** where **Attachment** is the **IAttachment** interface. For more information, refer to the *HP ALM Open Test Architecture API Reference*.
Type	Function
Returns	True or False
Availability	`Attachment_CanPost` (all modules)

Attachment_New

This event is triggered when an attachment is added to ALM.

Syntax	**Attachment_New(Attachment)** where **Attachment** is the **IAttachment** interface. For more information, refer to the *HP ALM Open Test Architecture API Reference*.
Type	Sub
Availability	`Attachment_New` (all modules)

CanAddComponentsToTest

This event is triggered before ALM adds business components to a test of type Flow or Business-Process, to check whether the specified components can be added.

Syntax	**CanAddComponentsToTest(Components)** where **Components** is an array of component IDs.
Type	Function
Returns	True or False
Availability	`CanAddComponentsToTest`

CanAddFlowsToTest

This event is triggered before ALM adds flows to a test of type Business-Process, to check whether the specified flows can be added.

Syntax	**CanAddFlowsToTest(Flows)** where **Flows** is an array of flow IDs.
Type	Function
Returns	True or False
Availability	`CanAddFlowstoTest`

CanAddTests

This event is triggered before ALM adds tests to a test set, to check whether the specified tests can be added.

Syntax	**<entity>_CanAddTests(Tests)** where **Tests** is an array of Test IDs.
Type	Function
Returns	True or False
Availability	`TestSet_CanAddTests`

CanCustomize

This event is triggered when a user attempts to open the Customization window, to check whether the specified user can customize the specified project.

Syntax	**CanCustomize(DomainName, ProjectName, UserName)** where **DomainName** is the domain name, **ProjectName** is the project name, and **UserName** is the user name.
Type	Function
Returns	True or False
Availability	`CanCustomize` (all modules)

CanDelete

This event is triggered before ALM deletes an object from the server, to check if the object can be deleted.

Syntax	**<entity>_CanDelete(Entity)**
Type	Function

Returns	True or False
Availability	• `AnalysisItem_CanDelete` • `AnalysisItemFolder_CanDelete` • `Baseline_CanDelete` • `Bug_CanDelete` • `BusinessModel_CanDelete` • `BusinessModelFolder_CanDelete` • `BusinessModelPath_CanDelete` • `Component_CanDelete` • `ComponentFolder_CanDelete` • `Cycle_CanDelete` • `DashboardFolder_CanDelete` • `DashboardPage_CanDelete` • `Library_CanDelete` • `LibraryFolder_CanDelete` • `Release_CanDelete` • `ReleaseFolder_CanDelete` • `Req_CanDelete` • `Resource_CanDelete` • `ResourceFolder_CanDelete` • `Test_CanDelete` • `TestConfiguration_CanDelete` • `TestFolder_CanDelete` • `TestSet_CanDelete` • `TestSetFolder_CanDelete`

Additional Syntax for Backward Compatibility

For purposes of backward compatibility, the following syntaxes are also available for certain objects. However, we recommend you use CanDelete instead.

- The syntax for tests or test subject folders:

Syntax	**Test_CanDelete(Entity, IsTest)** where: ▪ **Entity** is the test or subject folder. ▪ If **IsTest** is `True`, **Entity** refers to an ITest object. If **IsTest** is `False`, **Entity** refers to an ISubjectNode object. For more information on ITest and ISubjectNode, refer to the *HP ALM Open Test Architecture API Reference*.
Type	Function
Returns	True or False
Availability	`Test_CanDelete`

- The syntax for test sets or test set folders:

Syntax	**TestSet_CanDelete(Entity, IsTestSet)** where: ▪ **Entity** is the test set or test set folder. ▪ If **IsTestSet** is `True`, **Entity** refers to an ITestSet object. If **IsTestSet** is `False`, **Entity** refers to an ITestSetFolder object. For more information on ITestSet and ITestSetFolder, refer to the *HP ALM Open Test Architecture API Reference*.
Type	Function
Returns	True or False
Availability	`TestSet_CanDelete`

- The syntax for business components or business component folders:

Syntax	**Component_CanDelete(Entity, IsComponent)** where: ▪ **Entity** is the component or component folder. ▪ If **IsComponent** is `True`, **Entity** refers to an IComponent object. If **IsComponent** is `False`, **Entity** refers to an IComponentFolder object. For more information on IComponent and IComponentFolder, refer to the *HP ALM Open Test Architecture API Reference*.
Type	Function
Returns	True or False
Availability	`Component_CanDelete`

CanDeleteGroupsFromTest

This event is triggered when a user removes groups from a test of type Flow or Business-Process, to check whether the specified groups can be removed.

Syntax	**CanDeleteGroupsFromTest(Groups)** where **Groups** is an array of group IDs.
Type	Function
Returns	True or False
Availability	`CanDeleteGroupsFromTest`

CanLogin

This event is triggered to check whether the specified user can log in to the specified project.

Syntax	**CanLogin(DomainName, ProjectName, UserName)** where **DomainName** is the domain name, **ProjectName** is the project name, and **UserName** is the user name.
Type	Function
Returns	True or False
Availability	`CanLogin` (all modules)

CanLogout

This event is triggered to check whether the current user can log out of the current project.

Syntax	**CanLogout**
Type	Function
Returns	True or False
Availability	`CanLogout` (all modules)

CanPost

This event is triggered before ALM posts an object to the server, to check whether the object can be posted.

You can add code to this event procedure to prevent an object from being posted in specific cases. For example, see "Example: Object Validation" on page 93.

Syntax	**<entity>_CanPost**
Type	Function
Returns	True or False
Availability	• `AnalysisItem_CanPost` • `AnalysisItemFolder_CanPost` • `Baseline_CanPost` • `Bug_CanPost` • `BusinessModel_CanPost` • `BusinessModelFolder_CanPost` • `BusinessModelPath_CanPost` • `Component_CanPost` • `ComponentFolder_CanPost` • `Cycle_CanPost` • `DashboardFolder_CanPost` • `DashboardPage_CanPost` • `Library_CanPost`

	• `LibraryFolder_CanPost` • `Release_CanPost` • `ReleaseFolder_CanPost` • `Req_CanPost` • `Resource_CanPost` • `ResourceFolder_CanPost` • `Run_CanPost` • `Step_CanPost` • `Test_CanPost` • `TestConfiguration_CanPost` • `TestFolder_CanPost` • `TestSet_CanPost` • `TestSetFolder_CanPost` • `TestSetTests_CanPost` (does not appear in the Scripts Tree)

CanReImportModels

This event is triggered when attempting to import the specified business process models that already exist in ALM, to check if the business process models can be reimported.

Syntax	**<entity>_CanReImportModels(Models)** where **Models** is an array of Model IDs.
Type	Function
Returns	True or False
Availability	`CanReImportModels`

CanRemoveComponentsFromTest

This event is triggered when a user removes components from a test of type Flow or Business-Process, to check whether the specified components can be removed.

Syntax	**CanRemoveComponentsFromTest(Components)** where **Components** is an array of component IDs.
Type	Function
Returns	True or False
Availability	`CanRemoveComponentsFromTest`

CanRemoveFlowsFromTest

This event is triggered when a user removes flows from a test of type Business-Process, to check whether the specified flows can be removed.

Syntax	**CanRemoveFlowsFromTest(Flows)** where **Flows** is an array of flow IDs.
Type	Function
Returns	True or False
Availability	`CanRemoveFlowsFromTest`

CanRemoveTests

This event is triggered to check whether the specified tests can be removed from a test set.

Syntax	**<entity>_CanRemoveTests(Tests)** where **Tests** is an array of Test Instance IDs.
Type	Function
Returns	True or False
Availability	`TestSet_CanRemoveTests`

DefaultRes

This function is used to determine the default return value for ALM functions, such as FieldCanChange. All ALM workflow functions call this function (unless explicitly omitted by user) to determine the default return value. DefaultRes can be used to quickly replace the default return values of all ALM workflow functions.

Syntax	**DefaultRes**
Type	Function

Returns	True or False
Availability	`DefaultRes` (all modules)

DialogBox

This event is triggered when a dialog box is opened or closed.

Syntax	**DialogBox(DialogBoxName, IsOpen)** where **DialogBoxName** is the name of the dialog box, and **IsOpen** indicates whether the dialog box is open.
Type	Sub
Availability	`DialogBox` (all modules)

Note: For purposes of backward compatibility, this event is also triggered using backward compatible values for defect details (**DialogBoxName="Details"**) and test instance details (**DialogBoxName="TestInstanceDetails"**). These backward compatible values are not recommended.

EnterModule

This event is triggered when the user enters or switches to an ALM module. It is also triggered when the user logs in to ALM.

You can add code to this event procedure to perform an action whenever the user switches to the specified module.

Syntax	**EnterModule**
Type	Sub
Availability	`EnterModule` (all modules)

ExitModule

This event is triggered when the user exits the specified module.

Syntax	**ExitModule**
Type	Sub
Availability	`ExitModule` (all modules)

FieldCanChange

This event is triggered before ALM changes a field value, to determine whether the field can be changed.

You can add code to this event procedure to prevent a field from being changed in specific cases. For example, see "Example: Field Validation" on page 94.

Syntax	**<entity>_FieldCanChange(FieldName, NewValue)** where **FieldName** is the name of the field and **NewValue** is the field value.
Type	Function
Returns	True or False
Availability	• `AnalysisItem_FieldCanChange` • `AnalysisItemFolder_FieldCanChange` • `Baseline_FieldCanChange` • `Bug_FieldCanChange` • `BusinessModel_FieldCanChange` • `BusinessModelActivity_FieldCanChange` • `BusinessModelFolder_FieldCanChange` • `BusinessModelPath_FieldCanChange` • `Component_FieldCanChange` • `ComponentFolder_FieldCanChange` • `ComponentStep_FieldCanChange` • `Cycle_FieldCanChange` • `DashboardFolder_FieldCanChange` • `DashboardPage_FieldCanChange` • `DesignStep_FieldCanChange` • `Library_FieldCanChange` • `LibraryFolder_FieldCanChange` • `Release_FieldCanChange`

	• `ReleaseFolder_FieldCanChange` • `Req_FieldCanChange` • `Resource_FieldCanChange` • `ResourceFolder_FieldCanChange` • `Run_FieldCanChange` • `Step_FieldCanChange` • `Test_FieldCanChange` • `TestConfiguration_FieldCanChange` • `TestFolder_FieldCanChange` • `TestSet_FieldCanChange` • `TestSetFolder_FieldCanChange` • `TestSetTests_FieldCanChange`

The code for hiding a field that depends on another field should be placed in the FieldChange event procedure (not in the FieldCanChange event procedure).

FieldChange

This event is triggered when the value of the specified field changes.

Every change of value triggers the field change event when the field loses focus.

You can add code to this event procedure to perform an action when the value of a particular field is changed. For example, you can hide or display one field depending on the value the user enters into another field. For example, see "Example: Changing One Field Based on Another Field" on page 92.

Syntax	**<entity>_FieldChange(FieldName)** where **FieldName** is the name of the field.
Type	Sub
Availability	• `AnalysisItem_FieldChange` • `AnalysisItemFolder_FieldChange` • `Baseline_FieldChange` • `Bug_FieldChange` • `BusinessModel_FieldChange`

	• `BusinessModelActivity_FieldChange` • `BusinessModelFolder_FieldChange` • `BusinessModelPath_FieldChange` • `Component_FieldChange` • `ComponentFolder_FieldChange` • `ComponentStep_FieldChange` • `Cycle_FieldChange` • `DashboardFolder_FieldChange` • `DashboardPage_FieldChange` • `DesignStep_FieldChange` • `Library_FieldChange` • `LibraryFolder_FieldChange` • `Release_FieldChange` • `ReleaseFolder_FieldChange` • `Req_FieldChange` • `Resource_FieldChange` • `ResourceFolder_FieldChange` • `Run_FieldChange` • `Step_FieldChange` • `Test_FieldChange` • `TestConfiguration_FieldChange` • `TestFolder_FieldChange` • `TestSet_FieldChange` • `TestSetFolder_FieldChange` • `TestSetTests_FieldChange`

When a user changes a field value using the **Find/Replace** command, workflow events are not triggered. If restrictions implemented in workflow scripts are critical, consider disabling the **Replace** command for specific user groups, to ensure that your restrictions cannot be bypassed.

GetDetailsPageName

This event is triggered by ALM to retrieve the name of the page (tab) that has the index number specified in **PageNum** in the following dialog boxes:

- The Details dialog box for an entity
- The New <entity> dialog box for an entity

You can add code to this event procedure to customize the tab names for the Details dialog box. For example, see "Example: Changing Tab Names" on page 91.

Syntax	**GetDetailsPageName(PageName, PageNum)** where **PageName** is the default page (tab) name (for example, `Page 1`) and **PageNum** is the page (tab) number. **Note:** The page number is the absolute page number, regardless of the page's relative position in relation to the other displayed pages in the dialog box.
Type	Function
Returns	String containing the page name
Availability	`GetDetailsPageName` (all modules)

GetNewBugPageName

This event is triggered by ALM to retrieve the name of the New Defect dialog box page (tab) that has the index number specified in PageNum.

You can add code to this event procedure to customize the tab names on the New Defect dialog box. For example, see "Example: Changing Tab Names" on page 91.

Syntax	**GetNewBugPageName(PageName, PageNum)** where **PageName** is the default page (tab) name (for example, `Page 1`) and **PageNum** is the page (tab) number. **Note:** The page number is the absolute page number, regardless of the page's relative position in relation to the other displayed pages in the New Defect dialog box.
Type	Function

Returns	String containing the page (tab) name
Availability	`GetNewBugPageName`

Note: The GetNewBugPageName event is not listed in the Scripts Tree of the Script Editor. This event is triggered for backward compatibility purposes only. GetDetailsPageName should be used instead.

GetNewReqPageName

This event is triggered by ALM to retrieve the name of the New Requirement dialog box page (tab) that has the index number specified in PageNum.

You can add code to this event procedure to customize the tab names on the New Requirement dialog box. For example, see "Example: Changing Tab Names" on page 91.

Syntax	**GetNewReqPageName(PageName, PageNum)** where **PageName** is the default page (tab) name (for example, `Page 1`) and **PageNum** is the page (tab) number. **Note:** The page number is the absolute page number, regardless of the page's relative position in relation to the other displayed pages in the New Defect dialog box.
Type	Function
Returns	String containing the page name
Availability	`GetNewReqPageName`

Note: The GetNewReqPageName event is not listed in the Scripts Tree of the Script Editor. This event is triggered for backward compatibility purposes only. GetDetailsPageName should be used instead.

GetReqDetailsPageName

This event is triggered by ALM to retrieve the name of the Requirement Details dialog box page (tab) that has the index number specified in PageNum.

You can add code to this event procedure to customize the tab names on the Requirement Details dialog box. For example, see "Example: Changing Tab Names" on page 91.

Syntax	**GetReqDetailsPageName(PageName, PageNum)** where **PageName** is the default page (tab) name (for example, `Page 1`) and **PageNum** is the page (tab) number. **Note:** The page number is the absolute page number, regardless of the page's relative position in relation to the other displayed pages in the New Defect dialog box.
Type	Function
Returns	String containing the page name
Availability	`GetReqDetailsPageName`

Note: The GetReqDetailsPageName event is not listed in the Scripts Tree of the Script Editor. This event is triggered for backward compatibility purposes only. GetDetailsPageName should be used instead.

MoveTo

This event is triggered when the user changes focus from one object to another.

You can add code to this event procedure to perform actions when the user changes the focus. For example, see "Example: Presenting a Dynamic Field List" on page 95.

Tip: When moving from one object to another in a tree, the MoveTo event is not triggered. However, it is possible to trigger the event for Requirement trees. For details, see "ENABLE_ ENTITY_SELECTION_TREE_REQ_MOVE_TO" on page 30.

Syntax	**<entity>_MoveTo**
Type	Sub
Availability	• `AnalysisItem_MoveTo` • `AnalysisItemFolder_MoveTo` • `Baseline_MoveTo` • `Bug_MoveTo` • `BusinessModel_MoveTo` • `BusinessModelActivity_MoveTo` • `BusinessModelFolder_MoveTo`

	• BusinessModelPath_MoveTo • Component_MoveTo • ComponentFolder_MoveTo (formerly MoveToComponentFolder) • ComponentStep_MoveTo • Cycle_MoveTo • DashboardFolder_MoveTo • DashboardPage_MoveTo • DesignStep_MoveTo • Library_MoveTo • LibraryFolder_MoveTo • Release_MoveTo • ReleaseFolder_MoveTo • Req_MoveTo • Resource_MoveTo • ResourceFolder_MoveTo • Run_MoveTo • Step_MoveTo • Test_MoveTo • TestConfiguration_MoveTo • TestFolder_MoveTo • TestSet_MoveTo • TestSetFolder_MoveTo • TestSetTests_MoveTo

MoveToComponentFolder

This event is triggered when the user moves to the specified component folder in the business component tree.

Syntax	**MoveToComponentFolder(Folder)** where **Folder** is the **IComponentFolder** interface. For more information, refer to the *HP ALM Open Test Architecture API Reference*.
Type	Sub
Availability	`MoveToComponentFolder`

Note: The MoveToComponentFolder event is not listed in the Scripts Tree of the Script Editor. This event is supported for purposes of backward compatibility. We recommend you use ComponentFolder_MoveTo event instead.

MoveToFolder

This event is triggered when the user moves to the specified test set folder in the test sets tree.

Syntax	**MoveToFolder(Folder)** where **Folder** is the **ISysTreeNode** interface. For more information, refer to the *HP ALM Open Test Architecture API Reference*.
Type	Sub
Availability	`MoveToFolder`

Note: The MoveToFolder event is not listed in the Scripts Tree of the Script Editor. This event is supported for purposes of backward compatibility. We recommend you use MoveToFolder instead.

MoveToSubject

This event is triggered when the user moves to the specified subject in the test plan tree.

Syntax	**MoveToSubject(Subject)** where **Subject** is the **ISysTreeNode** interface. For more information, refer to the *HP ALM Open Test Architecture API Reference*.
Type	Sub
Availability	`MoveToSubject`

> **Note:** The MoveToSubject event is not listed in the Scripts Tree of the Script Editor. This event is supported for purposes of backward compatibility. We recommend you use MoveToSubject instead.

New

This event is triggered when an object is added to ALM.

You can add code to this event procedure to perform an action when a new object is added. For example, see "Example: Customizing a Defects Module Dialog Box" on page 88.

Syntax	**<entity>_New**
Type	Sub
Availability	• `AnalysisItem_New` • `AnalysisItemFolder_New` • `Baseline_New` • `Bug_New` • `BusinessModelFolder_New` • `BusinessModelPath_New` • `Component_New` • `ComponentFolder_New` • `ComponentStep_New` • `Cycle_New` • `DashboardFolder_New` • `DashboardPage_New` • `DesignStep_New` • `Library_New` • `LibraryFolder_New` • `Release_New` • `ReleaseFolder_New` • `Req_New`

	• Resource_New • ResourceFolder_New • Step_New • Test_New • TestConfiguration_New • TestFolder_New • TestSet_New • TestSetFolder_New

RemoveComponentFromTest

This event is triggered when the user removes a component from a test of type Flow or Business-Process in the Test Script tab.

Version Control: Changing components checked in or checked out by another user, using the RemoveComponentFromTest event, is not supported.

Syntax	**RemoveComponentFromTest**
Type	Sub
Availability	`RemoveComponentFromTest`

RunTests

This event is triggered when the user clicks the **Run** button to run tests in the Test Lab module, provided that Sprinter is not installed and none of the tests is automated.

Syntax	**RunTests(Tests)** where **Tests** is an array of Test Instance IDs.
Type	Sub
Availability	`RunTests`

RunTests_Sprinter

This event is triggered:

- When the user clicks the **Run** arrow and chooses **Run with** Sprinter to run tests in the Test Lab module.
- When the user clicks the **Run** button to run tests in the Test Lab module, if Sprinter is installed and all the tests are manual.

Syntax	**RunTests_Sprinter(Tests)** where **Tests** is an array of Test Instance IDs.
Type	Sub
Availability	`RunTests_Sprinter`

RunTestSet

This event is triggered when the user clicks the **RunTest Set** button to run a test set in the Test Lab module.

Syntax	**RunTestSet(Tests)** where **Tests** is an array of Test Instance IDs.
Type	Sub
Availability	`RunTestSet`

RunTestsManually

This event is triggered when the user clicks the **Run** arrow and chooses **Run Manually** to run tests in the Test Lab module.

Syntax	**RunTestsManually(Tests)** where **Tests** is an array of Test Instance IDs.
Type	Sub
Availability	`RunTestsManually`

Workflow Object and Property Reference

Workflow scripts can reference HP Application Lifecycle Management (ALM) objects to obtain information and to change project values. They can also use properties that return information about the current module and dialog box. This section lists the ALM objects and properties that are available to workflow scripts.

About ALM Objects and Properties 72
Actions Object 73

Action Object 73
Fields Objects 74
Field Object 76
Lists Object 77
TDConnection Object 78
User Object 78
ALM Properties 79

About ALM Objects and Properties

Workflow scripts can obtain information, make decisions based on that information, and change values in the project based on those decisions.

You can obtain information such as the user group to which the current user belongs, and the value of a field, by accessing objects such as the **User** object or the **Field** object.

You can also obtain information about the active module and active dialog box using workflow properties. For more information on these properties, see "ALM Properties" on page 79.

Your script can change the value of a field or field list. To do so, the script modifies the **Value** property or the **List** property of the appropriate **Field** object.

The following table lists the ALM objects that are available when you write a script.

Object	Description
Actions	The list of actions that are available. See "Actions Object" on the next page.
Action	The **Action** object is handled by the **Actions** object. See "Action Object" on the next page.
Fields	Includes the objects that provide access to specific fields. See "Fields Objects" on page 74.
Field	The **Field** object is handled by the **Fields** objects. See "Field Object" on page 76.
Lists	Includes the lists that are available in an ALM project. See "Lists Object" on page 77.
TDConnection	Provides access to open test architecture (OTA) objects. See "TDConnection Object" on page 78.
User	Includes the properties of the current user. This object is available in all modules. See "User Object" on page 78.

Note: In some cases, a function returns the object itself instead of the ID property of the

object. For example, after the following statement has been executed, `testsetf` is a reference to a **TestSetFolder** object:
`Set testsetf =` **TestSet_Fields**`("CY_FOLDER_ID")`**.Value.**

For each ALM object, this section lists the properties of the object. The list includes the property name, a description, and the data type of the property. It indicates whether the property is read-only (R) or whether your script can modify it (R/W).

Version Control: After enabling version control for a project, you should review all its workflow scripts and make adjustments for each checked in entity. This includes the following entities: **Req**, **Test**, **Resource**, and **Component**. For each checked in entity that includes a **Post** function in its script, you must modify the script. To modify, add a **Checkout** function before every **Post** function. Making this modification prevents the Check Out dialog box from opening each time a call to a **Post** function is made. For more information about the Post and Checkout functions, see the *HP ALM Open Test Architecture Reference*.

Actions Object

You can use the **Actions** object to manipulate toolbar buttons, menu commands, and dialog boxes.

The **Actions** object has the following property:

Property	R/W	Type	Description
Action	R	Object	Allows access to every action in a list. The index for this property is the action name.

Action Object

You can use the **Action** object to verify whether a button or command is enabled, checked, or visible. You can also use it to execute actions.

For example, to set the Defect Details dialog box to open automatically when the user moves from one defect to another in the Defects Grid, place the following code in the `Bug_MoveTo` event procedure:

```
Set NewDefectAction=Actions.Action("Defects.DefectDetails")
NewDefectAction.Execute
```

To obtain the name of an action, add the following lines to the ActionCanExecute event procedure, perform the action, and note the action name that is printed in the message:

```
Sub ActionCanExecute(ActionName)
        On Error Resume Next
        MsgBox "You have performed an action named: " & ActionName
        On Error GoTo 0
End Sub
```

This object has the following properties:

Property	R/W	Type	Description
Checked	R/W	Boolean	Indicates whether an action is checked in ALM.
Enabled	R/W	Boolean	Indicates whether an action is enabled. A disabled action cannot be invoked by the user, but can be invoked from the workflow script.
Visible	R/W	Boolean	Indicates whether an action is visible in ALM.

The **Action** object includes the following method:

Method	Description
Execute	Executes the action.

When a workflow script invokes an action using the **Execute** method of the **Action** object, the workflow events that would be triggered if a user initiated the action from a dialog box are by default not triggered. Therefore, when using **Action.Execute**, you must ensure that you do not bypass the site policies you are enforcing with workflow events.

To enable workflow events to be triggered from within a dialog box, set the value of the **AllowReentrancy** flag to **true**. To restore the default settings, so that these events are not triggered, set the value of the **AllowReentrancy** flag to **false**. For example, to set the Add Defect dialog box to open automatically when a user enters the Defects module, place the following code in the **EnterModule** event procedure:

```
AllowReentrancy=true
Set NewDefectAction=Actions.Action("Defects.DefectDetails")
NewDefectAction.Execute
AllowReentrancy=false
```

If the value of the **AllowReentrancy** flag is set to **false**, the dialog box opens as usual, but workflow customizations will not work in the dialog because the workflow events for the dialog box are not triggered.

> **Caution:** Consider carefully the implications of setting the value of this flag to **true**. If you set the value of the flag to **true**, you enable a function to call another function which may call the original function. This can cause an endless loop. This can also occur when functions call internal functions which call the original function.

Fields Objects

You can use the following objects in workflow scripts to access the fields of ALM modules:

Object	Description
AnalysisItem_Fields	Provides access to the fields of the reports and graphs in the Dashboard module.

Object	Description
AnalysisItemFolder_ Fields	Provides access to the fields of the report and graph folders in the Dashboard module.
Baseline_Fields	Provides access to the fields of the baselines in the Libraries module.
Bug_Fields	Provides access to the fields of the defects in the Defects module and the Manual Runner dialog box.
Component_Fields	Provides access to the fields of components in the Business Components module.
ComponentStep_ Fields	Provides access to the fields of component steps in the Business Components module.
Cycle_Field	Provides access to the fields of cycles in the Releases module.
DashboardFolder_ Fields	Provides access to the fields of dashboard page folders in the Dashboard module.
DashboardPage_ Fields	Provides access to the fields of dashboard pages in the Dashboard module.
DesignStep_Fields	Provides access to the fields of the design steps in the Test Plan module.
Library_Fields	Provides access to the fields of the libraries in the Libraries module.
LibraryFolder_ Fields	Provides access to the fields of the library folders in the Libraries module.
Release_Fields	Provides access to the fields of the releases in the Releases module.
ReleaseFolder_ Fields	Provides access to the fields of the release folders in the Releases module.
Req_Fields	Provides access to the fields of the Requirements module.
Resource_Fields	Provides access to the fields of the resources in the Test Resources module.
ResourceFolder_ Fields	Provides access to the fields of the resource folders in the Test Resources module.
Run_Fields	Provides access to the fields of the test runs in the Manual Runner dialog box.
Step_Fields	Provides access to the fields of the steps in the Manual Runner dialog box.
Test_Fields	Provides access to the fields of tests in the Test Plan module.
TestSet_Fields	Provides access to the fields of the test sets in the Test Lab module.

Object	Description
TestSetTest_Fields	Provides access to the fields of the test instances in the Test Lab module.

For example, to set a certain property for all fields in the **Req_Fields** object, you can refer to each field by its ID number (**Req_Fields.FieldById**). To set all fields to be visible (**IsVisible**)in a dialog box, you can use the following code:

```
For i = 1 to Req_Fields.Count
        Req_Fields.FieldById(i).IsVisible = True
Next
```

These objects have the following properties:

Property	R/W	Type	Description
Count	R	Long	Returns the number of fields in the current object.
Field (FieldName)	R	Object	Accesses the fields by field name or field label.
FieldById (FieldID)	R	Object	Accesses the fields by the field ID number.

Tip: To avoid errors if your script attempts to access a non-active or a non-existing field, include **On Error Resume Next** in the script.

Field Object

You can use the **Field** object to access the properties of an entity field.

For example, to display a message box when a user does not have permission to change a value in the **Status** field, you can use the following code:

```
Msgbox "You do not have permission to change "_
& "Bug_Fields.Field("BG_STATUS").FieldLabel field."
```

The **Field** object has the following properties:

Property	R/W	Type	Description
FieldLabel	R	String	The displayed label of the field.
FieldName	R	String	The logical name of the field.
IsModified	R	Boolean	Specifies whether the value was modified.
IsMultiValue	R	Boolean	Specifies whether the field can contain multiple values from a lookup list.

Property	R/W	Type	Description
IsNull	R	Boolean	Specifies whether the field value is absent.
IsReadOnly	R/W	Boolean	Specifies whether the field is read-only.
IsRequired	R/W	Boolean	Specifies whether a field value is required. This enables you to override field customization information. To modify the **IsRequired** property of a field, the **IsVisible** property must be `True`. Changes to **IsRequired** are ignored if the field is not visible. Users must always enter a value for a field that is set as required by the workflow. This applies whether they are modifying an existing record or adding a new record, and even if the field is already empty.
IsVisible	R/W	Boolean	Specifies whether the field is displayed.
List	R/W	List	Sets or retrieves the field list attached to a field of type lookup list.
PageNo	R/W	Integer	Sets or retrieves the page (tab) on which the field is displayed in the New Defect and Defect Details dialog boxes.
Value	R/W	Variant	Sets or retrieves the value of the field.
ViewOrder	R/W	Integer	Sets or retrieves the order in which the fields are displayed in the New Defect and Defect Details dialog boxes. You must set the value for every field in the dialog box.

Lists Object

You can use the **Lists** object to limit field input to a specific list of values.

For example, to set the list in the **Planned Closing Version** field, depending on the **Project** field value, you can use the following code:

```
If Bug_Fields.Field("BG_PROJECT").Value = "Project 1" Then
        Bug_Fields.Field("BG_PLANNED_CLOSING_VER").List _
        = Lists("All Projects")
       ' ...
End If
```

For more information, see "Example: Presenting a Dynamic Field List" on page 95.

The **Lists** object can be used only with fields that are defined as the **Lookup List** type or the **String** type in Project Customization of project entities.

The **Lists** object has the following properties:

Property	R/W	Type	Description
List	R	ISysTreeNode	Accesses the ALM lists.

> **Note:** When workflow customization has been used to change a list of values for a field that has transition rules defined, the field may only be modified in a way that satisfies both the workflow script and the transition rules.

TDConnection Object

In workflow scripts, the only objects that are available are the objects of the module in which the code is written and a limited number of global objects. One of the global objects is the **TDConnection** object. **TDConnection** provides access to the open test architecture (OTA) objects.

You can use the **TDConnection** object to access objects from other modules, and to access general session parameters. You can access **TDConnection** properties in any procedure, from any module.

For more information about the **TDConnection** object, and a list of **TDConnection** properties, refer to the *HP ALM Open Test Architecture API Reference*.

User Object

You can access the **User** object to retrieve the user name of the current user and to check whether the user belongs to a particular user group. You can retrieve or modify the first and last name of the user.

For example, to have a message box open when the user has project administrator permissions, use the following code:

```
If User.IsInGroup("TDAdmin") Then
        MsgBox "The user " & User.FullName & _
        " has administrative permissions for this project."
End If
```

For more information, see "Example: Changing a Field Based on the User Group" on page 93, and "Example: Controlling User Permissions" on page 96.

To access user properties that cannot be accessed by the **User** object, you can use the **TDConnection** object of the ALM open test architecture (OTA).

The **User** object has the following properties:

Property	R/W	Type	Description
FullName	R/W	String	Sets or retrieves the first and last name of the current user.
IsInGroup (GroupName)	R	Boolean	Checks whether or not the current user is a member of a predefined/user-defined group.
UserName	R	String	Returns the user name used when logging in to ALM.

ALM Properties

You can use the **ActiveModule** and **ActiveDialogName** properties to obtain information about the active module and dialog box.

This section includes:

ActiveModule Property 79

ActiveDialogName Property 80

ActiveModule Property

The **ActiveModule** property returns the name of the active ALM module. The following values can be returned:

- `Releases`
- `Libraries`
- `Analysis`
- `Dashboard`
- `Requirements`
- `Business Models`
- `Test Resources`
- `Business Components`
- `Test Plan`
- `Test Lab`
- `Test Runs`
- `Defects`

Example

To open a message box displaying the module name when you move to a new module, use the following code:

```
Sub EnterModule
        On Error Resume Next
        msgbox "You have just entered the " & ActiveModule & _
        " module."
```

```
        On Error GoTo 0
End Sub
```

ActiveDialogName Property

The **ActiveDialogName** property returns the name of the active dialog box.

Example

To open a message box displaying the dialog box name when you open a new dialog box, use the following code:

```
Sub DialogBox(DialogBoxName, IsOpen)
        On Error Resume Next
        msgbox "You have just opened the " & ActiveDialogName & _
        " dialog box."
        On Error GoTo 0
End Sub
```

Workflow Examples and Best Practices

About the Workflow Examples

The workflow examples presented in this section perform several types of tasks. The following table lists the examples that illustrate each type of task.

Workflow Task	See Examples
dialog box customization	"Example: Customizing a Defects Module Dialog Box" on page 88 "Example: Changing Tab Names" on page 91
field value automation	"Example: Adding a Template to a Memo Field" on page 92 "Example: Changing One Field Based on Another Field" on page 92 "Example: Changing a Field Based on the User Group" on page 93
data validation	"Example: Object Validation" on page 93 "Example: Field Validation" on page 94

Workflow Task	See Examples
dynamic field customization	"Example: Presenting a Dynamic Field List" on page 95 "Example: Changing Field Properties when a Field Changes" on page 96
user permission control	"Example: Controlling User Permissions" on page 96
functionality	"Example: Adding Button Functionality" on page 97
error handling	"Example: Error Handling" on page 97
using OTA to obtain session parameters	"Example: Obtaining Session Properties" on page 98
sending mail	"Example: Sending Mail" on page 99
using the **Settings** object	"Example: Storing the Last Values Entered" on page 100
copying values between modules	"Example: Copying Field Values to Another Object" on page 102

Best Practices for Writing Workflow Scripts

This section describes best practices for writing workflow scripts and making sure the scripts run as expected. In addition to the best practices provided in this section, you can refer to the Microsoft Developer Network VBScript Language Reference at http://msdn.microsoft.com/en-us/library/.

The following best practices are described in this section:

General VBScript Tips and Best Practices

- "Checking Value Types Before Use" on the next page
- "Anticipating Full Evaluation of Logical Expressions" on page 83
- "Defining Default Behavior for Select Case and If-Then-Else Statements" on page 84
- "Setting Return Values in Functions" on page 85

ALM Workflow Tips and Best Practices

- "Making Sure that Entity Properties Are Set Before an Entity Comes into Focus" on page 85
- "Check if a Dialog Box is Open" on page 87
- "Avoid Defining Duplicate Subroutines" on page 88

Checking Value Types Before Use

VBScript is a "weakly-typed" programming language. This means that you can create, use, and access data values without initially declaring their types. However, certain operations can be performed only on values of a specific type. Therefore, it is important to check the type of the data before performing any operations on them.

Values of different types behave differently in different statements. Object value behavior is even more unpredictable because the behavior depends on the object's implementation. For example, the object in the call **<entity>_CanDelete(Entity)** can either be text or a subject node.

Recommendations

To avoid unpredictable results:

- Check value types before use, especially for object types. When checking an object type, also check that the object has the properties you access.
- Assume as little as possible—do not assume that a value is of a certain type. Write scripts that can handle all possibilities by using Else statements and Select Case statements.
- Always check parameter types before use with various VBScript functions, such as IsArray, IsDate, IsNull, IsEmpty, IsNumeric, and IsObject.
- Do not assume an object's default property is of a specific type; the type can vary from object to object.
- Use VBScript built-in conversion functions to achieve a degree of type safety.
- When working with objects, check that the value you receive is neither Null or Empty by calling the IsNull and IsEmpty functions.

Examples

For the purposes of the following examples, assume the field values are declared as in the table below.

Field Values	Type
`Bug_Fields["BG_BUG_ID"].Value`	Integer
`Bug_Fields["BG_SUMMARY"].Value`	String
`Bug_Fields["BG_SUBJECT"].Value`	Object implementing the ISysTreeNode interface

In the following example, statement usage is correct. The integer is converted to a string.

```
If Bug_Fields["BG_BUG_ID"].Value = "10" Then...
```

In the following example, statement usage is correct. The strings are comparable.

```
If Bug_Fields["BG_SUMMARY"].Value = "some text" Then...
```

In the following example, statement usage is incorrect. This code can work only when the value of `BG_SUBJECT` field is neither Empty or Null. VBScript also assumes that this objects's default value (meaning, the default property) is either of string type or is comparable with the string type, which is not always the case.

```
If Bug_Fields["BG_SUBJECT"].Value = "My Tests" Then...
```

Anticipating Full Evaluation of Logical Expressions

The VBScript programming language does not short-circuit evaluation of Boolean conditions. VBScript evaluates all the terms in a Boolean logical expression, even if the expression can be established as True or False without evaluating all its terms. For example, in the following example, both <`statement1`> and <`statement2`> are evaluated, even if <`statement1`> resolves to False:

```
<statement 1> AND <statement 2>
```

Recommendations

To avoid errors, check that all values and objects are not Null before attempting to use them.

Examples

The following examples:

- demonstrate incorrect and correct usage of logical expressions
- take into consideration how logical expressions are evaluated

Incorrect Usage

`value.Name` is evaluated even when its value is Null. This causes an error.

```
Sub namecheck(value)
        If Not IsNull(value) And value.Name = "aName" Then
            ' ...
          End If
End Sub
```

Correct Usage

The code is correct on the condition that `value` is an object that contains the `Name` property. The code runs without errors.

```
Sub namecheck(value)
    If Not IsNull(value) And Not IsEmpty(value) Then
         If value.Name = "aName" Then
            ' ...
        End If
      End If
End Sub
```

Defining Default Behavior for Select Case and If-Then-Else Statements

Unpredictable results can occur when no default action is defined for Select Case statements or If-Then-Else statements.

Recommendations

To avoid unpredictable results, always define default behavior when using Select Case of If-Then-Else statements.

Example

The following are examples of incorrect and correct ways to define default behavior for situations not covered by the existing Select Case and If-Then-Else statements.

Incorrect Usage

The author of this subroutine intends for the `BG_USER_01` field to be visible only if the status of the defect is Open, New, or Reopen. However, if the IsVisible property of a Closed or Fixed defect was set to True prior to the instance of this subroutine, that Closed or Fixed defect will also be visible. This is because there is no case statement defined specifically for Closed and Fixed defects.

```
Sub Bug_FieldChange(FieldName)
    If FieldName="BG_STATUS" Then
        Select Case Bug_Fields(FieldName).Value
            Case "Open", "New", "Reopen" _
            Bug_Fields("BG_USER_01").IsVisible = True
        End Select
    End If
End Sub
```

Correct Usage

This subroutine effectively handles all possible cases.

```
Sub Bug_FieldChange(FieldName)
    If FieldName="BG_STATUS" Then
        Select Case Bug_Fields(FieldName).Value
            Case "Open", "New", "Reopen"
                Bug_Fields("BG_USER_01").IsVisible = True
            Case Else
                Bug_Fields("BG_USER_01").IsVisible = False
        End Select
    End If
End Sub
```

Setting Return Values in Functions

If a function ends without a return value, unpredictable and inconsistent results may occur. Also, it is difficult to debug behavior if a return code is not set.

Recommendations

To avoid unpredictable results, set a default return value at the beginning of each function.

Making Sure that Entity Properties Are Set Before an Entity Comes into Focus

It is common practice to set entity properties (such as IsVisible, IsRequired, and List) when creating or modifying a new entity (New or FieldChanged). When writing ALM workflow scripts, it is also important to set entity properties when the entity comes into focus (meaning, when the user navigates to that entity in the ALM graphical user interface). When an entity comes into focus, the MoveTo event is called.

If entity values are not set in the MoveTo event, the end user experience is unpredictable—for example, incorrect values might be displayed in drop-down lists.

Recommendations

To avoid unpredictable results, such as a drop-down list not containing the most up-to-date set of values:

- Make sure that all entity properties are set in the MoveTo event—not just in the New or FieldChanged events.
- Isolate entity properties customization code into a separate routine and call that routine from all relevant events.

Example

The following table provides an example of how to make sure that properties of a defect are set appropriately when the defect is in focus—and not just when it is modified or added.

```
Sub SetupBugFields(Context1, Context2)
    ' Code for customizing defect properties is entered here,
    ' such as set IsVisible, IsRequired, IsReadonly, Label, List...
    If Context1="Focus" Then
          ' Code for handling the focus event is entered here
    ElseIf Context1="FieldChange" Then
            If Context2="RQ_USER_01" Then
                ' Code for handling the FieldChange event
                ' is entered here
            ElseIf Context2="RQ_REQ_STATUS" Then
                ' ... Enter your code here
            Else
                ' ... Enter your code here
            End If
  End If
End Sub
Sub Req_FieldChange(FieldName)
    If FieldName = "RQ_REQ_STATUS" Then
        SetupBugFields("FieldChange", FieldName)
    Else
        '  ...Enter your code here
    End If
End Sub
Sub Req_MoveTo
        SetupBugFields("Focus")
End Sub
```

Check if a Dialog Box is Open

It is helpful to track whether a dialog box is open before performing certain actions. For example:

- Dialog boxes do not need to be refreshed but grid displays do.
- Certain workflow events are not allowed when a dialog box is open.

The DialogBox event can be used to track the visibility of dialog boxes.

Recommendations

To avoid unpredictable results, determine if a dialog box is open before any events occur.

Example

The following example checks whether the dialog box for creating a new defect is open. This is relevant because the `BG_USER_01` field can only be modified for a new defect. If a different dialog box is open, such as the dialog box for editing a defect, the `BG_USER_01` field cannot be modified.

```
' Declare a global variable for each dialog box of interest
Dim NewDefectDialogIsOpen
' Initialize the global variable
NewDefectDialogIsOpen = False
Sub DialogBox(DialogBoxName, IsOpen)
    If DialogBoxName="New Bug" Then
        NewDefectDialogIsOpen = True
    Else
        NewDefectDialogIsOpen = False
    End If
End Sub
Function Bug_FieldCanChange(FieldName, NewValue)
' Initialize the function's return value to avoid
' unpredictable behavior.
Bug_FieldCanChange = True
' The BG_USER_01 field can only be modified for a new defect.
If FieldName="BG_USER_01" Then
    If NewDefectDialogIsOpen Then
        Bug_FieldCanChange = True
    Else
        Bug_FieldCanChange = False
    End If
End If
End Function
```

Avoid Defining Duplicate Subroutines

If you define a subroutine in one section and then add another subroutine with the same name in another section, the subroutines will conflict. One of them will be ignored.

> **Example:** If you define the subroutine, **MySub**, in the Test Lab module script section, and then define another subroutine, MySub, in the Manual Runner script section, one of your defined subroutines will be ignored.

Recommendations

To avoid unpredictable conflicts when defining subroutines, always check if another subroutine with the same name already exists in your project.

Workflow Examples

Example: Customizing a Defects Module Dialog Box

This example shows how you can customize the field layout and other field properties in the Add Defect dialog box. You can create similar code to arrange the layout of the Defect Details dialog box.

This example illustrates a solution that customizes field properties for all user groups. You can also use the script generators to customize the layout of the Defects module dialog boxes. If you use the script generators, you must perform customization separately for each user group.

This example involves the following procedures:

- `SetFieldApp` is a general purpose procedure that receives a field name and its properties as parameters, and assigns the properties to the field. See "SetFieldApp" on the next page.

- `FieldCust_AddDefect` calls `SetFieldApp` for each field in the Add Defects dialog box, to set the properties of the field. For some of the fields, `FieldCust_AddDefect` checks the user group to which the current user belongs, and customizes the field properties accordingly. A call to `FieldCust_AddDefect` is placed in the `Bug_New` event procedure. See "FieldCust_AddDefect" on the next page.

> **Note:** To implement this example, you can run the **Add Defect Field Customization** script generator and then modify the resulting scripts.
>
> - Rename the generated function `WizardFieldCust_Add` to `FieldCust_AddDefect` and modify it as necessary. (Before you modify a generated script, you must rename it so that it is not overwritten the next time you run the script generator.)
>
> - The script generator places a call to `WizardFieldCust_Add` in the event procedure `Bug_New`. Change this to `FieldCust_AddDefect.`

- The function SetFieldApp is generated when you run the script generator. You do not need to rename or modify this function.

SetFieldApp

The subroutine SetFieldApp receives a field name and its properties as parameters, and assigns the properties to the field.

The subroutine assigns the following field properties: field visibility, whether the field is required, the number of the page (tab) on which the field should be displayed, and the view order (from left to right and from top to bottom).

Add a call to the subroutine SetFieldApp in the user-defined function FieldCust_AddDefect. For more information on this function, see "FieldCust_AddDefect" below.

```
Sub SetFieldApp(FieldName, Vis, Req, PNo, VOrder)
    On Error Resume Next
    With Bug_Fields(FieldName)
        .IsVisible = Vis
        .IsRequired = Req
        .PageNo = PNo
        .ViewOrder = VOrder
    End With
    PrintError "SetFieldApp"
    On Error GoTo 0
End Sub
```

FieldCust_AddDefect

The user-defined function FieldCust_AddDefect calls the function SetFieldApp.

The function first sets all fields to be invisible, not required, and to appear on page 100 at location 0. This ensures that if you add a new field using the **Project Entities** link on the Project Customization window, the layout will not be changed.

Add a call to FieldCust_AddDefect in the Bug_New event procedure so that it will be triggered when a user adds a new defect:

```
Sub Bug_New
        FieldCust_AddDefect
End Sub
```

First, the code handles the fields that are common to all user groups. It uses conditional statements for the fields that will appear in the dialog box only for specific user groups, or that will have different properties for different users.

```
Sub FieldCust_AddDefect
        On Error Resume Next
        ' Initialize the fields of the defect
        For i= 0 To Bug_Fields.Count -1
```

```
        SetFieldApp Bug_Fields.FieldByID(i).FieldName, _
        False, False, 100, 0
    Next
    ViewNum = 0
    PageNum = 0
   ' Set fields that are in common for all user groups
    SetFieldApp "BG_BUG_ID", True, True, PageNum, ViewNum
    ViewNum = ViewNum + 1
    SetFieldApp "BG_DESCRIPTION", True, False, PageNum, ViewNum
    ViewNum = ViewNum + 1
    SetFieldApp "BG_SUMMARY", True, True, PageNum, ViewNum
    ViewNum = ViewNum + 1
    SetFieldApp "BG_DETECTED_BY", True, True, PageNum, ViewNum
    ViewNum = ViewNum + 1
    SetFieldApp "BG_DETECTION_DATE", _
    True, True, PageNum, ViewNum
    ViewNum = ViewNum + 1
    SetFieldApp "BG_DETECTION_VERSION", True, True, PageNum, _
    ViewNum
    ViewNum = ViewNum + 1
    SetFieldApp "BG_SEVERITY", True, True, PageNum, ViewNum
    ViewNum = ViewNum + 1
    SetFieldApp "BG_PRIORITY", True, True, PageNum, ViewNum
    ViewNum = ViewNum + 1
    SetFieldApp "BG_PROJECT", True, False, PageNum, ViewNum
    ViewNum = ViewNum + 1
    SetFieldApp "BG_REPRODUCIBLE", True, False, PageNum, ViewNum
    ViewNum = ViewNum + 1
    SetFieldApp "BG_STATUS", True, False, PageNum, ViewNum
    ViewNum = ViewNum + 1
    ' Set fields that are different for different user groups.
    ' Since one user can belong to multiple user groups,
    ' or none of these groups, there is no need for an Else statement.
    If User.IsInGroup("Developer") Then
        SetFieldApp "BG_PLANNED_CLOSING_VERSION", True, False, _
        PageNum, ViewNum
        ViewNum = ViewNum + 1
        SetFieldApp "BG_PLANNED_FIX_TIME", True, False, PageNum, _
        ViewNum
        ViewNum = ViewNum + 1
    End If

    If User.IsInGroup("QATester") Then
        PageNum = PageNum + 1
        SetFieldApp "BG_USER_01", True, False, PageNum, ViewNum
        ViewNum = ViewNum + 1
        SetFieldApp "BG_USER_02", True, False, PageNum, ViewNum
        ViewNum = ViewNum + 1
    End If
```

```
        SetFieldApp "BG_ACTUAL_FIX_TIME", True, False, PageNum, _
        ViewNum
        ViewNum = ViewNum + 1
        ' ...
        PrintError "FieldCust_AddDefect"
        On Error GoTo 0
End Sub
```

Example: Changing Tab Names

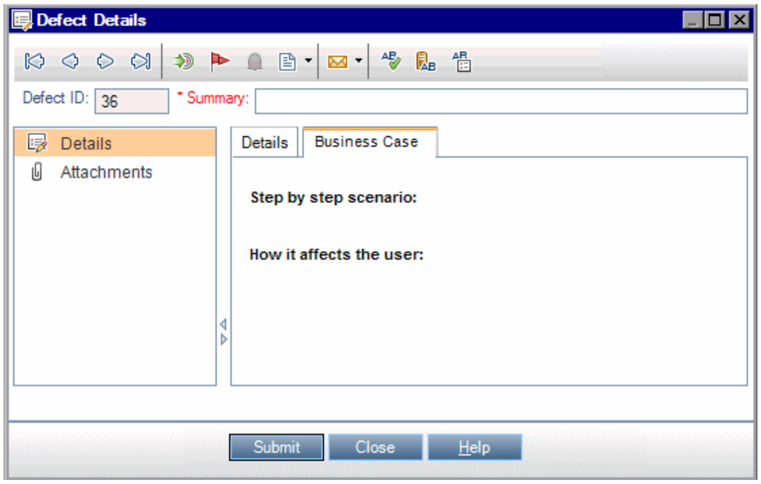
You can change the names of the tabs on the Add Defect dialog box. This example sets the tabs to `General`, `Environments`, and `Business Case`.

Add the following code to the `GetNewBugPageName` event procedure, which is triggered before ALM opens the Add Defect dialog box. To change the tab names on the Defect Details dialog box, add similar code to the `Defects_GetDetailsPageName` event procedure.

```
Sub Bug_New
       On Error Resume Next

               Bug_Fields.Field("BG_ACTUAL_FIX_TIME").PageNo = 1
               Bug_Fields.Field("BG_ESTIMATED_FIX_TIME").PageNo = 2

       On Error GoTo 0
End Sub

Function GetDetailsPageName(PageName,PageNum)
On Error Resume Next

if ActiveDialogName = "New Bug" then
       Select case PageNum
               case "1"
                       GetDetailsPageName="General"
               case "2"
                       GetDetailsPageName="Environments"
               case else
                       GetDetailsPageName="Business Case"
       End Select
end if

On Error GoTo 0
End Function
```

Example: Adding a Template to a Memo Field

You can use workflow scripts to add a default template to a memo field. This example adds text to a memo field called **Business Case** to display the following template:



Perform this customization by placing the HTML code for the text into the **BG_USER_25** field when a defect is added. This example assumes that the user-defined field **BG_USER_25** stores a business case string.

Add the code to the `Bug_New` event procedure, which is triggered when a user adds a new defect.

```
Sub Bug_New
        On Error Resume Next
        Bug_Fields("BG_USER_25").value = _
        "<html><body><b>Step by step scenario:</b>" & _
        "<br><br><br><b>How it affects the user:</b></body></html>"
        PrintError "Bug_New"
        On Error GoTo 0
End Sub
```

Example: Changing One Field Based on Another Field

This example demonstrates how you can change a field value based on the value entered into another field.

For example, you can cause defects to be assigned to user **alex_qc** when `UI Suggestion` is typed into the **Category** field, and to user **alice_qc** when `Security Issues` is typed.

The example assumes that the user-defined field **BG_USER_05** is used to store the category. When the **Category** field is changed in the Defects module, the **BG_RESPONSIBLE** field is assigned the appropriate value.

Add the code to the Bug_FieldChange event procedure so that it is triggered when a user changes a field value for a defect.

```
Sub Bug_FieldChange(FieldName)
        On Error Resume Next
```

```
        If FieldName = "BG_USER_05" then
            Select case Bug_Fields("BG_USER_05").Value
            case "UI Suggestion"
                Bug_Fields("BG_RESPONSIBLE").value="alex_qc"
            case "Security Issue"
                Bug_Fields("BG_RESPONSIBLE").value="alice_qc"
            Case Else
                Bug_Fields("BG_RESPONSIBLE").value="non-assigned"
            End Select
        End If
        PrintError "Bug_FieldChange"
        On Error GoTo 0
End Sub
```

Example: Changing a Field Based on the User Group

This example demonstrates how you can change a field value according to the user group of the user entering the defect.

In this example, the user-defined field **BG_USER_01** is a detection mode field in which the user who detected the defect can enter the way in which it was discovered. Possible values are `Formal testing`, `Informal testing`, and `BTW`.

The example sets the value of the detection mode field to `BTW` when a defect is opened by a user who is not in the `QA Tester` group. If the defect is opened by a user who is in the `QA Tester` group, the default value `Formal testing` is set.

Add the code to event procedure `Bug_New`, so that it is triggered when a defect is added.

```
Sub Bug_New
        On Error Resume Next
        If not User.IsInGroup("QATester") then
            Bug_Fields("BG_USER_01").Value = "BTW"
        Else
            Bug_Fields("BG_USER_01").Value = "Formal testing"
        End If
        PrintError "Bug_New"
        On Error GoTo 0
End Sub
```

Example: Object Validation

This example demonstrates how you can perform validations of all fields by using the `CanPost` event procedure. For example, this code segment ensures that a user cannot reject a defect without adding a comment.

In this example, a user may not post a defect where the defect status (**BG_STATUS**) has been changed to `Rejected` unless some explanatory text has been typed in the **R&D Comment** field (**BG_DEV_COMMENTS**).

Add the code to the Bug_CanPost event procedure so that the check is performed when the user attempts to submit the defect.

```
Function Bug_CanPost
        ' Initialize the function's return value
        ' to avoid unpredictable behavior.
        Bug_CanPost =  False
        On Error Resume Next
        If Bug_Fields("BG_STATUS").IsModified and _
        Bug_Fields("BG_STATUS").Value = "Rejected" and _
        not Bug_Fields("BG_DEV_COMMENTS").IsModified then
            Bug_CanPost = False
            msgbox "You must enter a comment when rejecting a defect."
        Else
            Bug_CanPost = True
        End If
        PrintError "Bug_CanPost"
        On Error GoTo 0
End Function
```

Example: Field Validation

This example demonstrates how to validate a single field value. For example, the following code segment shows how you can ensure that a user in a specific group cannot lower the priority of a defect.

In this example, if the user is in the QATester group and the **BG_PRIORITY** field is being modified, the new value of the **BG_PRIORITY** field cannot be lower than the current value.

This example assumes that in the **Priority** field list for the project, lower priorities come first when the values are sorted in ascending order. For example, the list meets this requirement if the elements are as follows: 1-Low, 2-Medium, 3-High.

Add the code to the Bug_FieldCanChange event procedure so that it is triggered when the user attempts to change a defect field value.

```
Function Bug_FieldCanChange(FieldName, NewValue)
        ' Initialize the function's return value
       ' to avoid unpredictable behavior.
        Bug_FieldCanChange = True
        On Error Resume Next
        If User.IsInGroup("QATester") and FieldName ="BG_PRIORITY" _
        Then
            If NewValue < Bug_Fields("BG_PRIORITY").Value then
                Bug_FieldCanChange = False
                msgbox "You do not have permission to lower " _
                & "defect priority."
            Else
                Bug_FieldCanChange = True
            End If
        Else
```

```
            ' Enter your code here.
        End If
        PrintError "Bug_FieldCanChange"
        On Error GoTo 0
End Function
```

Example: Presenting a Dynamic Field List

This example demonstrates how you can present a different field list in a field, depending on the value of another field.

The user-defined function `SW_SetLists_Environment` checks the value of the **Environment Specification** field and assigns the appropriate field list to the **Environment Type** field.

This example assumes that the field lists have been defined in the project.

> **Note:** To use workflow scripts to change or create lists that can be assigned to fields, you must use the Open Test Architecture (OTA) interface.

Add code to the `Bug_MoveTo` event procedure so that the user-defined function `SW_SetLists_Environment` is called when the user changes focus in the defects module.

```
Sub Bug_MoveTo()
        On Error Resume Next
        SW_SetLists_Environment
        PrintError "Bug_MoveTo"
        On Error GoTo 0
End Sub
```

Add code to the `Bug_FieldChange` event procedure so that the user-defined function `SW_SetLists_Environment` is called when a user changes the value of the **Environment Type** field in the Defects module.

```
Sub Bug_FieldChange(FieldName)
        On Error Resume Next
        If FieldName = "BG_USER_01" then
            SW_SetLists_Environment
        Else
            ' Enter your code here.
        End If
        PrintError "Bug_FieldChange"
        On Error GoTo 0
End Sub
```

The user-defined function `SW_SetLists_Environment` checks the value of the **Environment Specification** field (**BG_USER_02**) and assigns the appropriate field list to the **Environment Type** field (**BG_USER_01**).

```
Sub SW_SetLists_Environment()
        Dim listName
        On Error Resume Next
        Select Case Bug_Fields("BG_USER_01").Value
```

```
        Case "Browser"
            listName = "Browsers"
        Case "Database Type"
            listName = "Database Type"
        Case "Operating System"
            listName = "Platform"
        Case "Web Server"
            listName = "Web Server"
        Case Else
            listName = "Environment Specification"
        End Select
        Bug_Fields("BG_USER_02").List = Lists(listName)
        PrintError ("Set Environment List")
        On Error GoTo 0
End Sub
```

Example: Changing Field Properties when a Field Changes

This example demonstrates how you can change the properties of a field when a different field is changed.

In this example, if the status of the defect (**BG_STATUS**) is changed to `Closed`, the user must provide a value in the field **Closed in Build** (**BG_CLOSING_VERSION**).

Add the code to the `Bug_FieldChange` event procedure, to make the **Closed in Build** field a required field if the status is changed to `Closed`.

```
Sub Bug_FieldChange(FieldName)
        On Error Resume Next
        If FieldName= "BG_STATUS" then
            If Bug_Fields("BG_STATUS").value="Closed" then
                Bug_Fields("BG_CLOSING_VERSION").IsRequired=True
            Else
                Bug_Fields("BG_CLOSING_VERSION").IsRequired=False
            End If
        Else
            ' Enter your code here.
        End If
        PrintError "Bug_FieldChange"
        On Error GoTo 0
End Sub
```

Example: Controlling User Permissions

This example demonstrates how you can prevent members of specific user groups from performing an action.

The code allows a user to replace a defect field value only if the user belongs to the `Admin` user group.

Add the code to the ActionCanExecute event procedure so that the check is performed when a user attempts to execute an action.

```
Function ActionCanExecute(ActionName)
        ' Initialize the function's return value
       ' to avoid unpredictable behavior.
        ActionCanExecute = False
        On Error Resume Next
        If ActionName = "UserDefinedActions.BugReplaceAction1" _
            And Not User.IsInGroup("Admin") then
            ActionCanExecute = False
            msgbox "You do not have permission to perform this action"
        Else
            ActionCanExecute = True
        End If
        PrintError "ActionCanExecute"
        On Error GoTo 0
End Function
```

Example: Adding Button Functionality

This example opens a calculator when a user clicks a button defined with action name `Calculator`.

Add the code to the `ActionCanExecute` event procedure, so that it is triggered when a user initiates an action.

For information about the **Wscript.Shell** object, refer to the Microsoft documentation. To access help for the VBScript language, choose **Help > VBScript Home Page** in the Script Editor.

```
Function ActionCanExecute(ActionName)
        ' Initialize the function's return value to
        ' avoid unpredictable behavior.
        ActionCanExecute = DefaultRes
        On Error Resume Next
        If ActionName = "UserDefinedActions.Calculator" Then
            Set shell = CreateObject("Wscript.Shell")
            shell.Run "Calc"
            Set shell = Nothing
        End If
        ActionCanExecute = DefaultRes
        PrintError "ActionCanExecute"
        On Error GoTo 0
End Function
```

Example: Error Handling

This example demonstrates how you can display a standard error message. Error handling should be added to each workflow script that you write, because errors that are not detected by the workflow code can cause the user's browser to crash.

The user-defined function PrintError receives the name of the calling procedure as a parameter. If an error has occurred, PrintError prints out the error number, description and severity, and the name of the procedure in which the error occurred.

You do not need to create an **Err** object, because it is intrinsic to VBScript. For more information about the **Err** object, refer to the Microsoft documentation.

```
Sub PrintError(strFunctionName)
        If Err.Number <> 0 Then
            MsgBox "Error #" & Err.Number & ": " & Err.Description, _
            vbOKOnly+vbCritical, _
            "Workflow Error in Function " & strFunctionName
        End If
End Sub
```

The following code segment illustrates how you can add error handling to your subroutines.

```
Sub <sub_name>()
        On Error Resume Next
        ...
        [Your code here]
        ...
        PrintError "<sub_name>"
End Sub
```

The following code segment illustrates how you can add error handling to your functions.

```
Function <function_name>()
        On Error Resume Next
        ...
        [Your code here]
        ...
        PrintError "<function_name>"
End Function
```

Example: Obtaining Session Properties

This example demonstrates how to use the **TDConnection** object to obtain the properties of the current session. Add the code to the procedure where these properties are needed. The properties do not depend on each other, so each of the properties can be retrieved separately.

The following are examples of session properties:

```
TDConnection.ServerName
TDConnection.ServerTime
TDConnection.DomainName
TDConnection.ProjectName
User.UserName
```

Note that there is no need to use **TDConnection** to retrieve the user name because the workflow has a predefined **User** object. For more information, see "TDConnection Object" on page 78.

The example below tests the first five characters of the server URL to determine whether the user is connected to the server using HTTP or HTTPS:

```
If Left(UCase(TDConnection.ServerName), 5) = "HTTPS" Then
        MsgBox "You are currently connected to the server using SSL."
Else
        MsgBox "You are not using SSL."
End If
```

Example: Sending Mail

These examples demonstrate how to use the **TDConnection** object to send mail when a defect is submitted, and to send mail when a field value changes in the Test Plan module.

Sending Mail when a Defect is Submitted

This example sends mail when a defect is submitted.

Add a call to the `SendDefect` procedure in the `Bug_AfterPost` event procedure.

> **Note:** If the `SendDefect` procedure is called before the defect is submitted, the values that were changed in the current modification will not be included. The database is updated with the new values only after the defect is posted.

```
Sub SendDefect (iObjectId, strTo, strCc, strSubject, strComment)
        On Error Resume Next
        Dim objBugFactory, objBug
        Set objBugFactory = TDConnection.BugFactory
        Set objBug = objBugFactory.Item(iObjectId)
        objBug.Mail strTo, strCc, 2, strSubject, strComment
        Set objBug = Nothing
        Set objBugFactory = Nothing
        PrintError "SendDefect"
        On Error GoTo 0
End Sub
```

The constant `2` in the call to `objBug.Mail` indicates that the history should be included with the mail. For a list of the constants that can be used to customize email, refer to the `tagTDMAIL_FLAGS` enumeration in the *HP ALM Open Test Architecture API Reference*. In workflow scripts, use numeric constants and not the enumeration values.

Sending Mail when a Test Plan Module Field Value Changes

The example below demonstrates mail notification when the value of the status field is changed in the Test Plan module.

The code is added to the `Test_FieldChange` event procedure. It constructs a subject and comment for the email, and calls a user-defined function, `SendTest`. SendTest sends mail from the

Test Plan module. You can code `SendTest` similarly to the `SendDefect` subroutine shown in "Sending Mail when a Defect is Submitted" on the previous page.

```
Sub Test_FieldChange(FieldName)
        On Error Resume Next
        Dim strSubject, strComment
        If FieldName = "TS_STATUS" Then
            strSubject = "Test Change Notification" & _
                " for project " & TDConnection.ProjectName & _
                " in domain " & TDConnection.DomainName
            strComment = "The user " & User.FullName & _
                " changed the status of the test " & _
                Test_Fields("TS_NAME").Value & _
                " to " & Test_Fields("TS_STATUS").Value
            SendTest Test_Fields("TS_TEST_ID").Value, _
                Test_Fields("TS_RESPONSIBLE").Value, "[QA Testers]", _
                strSubject, StrComment
        End If
End Sub
```

Example: Storing the Last Values Entered

This example shows how to use the **TDConnection** object to implement persistent data between actions. The lifetime of a variable in a routine is only for the routine run. Therefore, persistent data must be stored if it must be available later. It is recommended that you use the ALM API to store persistent data whenever possible instead of using external objects, files, or the registry.

In this example, a user-defined function `SW_KeepLastValue` uses the **Settings** object to save the values typed into the fields **BG_DETECTION_VERSION**, **BG_USER_01**, and **BG_USER_03** when a user posts a defect. These values are retrieved and assigned as default values when this user adds a new defect.

The user-defined function is called with the `SET` action from `Bug_CanPost`, before a new defect is posted by the user. The values in the fields are stored.

```
Function Bug_CanPost()
        ' Initialize the function's return value to
        ' avoid unpredictable behavior.
        Bug_CanPost = True
        If Bug_Fields("BG_BUG_ID").Value = "" Then
            SW_KeepLastValue ("SET")
        End If
End Function
```

The function is called with the `GET` action from the `Bug_New` event procedure. When a user adds a new defect, the values stored in the fields for this user are entered into these fields.

```
Sub Bug_New()
        SW_KeepLastValue ("GET")
End Sub
```

Depending on the action passed as a parameter, the user-defined function `SW_KeepLastValue` stores the values of the fields in the common settings table for the current user, or reads the values from the **Settings** object and assigns the values to the appropriate fields.

```
Sub SW_KeepLastValue(action)
Dim tdc, vals, flds
Dim uset, pairs, pair
Dim bld
On Error Resume Next
        bld = ""
        Set tdc = TDConnection
        Set uset = tdc.UserSettings
        If action = "SET" Then
            flds = Array("BG_DETECTION_VERSION", _
            "BG_USER_01", "BG_USER_03")
            vals = ""
            For i = 0 To UBound(flds)
                If vals <> "" Then vals = vals & ";"
                vals = vals & flds(i) & "=" & _
                Bug_Fields(flds(i)).Value
            Next
            'Open category KeepLValueSetting
            uset.Open ("KeepLValueSetting")
            'Setting KeepValueFields in category KeepLValueSetting
            uset.Value("KeepValueFields") = vals
            uset.Close
        End If 'SET
        If action = "GET" Then
            uset.Open ("KeepLValueSetting")
            vals = uset.Value("KeepValueFields")
            If vals <> "" Then
                pairs = Split(vals, ";")
                For i = 0 To UBound(pairs)
                    pair = Split(pairs(i), "=")
                    If UBound(pair) = 1 Then
                        Select Case pair(0)
                            Case "BG_USER_03"
                                bld = pair(1)
                            Case Else
                                If Bug_Fields(pair(0)).Value = "" Then
                                   Bug_Fields(pair(0)).Value = pair(1)
                                End If
                        End Select
                        If Bug_Fields("BG_DETECTION_VERSION").Value _
                        <> ""
                        And bld <> "" Then
                            SW_SetLists_VersionsBuilds _
                            "BG_DETECTION_VERSION", _
                            "BG_USER_03"
                            Bug_Fields("BG_USER_03").Value = bld
```

```
                            If Err.Number <> 0 Then Err.Clear
                        End If 'Bug_Fields
                    End If 'UBound(pair)
                Next
            End If 'vals <> ""
        End If 'GET
        uset.Close
        PrintError ("Keep Last Value (" & action & ")")
        On Error GoTo 0
End Sub
```

Example: Copying Field Values to Another Object

This example shows how to use the **TDConnection** object to copy the value from the **Build Number** field of a Run (**RN_USER_02**) to the **Last Ran On Build** field of a Test in a Test Set (**TC_USER_03**).

Add the code to the `Run_AfterPost` event procedure.

```
Sub Run_AfterPost
        On Error Resume Next
        Dim tdc
        set tdc = TDConnection
        Dim TSFact 'As TestSetFactory
        Set TSFact = tdc.TestSetFactory
        Dim TstSet 'As TestSet
        Set TstSet = TSFact.Item(Run_Fields("RN_CYCLE_ID").Value)
        MsgBox TstSet.Name
        Dim TSTestFact 'As TSTestFactory
        Set TSTestFact = TstSet.TSTestFactory
        Dim TSTst 'As TSTest
        Set TSTst = _
        TSTestFact.Item(Run_Fields("RN_TESTCYCL_ID").Value)
        MsgBox TSTst.Name

        TSTst.Field("tc_user_03").value = _
        Run_Fields("RN_USER_02").Value
        TSTst.Post

        PrintError ("Run_AfterPost")
        On Error GoTo 0
End Sub
```

Installation

Prerequisites

Pre-Installation Checklist

Review and verify the following checklist before installing ALM. This checklist outlines the information that you must supply during the installation process. For detailed prerequisite information, see the chapters in this part that are relevant to your installation.

Check	Information Required
Installation Machine	• Operating system version • CPU type • Free disk space • Free memory For the list of supported system environments, refer to the *Readme*. **Note:** The supported environment information in the *Readme* is accurate for the ALM12.00 release, but there may be subsequent updates. For the most up-to-date supported environments, refer to the HP Software Web site using the following URL: http://www.hp.com/go/TDQC_SysReq.
Setup Paths	• Installation path • Deployment path **Note:** • You can accept the default paths offered by the Installation and Configuration wizards, or enter alternative paths. • The installation path must not include folders with accented characters (for example, ä, ç, ñ). • The installation path and the deployment path cannot contain non-English characters. • You must have full permissions on the installation and deployment directories.

Check	Information Required
License Key	License file
Cluster Description	• Is clustering used? • Cluster hosts
Encryption Passphrases	• Communication security passphrase • Confidential data passphrase **Note:** In a cluster, use the same passphrase on all nodes.
Application Server	The port number
Mail Server	• Server type • Server host • Server port
Demo Project	Do you require the Web-based demo application for work with the *HP Application Lifecycle Management Tutorial*?
Database Server	• Database type • Database version • Database server name • Database administrator user name • Database administrator user password • Database port • Database SID (Oracle only) • Default tablespace (Oracle only) • Temp tablespace (Oracle only)
Site Administration	• Site administrator user name • Site administrator password

Check	Information Required
Existing ALM/Quality Center Installation	If there is an existing Site Administration schema, provide the following information for the existing version: • ALM/Quality Center version • ALM/Quality Center host • Confidential data passphrase • Database server name • Database administrator user name • Database administrator password • Site Administration database schema name • Site Administration database schema password • Repository folder location • Site administrator user name • Site administrator password
Repository	Repository folder location

Prerequisites: Windows Operating Systems

System Configurations: Windows

Verify that your server machine meets the ALM system configurations. For the recommended and supported system configurations for your ALM server machine, refer to the *Readme*.

> **Note:** The supported environment information in the *Readme* is accurate for the ALM 12.00 release, but there may be subsequent updates. For the most up-to-date supported environments, refer to the HP Software Web site using the following URL: http://www.hp.com/go/TDQC_SysReq.

ALM can be deployed on a VMware ESX/ESXi server according to the VMWare guest operating system compatibility matrix.

Required Permissions: Windows

Verify that you have the required permissions to install ALM on a server machine.

> **Note:** Some permissions require access to the **ProgramData** folder. This folder is hidden by default. To show hidden files and folders, perform the relevant steps for your operating system.

- If you are upgrading from a previous version of ALM/Quality Center with a remote repository, the ALM/Quality Center application server user account must have network access to the remote repository. For more information, contact your network administrator.
- You must be logged on as a local or domain user with administrator permissions. Your user name cannot include a pound sign (**#**) or accented characters (such as, **ä**, **ç**, **ñ**).
- You must disable User Account Control (UAC) during the ALM installation and configuration.

  > **Note:** In Windows 8, UAC cannot be completely disabled. Instead, use the **Run as Administrator** option during installation and configuration.

- The Distributed Link Tracking Client service must be stopped during the ALM installation and configuration.
- We recommend disabling anti-virus software during the ALM installation and configuration.
- You must have the following file system and registry key permissions:
  - Full read permissions to all the files and directories under the directory in which ALM is installed. The installation directory path is specified by the user during installation. By default, ALM writes the installation files to: **C:\Program Files\HP\HP Application Lifecycle Management**.
  - Full read, write, and execute permissions to the directory on which ALM is deployed. The deployment directory is specified by the user during installation. By default, ALM is deployed in **C:\ProgramData\HP\ALM**.
  - Full read and write permissions to the repository directory, which contains the **sa** and **qc** directories. The repository path is specified by the user during installation. By default, it is located under the ALM deployment directory. For more information on the repository, refer to the *HP Application Lifecycle Management Administrator Guide*.
  - Full read permissions to the system root (**%systemroot%**) directory. If you do not have these permissions, you can still install ALM, but you cannot install any patches.
  - Full read and write permissions to the installation and configuration log files directory. Installation and configuration log files are written to **C:\ProgramData\HP\ALM\log**.
  - Full read and write permissions to all the keys under **HKEY_LOCAL_MACHINE\SOFTWARE\Mercury Interactive**.

Clustering: Windows

Check with your system administrator whether you are installing ALM on a single node or as a cluster.

If you are installing ALM on cluster nodes, verify which machine to use as the first node to start the installation and the number of machines you should use. This depends on the number of users and availability considerations.

When installing on additional nodes:

- **ALM version.** You must install the same version of ALM on all nodes.
- **Operating System.** You must install the same version of the operating system, including all patches, updates, or hot fixes, on all nodes.
- **Site Administration schema.** All nodes must point to the Site Administration schema.
- **Database details.** Configure all nodes with the same database information.
- **Confidential Data Passphrase.** You must use the same Confidential Data Passphrase on all nodes.
- **Repository path.** All nodes must point to the repository path that is defined on the first node. It is important that you enter the repository path using the exact same characters on all nodes. For example, you cannot have the path on the first server node defined as **c:\alm\repository** and on additional nodes defined as **\\server1\c$\alm\repository**—the **\\server1\c$\alm\repository** path must appear on every node.

ALM Repository Path: Windows

The location of the repository directory is specified by the user during installation. The default location is: **C:\ProgramData\HP\ALM\repository**. You must have full control permissions to the ALM repository path as described in "Required Permissions: Windows" on the previous page.

Prerequisites: Linux/Oracle Solaris Operating Systems

System Configurations: Linux

Verify that your server machine meets the ALM system configurations. For the recommended and supported system configurations for your ALM server machine, refer to the *Readme*.

> **Note:** The supported environment information in the *Readme* is accurate for the ALM12.00 release, but there may be subsequent updates. For the most up-to-date supported environments, refer to the HP Software Web site using the following URL: http://www.hp.com/go/TDQC_SysReq.

Consider the following for implementing ALM configurations:

- Verify that you have a supported kernel by running uname **-a**.
- ALM can be deployed on a VMware ESX/ESXi server according to the VMWare guest operating system compatibility matrix.

Installing ALM for Non-Root Users

By default, the ALM installer for the Linux operating systems requires a root user.

If you are unable to work with ALM using the root user because of security concerns, speak to your system administrator about using a non-root user with sudo permissions to install and run ALM.

> **Note:** Installing ALM as non-root user without sudo permissions is not supported and causes installation problems.

To use a non-root user with sudo permissions to install and run ALM:

> **Note:** The **sudo** package is included by default on some systems. These instructions assume that sudo is installed on the target machine. If sudo is not included by default, it can be downloaded and installed from http://www.gratisoft.us/sudo/download.html.

1. Create an **ALM_Admin** user.
2. Edit the **sudoers** file to grant sudo permissions to the **ALM_Admin** user within the ALM installation directory. This allows the **ALM_Admin** user to run the installation file with root privileges.

   > **Example**
   >
   > If the administrator decides that the ALM installation directory is **/user/Install/ALM**, the following line should be added to the sudoers file: **qcadmin ALL=NOPASSWD:/user/Install/ALM**

3. Check if the **/var/opt/HP** folder exists. If it does not exist, create it.
4. Give the **ALM_Admin** user Read/Write/Execute permissions to the **/var/opt/HP** folder.
5. Move the ALM installation file to the installation directory, **/user/Install/ALM**.
6. Use the **ALM_Admin** user to run the installation script and start ALM.

Required Permissions: Linux

The following permissions are required:

- Verify that you have the required permissions to install ALM on a server machine.
- If you are upgrading from a previous version of ALM/Quality Center with a remote repository, the ALM/Quality Center application server user account must have network access to the remote repository. For more information. contact your network administrator.
- You must be logged on as a local or domain user with administrator permissions. Your user name cannot include a pound sign (#) or accented characters (such as, ä, ç, ñ).
- To install ALM, you must have the following file system permissions:
  - Full read and write permissions for all the files and directories under the directory on which ALM is installed. The installation files are used for configuring the server. By default, the ALM installation files are written to: **/var/opt/HP/HP_ALM_Server**.
  - Full read and write permissions to the directory on which ALM is deployed. The deployment directory is specified by the user during installation. By default, ALM is deployed on: **/var/opt/HP/ALM**.
  - Full read and write permissions to the repository directory, which contains the **sa** and **qc** directories. The repository path is specified by the user during installation. By default, it is located under the ALM deployment directory. For more information on the repository, refer to the *HP Application Lifecycle Management Administrator Guide*.
  - Full read and write permissions to the installation and configuration log files directory. Installation and configuration log files are written to: **/var/opt/HP/ALM/log**.
  - Full read and write permissions to the file delivery logs. The log files are written to: **/var/log**.
  - If the file repository is located on a remote machine:
    - On the file server machine, share the file repository directory so that the user running the installation is the owner of the files.
    - On the ALM machine, or on each cluster node, create a mount directory that points to the file repository directory.

Minimum Disk Space Requirements

The following partitions have minimum disk space requirements:

- **/opt.** Requires at least enough free space to accommodate the size of ALM after it has been installed. The approximate size of an installation is 300MB, though the exact amount of space may vary from installation to installation.

- **/var.** Requires at least enough free space equal to the space on the installation DVD, approximately 150MB. A copy of the installation is stored in this partition.
- **/tmp.** Requires a large amount of free space. The exact amount cannot be specified as this partition is also consumed by the operating system. It is advisable that the amount of free space is equal in size to ALM after it has been installed, which is approximately 300MB.

Clustering: Linux

Check with your system administrator whether you are installing ALM on a single node or as a cluster.

If you are installing ALM on cluster nodes, verify which machine to use as the first node to start the installation and the number of machines you should use. This depends on the number of users and availability considerations.

When installing on additional nodes:

- **ALM version.** You must install the same version of on all nodes.
- **Operating System.** You must install the same version of the operating system, including all patches, updates, or hot fixes, on all nodes.
- **Site Administration schema.** All nodes must point to the Site Administration schema.
- **Database details.** All nodes must be configured with the same database information.
- **Confidential Data Passphrase.** You must use the same Confidential Data Passphrase on all nodes.
- **Repository path.** You must mount the file system repository before you start the installation process. The mount should not use any cache mechanisms. For details, contact your network administrator.

  All nodes must mount the shared file server with the same mount name. For example, if the file server is **some.server.org**, and it is mounted on **/mnt/some_server** on the first node, it should be mounted with **/mnt/some_server** on all nodes.

ALM Repository Path: Linux

The location of the repository directory is specified by the user during installation. The default location is: **/var/opt/HP/ALM/repository**. You must have full control permissions to the ALM repository path as described in "Required Permissions: Linux" on the previous page.

Prerequisites: Oracle Database Servers

Connecting ALM to an Oracle Database Server

User Permissions for Connecting ALM to an Oracle Database Server

Database Administrative User Privileges

Following are the privileges required by the ALM database administrative user. Additional explanations about these privileges can be found in the notes at the end of the table.

Privilege	Description
CREATE SESSION WITH ADMIN OPTION (1)	ALM uses this privilege to connect to the database as the ALM database administrative user.
CREATE USER	Required to create a new project user schema when creating a new ALM project.
DROP USER	When deleting an ALM project, ALM attempts to remove the Site Administration database schema from the database server. If there is an insufficient privileges error, ALM ignores the error and requests that the user notify the database administrator to delete (drop) the database user schema.
CREATE TABLE WITH ADMIN OPTION (1)	Required for granting this permission to a newly created ALM project user schema.
CREATE VIEW WITH ADMIN OPTION (1)	Required to create views for ALM projects.
CREATE TRIGGER WITH ADMIN OPTION (1)	Required to create triggers for ALM projects. ALM uses database triggers to collect change history for specific tables.
CREATE SEQUENCE WITH ADMIN OPTION (1)	Required to create sequences for ALM projects.

Privilege	Description
CREATE PROCEDURE WITH ADMIN OPTION (1)	Required to create stored packages for ALM projects. ALM uses packages to collect change history for specific tables.
CTXAPP ROLE WITH ADMIN OPTION (1)	Enables ALM to use the Oracle text searching feature. This role exists only if the Oracle text search component was installed and enabled on the database server.
SELECT ON DBA_FREE_ SPACE (2)	Required to check free space on the database server prior to creating a new Site Administration database schema or a new project.
SELECT ON SYS.DBA_ TABLESPACES (2)	Required to collect a list of tablespaces that exist on the database server prior to creating a new Site Administration database schema or a new project.
SELECT ON SYS.DBA_ USERS (2)	Required to verify the existence of specific database project users. For example, you might want to verify the existence of an Oracle CTXSYS user before creating a new ALM project.
SELECT ON SYS.DBA_ REGISTRY (2)	Required to verify that the text search component is installed on the database server.
SELECT ON SYS.DBA_ ROLES (2)	Required to verify that the text search role (CTXAPP) is installed on the database server.
SELECT ANY TABLE WITH ADMIN OPTION (1) **and** **INSERT ANY TABLE**	Required for various administrative operations when upgrading the Site Administration database schema during installation using the copy and upgrade method, and for enhancing performance when copying a project that has the same source and target database server.

Note:

- (1) An ALM database administrative user must have privileges with Admin Option.
- (2) The SELECT ON SYS privileges can be given directly by the table owner, or through a database application role. To avoid giving these privileges each time, you can grant this role to the ALM database administrative user. The recommended name for this role is **QC_ SELECT_ON_SYS_OBJECTS**. You can create this role using the **qc_sys_db___ oracle.sql** example script, which is located in the **\Utilities\Databases_scripts** directory

on the installation DVD. You should run this script before you run the **qc_admin_db___oracle.sql** script.

Project User Privileges

When creating a new project, ALM creates a project user schema. This user schema hosts all the tables that are used by the project for storing and retrieving data. Following are the required privileges for an ALM project user schema:

Project User Schema Privilege	Description
QUOTA UNLIMITED ON <default tablespace>	Required for creating database objects that are owned by the ALM project user schema. This privilege allows users to create tables in the default tablespace. It replaces the **UNLIMITED TABLESPACE** system privilege that gave users system privileges to create tables in any tablespace, including the **SYSTEM** tablespace.
CREATE SESSION	ALM uses this privilege to connect to the database user schema to perform required operations. For example creating database objects such as tables, and using them to insert, retrieve, and delete data.
• **CREATE TABLE** • **CREATE VIEW** • **CREATE TRIGGER** • **CREATE SEQUENCE** • **CREATE PROCEDURE** • **CTXAPP Role**	For a description of these privileges, see "Database Administrative User Privileges" on page 111.

Tip: The installation DVD contains an example script that describes the recommended permissions required for the ALM database project user schema. This script contains information and does not need to be run. It is located at **\Utilities\Databases_scripts\qc_project_db_oracle.sql**.

Site Administration Database Schema Considerations: Oracle

Be aware of the following schema name and password considerations:

- The default Site Administration database schema name is **qcsiteadmin_db**. If you want to rename the schema, you can change the name when configuring the ALM installation.
- You can create your own ALM user password for accessing the Site Administration database schema.
- If there is an existing Site Administration database schema, you can create a copy of the existing schema and upgrade the copy. This enables you to work with ALM 12.00 and previous versions of ALM/Quality Center simultaneously.

> **Note:** This scenario does not apply to working with Performance Center projects. After you upgrade LAB_PROJECT, you must then upgrade Performance Center projects before they can be used.

Oracle RAC Support

Oracle RAC is a way to enhance Oracle database availability and scalability, allowing it to interact with more than one database instance.

ALM RAC support includes:

- Load balancing between Oracle instances.
- Failover between all specified Oracle RAC nodes at initial connection.

ALM RAC support does not include:

- TAF (Transparent Application Failover) support. A user failing to complete a request upon an Oracle instance crash is required to perform the activity again with a working Oracle instance.

To enable Oracle RAC support:

1. Verify that a file containing information of Oracle database addresses is saved on your ALM machine. The file is named **tnsnames.ora**. The file should contain information similar to the following examples:

   a. This first example shows an RAC TNS Alias using all cluster nodes in the ADDRESS sub-section and a sample of utilizing the Load balance and Failover features:

   ```
   OrgRAC =
   ```

```
(DESCRIPTION =
        (ADDRESS_LIST=
                (FAILOVER = on)
                (LOAD_BALANCE = on)
                (ADDRESS= (PROTOCOL = TCP)(HOST = server1)(PORT = 1521))
                (ADDRESS= (PROTOCOL = TCP)(HOST = server2)(PORT = 1521))
                (ADDRESS= (PROTOCOL = TCP)(HOST = server3)(PORT = 1521))
        )
        (CONNECT_DATA=
        (SERVICE_NAME = myrac.yourcompany.com)
        )
)
```

b. This second example shows an RAC TNS Alias using Single Client Access Name (SCAN). This enables Oracle 11gR2 clients to connect to the database with the ability to resolve multiple IP addresses, reflect multiple listeners in the cluster and handle public client connections. For more information on working with RAC SCAN, refer to the Oracle documentation.

```
OrgRAC_Scan =
(DESCRIPTION =
        (ADDRESS_LIST=
                (FAILOVER = on)
                (LOAD_BALANCE = on)
                (ADDRESS= (PROTOCOL = TCP)(HOST = myrac-cluster-scan)(PORT = 1521))
        (CONNECT_DATA=
        (SERVICE_NAME = myrac.yourcompany.com)
        )
)
```

2. Verify that you have the address of the TNS server to which ALM should refer, for example, OrgRAC.

Prerequisites: Microsoft SQL Database Servers

Connecting ALM to a Microsoft SQL Database Server

Verify the following:

Database type and version	Verify that ALM supports your database type and version. For the list of supported databases, refer to: refer to the *Readme*. **Note:** The supported environment information in the *Readme* is accurate for the ALM12.00 release, but there may be subsequent updates. For the most up-to-date supported environments, refer to the HP Software Web site using the following URL: http://www.hp.com/go/TDQC_SysReq.
Database server name	Verify the name of the database server.
Database user permissions	Verify that you have the database permissions required to connect ALM to the Microsoft SQL database server (not applicable for Windows Authentication). For a list of required permissions, see "User Permissions for Connecting ALM to a Microsoft SQL Database Server" on the next page.

Site Administration database schema	To install ALM on an existing Site Administration database schema (second node or upgrade), you must have: • The existing database schema name and the database administrator permissions required to connect ALM to the database server. • Full read/write permissions on the existing repository. • ALM must have access to the previous Site Administration schema repository path. • Full read/write permissions for the ALM user to the previous schema repository path. • The Confidential Data Passphrase that was used to create the existing schema. For schema name and password considerations, see "Site Administration Database Schema Considerations: SQL" on page 119.
Text Search	Verify that the text search component is installed on the server, even if you do not intend to use it.

User Permissions for Connecting ALM to a Microsoft SQL Database Server

To connect ALM to a Microsoft SQL database server, the installing database user must have sufficient permissions to perform certain administrative tasks in SQL.

If you have the SQL **sa** login, you can use it to install ALM. If you are unable to use the SQL **sa** login due to security reasons, it is recommended that your database administrator create an ALM database administrative login, for example **td_db_admin**, with the specific privileges required to install ALM.

The **td_db_admin** login must have the Database Creators role. You must also grant the **td_db_admin** login the Security Administrators role. This allows the **td_db_admin** login to create and add the **td** user with only those privileges required for running ALM, and to run the Maintain Project activities, such as Verify, Repair, and Update.

To create an ALM database administrative login on a Microsoft SQL Server:

1. Open the **SQL Server Management Studio**.
2. In the **Object Explorer** pane, under the ALM database server, expand the **Security** folder.
3. Right-click the **Logins** folder, and select **New Login**.
4. Type **td_db_admin** as the login name, and select the authentication type (enter password if necessary).
5. Click the **Server Roles** tab, and select the **dbcreator** and **securityadmin** options.
6. Click **OK.**

To test the ALM database administrative login after connecting via this login (SQL Server Authentication):

1. Verify the **select sysdatabases table** permission in the master database:

```
SELECT name FROM sysdatabases where name=<db_name>
```

2. Verify the **create database** permission:

```
CREATE DATABASE <dbName> -- the database name must not already exist
```

3. Verify the **drop database** permission:

```
DROP DATABASE <database_name> -- the database name must exist
```

4. Verify the **select syslogins** permission:

```
SELECT COUNT(*) FROM master..syslogins WHERE name=<dbOwnerName>
```

Note: The **dbOwnerName** must be set to **td**.

To test the ALM database administrative login permissions after connecting via this login (Windows Authentication):

1. Verify the **change database context** permission:

```
USE <dbName>
```

2. Verify the **create database** permission:

```
CREATE DATABASE <dbName> -- the database name must not already exist
```

3. Verify the select on **syslogins** permission:

```
SELECT COUNT(*) FROM master..syslogins WHERE name='<dbOwnerName>'
```

4. Verify the select on **sysusers** permission:

```
SELECT COUNT(*) FROM master..sysusers WHERE name='<dbOwnerName>'
```

Site Administration Database Schema Considerations: SQL

Be aware of the following schema name and password considerations:

- The default Site Administration database schema name is **qcsiteadmin_db**. If you want to rename the schema, you can change the name when configuring the ALM installation.
- You can create your own ALM user password for accessing the Site Administration database schema.
- If there is an existing Site Administration database schema, you can create a copy of the existing schema and upgrade the copy. This enables you to work with ALM 12.00 and previous versions of ALM/Quality Center simultaneously.

  Note: This scenario does not apply to working with Performance Center projects. After you upgrade LAB_PROJECT, you must then upgrade Performance Center projects before they can be used.

Prerequisites: Miscellaneous

License File

Verify that you have the ALM license file.

To activate your license, visit the HP Software Licensing Portal (http://www.hp.com/software/licensing) and enter your Entitlement Order Number.

The license file has a **.dat** file extension by default. Make a note of where you save the file, as during the ALM configuration process you need to specify a path to it.

If you do not have a license, visit the HP Software Licensing Portal and click the **Contact Licensing Support** link.

Security Passphrases

Verify that you have passphrases for confidential data and communication security encryption.

For secondary cluster nodes, verify that you have the confidential data encryption passphrase that you used to install the primary cluster.

When upgrading from an ALM 11.00 or later version of the Site Administration database schema, you must use the same confidential data passphrase as was used for the previous installation.

> **Performance Center:** You must use the same communication security passphrase for the ALM and Performance Center server configurations.

Mail Server Information

A mail server enables ALM users to send emails to other users in a project. You select which server to use as part of the installation configuration process.

Before installing ALM, decide which mail server to use. Ask your system administrator for assistance. If you are using an SMTP Server, check that you have the SMTP Server name and port. The installer checks that the specified mail server name and port are valid and that the mail server is running.

Conflicting Applications

To work with ALM, you may need to disable conflicting applications that are running on the ALM machine. For a list of these applications, see HP Software Self-solve knowledge base article KM176429 (http://h20230.www2.hp.com/selfsolve/document/KM176429). (Requires HP Passport sign-in credentials.)

Prerequisites: Client-side

System Configurations

Required Software

The following must be installed on client machines:

- Microsoft .NET Framework 4.5

Additional Considerations

The following considerations must also be taken into account:

- If you are integrating ALM with other HP testing tools, you must modify the DCOM permissions on your client machine. For more information, see HP Software Self-solve knowledge base article KM187086 (http://h20230.www2.hp.com/selfsolve/document/KM187086). (Requires HP Passport sign-in credentials.)

  ALM Edition: Modifying DCOM permissions is not required for running Functional test sets (server-side test execution).

- You can work with the ALM client using a remote desktop.

- For customers using remote or mass distribution mechanisms, ALM client components can be deployed locally on client machines by running a self-extracting **msi** file. You build the **msi** file by running the HP ALM Client MSI Generator, available from the HP Application Lifecycle Management Add-ins page (**Help > Add-ins**).

Permissions Required to Download ALM Client Components

To enable ALM to work with HP testing tools as well as various other integrations and third-party tools, you need to log in to the client machine with administrator privileges. These privileges are required to install the HP ALM Client Registration add-in, which you use to register ALM client components and Site Administration client components on your client machine.

File System Permissions

You must have the following file system permissions:

- Full read and write permissions on the HP\ALM-Client deployment folder. This is located at %ALLUSERSPROFILE%.

- Full read and write permissions to the Temp (%TEMP% or %TMP%) directory. The installer program writes installation and log files to this directory. This is generally located at C:\Users\<username>\AppData\Local\Temp.

Internet Explorer Configuration

Before you download Application Lifecycle Management on a client machine, you must perform the following configurations to the Internet Explorer browser on the client machine.

- Configure the Custom Level security settings. The Custom Level security setting should be configured for the specific zone of the ALM server.

- Set Internet Explorer as the default Web browser. This ensures that external links to ALM entities can open in ALM.

To configure security settings on the client machine:

1. In Internet Explorer, select **Tools > Internet Options**. The Internet Options dialog box opens.

2. Click the **Security** tab. The Web content zone of the ALM server (Internet or Local intranet) is automatically selected. Click **Custom Level**.

3. In the Security Settings dialog box, configure the following settings:

   Under .NET Framework-reliant components:

   - Set **Run components not signed with Authenticode** to **Enable**.
   - Set **Run components signed with Authenticode** to **Enable**.

   Under ActiveX controls and plug-ins:

   - Set **Run ActiveX controls and plug-ins** to **Enable**.
   - Set **Download signed ActiveX controls** to **Enable** or **Prompt**.

   > **Note:** You do not need to enable **Download signed ActiveX controls** if you install the ALM client using the HP ALM Client MSI Generator Add-in. This allows you to install all ALM modules on a client machine without downloading them through a browser.

4. On Windows 7:

   - It is suggested that you add the ALM server site to the Trusted Sites security zone. This is not mandatory.
   - Disable **the Protected Mode** for the Trusted Sites security zone.

5. Click OK.

To set Internet Explorer as the default web browser:

1. In Internet Explorer, select **Tools > Internet Options**. The Internet Options dialog box opens.

2. Click the **Programs** tab.

3. Under **Default web browser**, make sure that Internet Explorer is set as the default browser. If not, click the **Make default** button.

Enabling User Account Control (UAC)

If you enable UAC on a Microsoft Windows 7, 2008R2, or 2012 operating system, be aware of the following considerations:

- To register ALM client components, you must run Internet Explorer as the administrator.
- To register ALM client components on a shared location of a client machine, you must run Internet Explorer as the administrator.
- Administrator permissions are required to run the **ClientMSIGenerator.exe** file. In addition, you must run the **.exe** file as the administrator.

Troubleshooting the ALM Installation

Disabling Validation Checks for the Installation Wizard

The ALM Installation Wizard automatically performs validation checks to verify that particular system configurations requirements are met. If the ALM configuration does not complete due to a failed validation, you can fix the problem or disable selected validation checks, and rerun the installation.

Note:

- You should disable validation checks only if you decide to take responsibility for the ALM server installation.
- To resolve failures that occur during the ALM Installation Wizard, see "Checking the Installation and Configuration Log Files" on page 128, or "ALM Installation Already Exists" on page 129.
- For troubleshooting tips on database validations, see "Database Validator Fails" on page 129.

To disable configuration validators and rerun the ALM Installation Wizard in Linux:

Note: These instructions also apply when running the Windows silent installation.

1. In the ALM installation directory, locate the **validations.xml** file, which is near the installation executable (**ALM_installer.bin**).
2. Edit the **validations.xml** file by changing the validation value from **true** to **false** as required. Following is an example of the file with all configuration validators active.

```
<validations>
        <os enabled="true" />
        <memory enabled="true" threshold="8" />
        <installation_disk_space enabled="true" threshold="8" />
        <sa-schema enabled="true" />
        <db enabled="true" />
        <mail enabled="true" />
        <license-key enabled="true" />
        <repository enabled="true" />
```

```
        <sa-user enabled="true" />
        <security enabled="true" />
        <alm-services enabled="true" />
        <web-server enabled="true" />
</validations>
```

3. Save the file and rerun the installation.

Configuration Validators

Validator	Checks	To Disable
os	Checks that the operating system is supported. For the list of supported system environments, refer to the *Readme*. **Note:** The supported environment information in the *Readme* is accurate for the ALM12.00 release, but there may be subsequent updates. For the most up-to-date supported environments, refer to the HP Software Web site using the following URL: http://www.hp.com/go/TDQC_SysReq.	<os enabled="false" />
memory	Checks that the customer machine has at least x GB of memory (x is defined by the threshold value, the default is 8 GB).	<memory enabled="false" />
installation_ disk_space	Checks that the installation location has at least x GB of free disk space (x is defined by the threshold value, the default is 8 GB). **Note:** This validation is related only to the installation location. If the installation fails because of a lack of free space in the temporary folder, changing the threshold value or disabling this validation does not affect the failure.	<installation_ disk_space enabled="false" />
sa-schema	Checks Site Administration database settings.	<sa-schema enabled="false" />
db	Checks database connectivity.	<db enabled="false" />

Validator	Checks	To Disable
mail	Checks that the mail server is valid.	<mail enabled="false" />
license-key	Checks the license file key.	<license-key enabled="false" />
repository	Checks that the repository folder is accessible, and has sufficient space.	<repository enabled="false" />
sa-user	Checks site administrator user settings.	<sa-user enabled="false" />
security	Checks encryption passphrases.	<security enabled="false" />
alm-services	Checks Windows service settings.	<alm-services enabled="false" />
web-server	Checks that the HTTP port and web server deployment folder is accessible, and has sufficient space	<web-server enabled="false" />

To disable configuration validators and rerun the ALM Installation Wizard in Windows:

> **Note:** These instructions do not apply when running the Windows silent installation. For Windows silent installation, follow the Linux instructions above.

1. In the ALM installation directory, locate the **validations.xml** file, which is near the installation executable (**ALM_installer.exe**).

2. Edit the **validations.xml** file by changing the validation value from **true** to **false** as required. Following is an example of the file with all configuration validators active.

```
<validations>
        <os enabled="true" />
        <memory enabled="true" threshold="8" />
        <installation_disk_space enabled="true" threshold="8" />
        <sa-schema enabled="true" />
        <db enabled="true" />
```

```
        <mail enabled="true" />
        <license-key enabled="true" />
        <repository enabled="true" />
        <sa-user enabled="true" />
        <security enabled="true" />
        <alm-services enabled="true" />
        <web-server enabled="true" />
</validations>
```

3. Only the following configuration validators are used in the Windows installation wizard:

Validator	Checks	To Disable
os	Checks that the operating system is supported. For the list of supported system environments, refer to the *Readme*. **Note:** The supported environment information in the *Readme* is accurate for the ALM12.00 release, but there may be subsequent updates. For the most up-to-date supported environments, refer to the HP Software Web site using the following URL: http://www.hp.com/go/TDQC_SysReq.	<os enabled="false" />
memory	Checks that the customer machine has at least x GB of memory (x is defined by the threshold value, the default is 8 GB).	<memory enabled="false" />
installation_ disk_space	Checks that the installation location has at least x GB of free disk space (x is defined by the threshold value, the default is 8 GB). **Note:** This validation is related only to the installation location. If the installation fails because of a lack of free space in the temporary folder, changing the threshold value or disabling this validation does not affect the failure.	<installation_ disk_space enabled="false" />
db	Checks database connectivity.	<db enabled="false" />

4. Save the file and rerun the installation.

5. On the Installation Summary page, before clicking **Done**, edit the **run_configuration.bat** file, located under the <installation folder>, to disable validations.

Validator	Checks	To Disable
Existing installation	Checks if an older version of ALM or Quality Center is installed.	-wPreviousInstallationValidator
License file	Checks license file key.	-wLicenseTypeValidator
Security passphrases	Checks encryption passphrases.	-wEncryptionStepValidator
Mail server	Checks that the mail server name is valid.	wMailServerValidator
Database settings	Checks Site Administration database settings.	-wSaSchemaValidator
Site administrator	Checks site administrator user settings.	-wSiteAdminUserValidator
repository folder	Checks that the repository folder is accessible, and has sufficient space.	-wRepositoryValidator

6. Save the **run_configuration.bat** file and click **Done** to continue the installation.

Checking the Installation and Configuration Log Files

If you encounter problems installing ALM, check for errors in the following log files:

Windows File Delivery Logs

Log	Path
Install Completed	<installation folder>\log
Install Failed	on the desktop: HP_Application_Lifecycle_Management_Install_<mm_dd_yyyy_hh_mm_ss>.log

Linux File Delivery Logs

Log	Path
Install Completed	<installation folder>/log
Install Failed	in the user's home folder: HP_Application_Lifecycle_Management_Install_<mm_dd_yyyy_hh_mm_ss>.log

Application Logs

Log	Path
Configuration logs	• **Windows.** <ALM deployment folder>\log • **Linux.** <ALM deployment folder>/log
Site Administration database schema creation logs	• **Windows.** <ALM deployment folder>\log\sa • **Linux.** <ALM deployment folder>/log/sa

ALM Installation Already Exists

If an error message displays during the installation indicating that an ALM installation already exists, uninstall the existing ALM installation and remove all traces of it from the server machine.

Database Validator Fails

During the ALM Server configuration, the database validator performs the following checks:

- Check that the input parameters are correct.
- Check that the Site Administration database schema name was provided.
- Check whether the same authentication type was used as the one used in the previous installation.

Perform the following steps:

1. Check whether the parameters are correct:
   - Read the error message that displays during installation and try to understand and resolve the problem from the root cause.

   - For further clarifications, check with your database administrator.
   - If no error was found and you are sure that the parameters are correct, disable the DB parameters validator. For details, see "Disabling Validation Checks for the Installation Wizard" on page 124.

2. Check that the Site Administration Database Schema name was provided:
   a. Open a database query tool.
   b. Make sure the **PROJECTS** table exists in the Site Administration Database Schema. This table does not exist in the project schema.
3. To check the authentication type of a previous installation:
   a. Navigate to **C:\Program Files\HP\ALM_Server** on Windows, and **/opt/HP/HP_ALM_Server** on Linux and open the application folder.
   b. Extract the contents of **qcbin.war** into a temp file, and open the **siteadmin.xml** file in a text editor.
   c. Search for the **native** property. If its value is set to **Y**, Windows authentication was used. Make sure that the new installation uses the same authentication type (Microsoft SQL Server authentication or Windows authentication) as the previous installation.

Monitoring ALM Server Fails

When running one of the Java-based tools to monitor ALM you receive the following message:

"Not enough storage is available to process this command."

This problem is caused because the JVM running ALM Server is running with a service account.

Choose one of the following solutions, depending on which tool you are running:

- **jmap and jstack.** See the suggestion in the following link:

  http://stackoverflow.com/questions/906620/jstack-and-not-enough-storage-is-available-to-process-this-command

  You will be required to download the pstools tool from the following address:

  http://technet.microsoft.com/en-us/sysinternals/bb897553

- **jconsole and jvisualvm.** Download the following tool from the following address:

  http://www.iopus.com/guides/srvany.htm

  Also refer to the following Microsoft article: http://support.microsoft.com/kb/137890

Upgrade Preparation Troubleshooting

General Validation

Supported Database Version

The verification process checks that the project schema is stored in a supported database server. If the verification process detects that the database server version is not supported, it displays a warning. For details about the database servers versions supported by ALM, refer to the *Readme*.

> **Note:** The supported environment information in the *Readme* is accurate for the ALM12.00 release, but there may be subsequent updates. For the most up-to-date supported environments, refer to the HP Software Web site using the following URL: http://www.hp.com/go/TDQC_SysReq.

Valid Database User Schema Name

The upgrade mechanism does not support databases that include special characters in the database name. If the verification process finds special characters, you must remove them. For SQL databases, periods are also not supported in the database user schema name.

To remove special characters from database names:

1. Deactivate the project.
2. Ask your database administrator to rename the database user schema to a name that does not include special characters, or periods for SQL databases.
3. Remove the project from Site Administration.
4. Update the **Dbid.xml** file to point to the new database user schema name.
5. Restore the project by using the updated **Dbid.xml** file.
6. Run the verification process again to make sure the problem is resolved.

Mixed Table Ownership

ALM can connect to Microsoft SQL server by using SQL authentication or Windows authentication.

For each of these methods, a different user owns the tables of a project:

- **SQL Authentication.** Table owner is the user `td`.
- **Windows Authentication.** Table owner is the user `dbo` (a user mapped to the operating system user that runs the ALM server).

If you create a project with one type of authentication (for example, SQL), and then restore it with the other type of authentication (for example, Windows), these tables cannot be accessed. In this case, new tables are created with owners that are different from those of the old tables. You will not be able to work with the project. It is likely that the upgrade will fail.

To prevent this problem, the duplicate ownership validator checks that the owner of all of the tables in the project database user schema matches the connection method that ALM is using to connect to the server.

To fix table ownership manually, do one of the following:

- **SQL Authentication:** Run the following query to make `td` the table owner:

```
EXEC sp_changeobjectowner '<table name>', 'td'
```

- **Windows Authentication:** Run the following query to make `dbo` the table owner:

```
EXEC sp_changeobjectowner 'td.<table name>', 'dbo'
```

Repository over Database Feature

The **Repository over Database** feature is not supported in Quality Center 10.00 or in ALM versions 11.00 and later.

If you use this feature in Quality Center 9.2, you should migrate the repository from the database to the file system (available from Quality Center 9.2 Patch 12) before upgrading the project to Quality Center 10.00, and then upgrade the project to ALM 11.00.

For more information about the tool for migrating the project repository from the database to the file system, see the *ReadMe* files for Quality Center 9.2 Patch 12. The verification process checks whether the project is using the **Repository over Database** feature. If the project is using the feature, the validator displays a warning.

Version Control Validation

- **Legacy version control projects.** Integration with external version control tools is not supported in ALM12.00. Quality Center version 10.00 and ALM include a built-in version control functionality to support your projects. To work with projects from Quality Center 9.2 that use version control, you must first upgrade to ALM 11.00, migrate legacy version control data, and then upgrade to ALM12.00.

- **Version control enabled projects.** Version control enabled projects cannot be upgraded to ALM12.00 while there are checked out entities. The verification process checks that there are no checked out entities. If there are checked out entities, they must be checked in. To determine if there are checked out entities, see HP Software Self-solve knowledge base article KM00470884 (http://h20230.www2.hp.com/selfsolve/document/KM00470884). (Requires HP Passport sign-in credentials.)

Database Permissions

To enable an upgrade to the current ALM version, the project schema requires a set of minimum required permissions. The verification process makes sure that both the project user and the administrator user have all the privileges needed to perform the upgrade.

Text Search Configuration

Quality Center versions 9.0 and later support the database text search feature. However, not all databases are configured to support this feature. If your database does support text search, ALM installs the required components when creating a new project database. ALM also activates the text search for the new database. The verification process checks whether your project has the text search feature enabled, and that it is configured correctly.

The verification process validates the following:

Validity of the Text Search Configuration

The verification process checks that text search components are installed and are valid on the database server. If a database server is text search-enabled in the DB Servers tab in Site Administration, text search must also be enabled on the Oracle or SQL database server. If the verification process detects that text search is not enabled or configured incorrectly on the Oracle or SQL database server, the upgrade process does not run until you manually repair the problem.

We recommend that you ask your database administrator to reconfigure text search on the Oracle or SQL database server. Alternatively, as a workaround, you can disable text search for the database server from Site Administration.

To disable the text search for the database server:

1. Run the following query on your Site Administration schema:

   ```
   update <SA Schema>.dbservers set db_text_search_enabled = null where dbserve
   r_name = '<DB logical name>'
   ```

2. Restart the ALM server.

3. Run the repair process for your projects.

4. When the repair process completes, run the following query:

   ```
   update <SA Schema>.dbservers set db_text_search_enabled = 'Y' where dbserver_
   name = '<DB logical name>'
   ```

5. Restart the ALM server.

Only Valid Fields Configured Under "Text Search"

The verification process checks that only valid fields are defined as searchable. You can enable the text search only for specific entities, and only on fields of the type string or memo. The following entities are supported: BUG, COMPONENT, COMPONENT_STEP, DESSTEPS, REQ, TEST, BPTEST_TO_COMPONENT, and CYCLE. Any other configuration could cause functionality problems during upgrade or customization. This problem is fixed automatically by the repair process.

Text Search Validation for Oracle Database Server

For an Oracle Database server, the verification process checks the following:

- **Validity of Text Search Indexes.** The verification process checks that database text search indexes are valid. Invalid text search indexes can cause functionality problems and even upgrade failure in ALM. If the verification process detects an invalid index, try to recreate the index by dropping it from the schema and creating it again. In Site Administration, click the **Site Projects** tab. Select the relevant project and click the **Enable/Rebuild Text Search** button. If this procedure returns an error, consult your database administrator or contact HP Support.

- **Validity of Project Database User Permissions.** The verification process checks that the project database user has the required permissions to work with text search. When text search is installed on the database, the role CTXAPP is created automatically. ALM requires that this role be granted to all projects database users that support text search. (ALM grants the CTXAPP role automatically when creating the project or enabling the text search for a project.) If this role is not granted to the project database user (configured to support text search), the verification process returns a warning. In these cases, ask your database administrator to grant the required role to the project database user.

Text Search Validation for Microsoft SQL Database Server

The verification process checks that the project database user schema enables the text search feature. To work with text search on SQL project, you need to enable the text search on the database.

To enable text search on the database:

1. Select the database from the SQL server Enterprise Manager.

2. Right-click the database name.

3. Select **Properties/Files**.

4. Select **Use Full-Text Indexing**.

Schema Validation

The verification process helps to ensure that the project database user schema is correct and configured as expected.

The verification process performs two types of schema verifications:

- **Schema Correctness.** Checks that the project database schema includes all of the required schema objects, as defined in the expected database user schema for the project. This verification ensures that all of the required entities exist and are defined as expected. It also ensures that there are no extra entities defined on top of the schema.

- **Alignment to the current version.** Notifies you about differences in the project database user schema caused by internal changes made in Quality Center or ALM. In this way, the verification process aligns the schema with the latest internal changes to the schema made in preparation for the upgrade.

The verification process displays warnings in the verification report if it finds the following:

- Extra entities defined. For example, Table, Column, Trigger, View, and Sequence.
- Differences from the expected definitions. For example, Column Size and Index Attributes.
- Missing objects.

Schema differences found by the verification process can cause upgrade failures or usage problems. As long as the verification process still finds these differences, an upgrade to the current ALM version will not start.

Note: Many of the schema changes can be fixed automatically by the repair process.

The following sections contain possible warnings, grouped by the different database objects, that the verification process can display in the verification report:

Tables 135
Columns 136
Indexes and Constraints 138
Triggers 141
Sequences 141
Internal Quality Center Changes 142

Tables

Database tables can contain the following warnings:

Extra Table

The ALM schema should contain only the tables that are defined in the schema configuration file. Adding extra tables on top of the schema is not supported and might cause future problems with ALM.

Problem: If the verification process finds extra tables that were added manually to the schema, it generates an **Extra Table** warning.

Note: This problem requires manual repair. The repair process cannot fix it.

Solution: Do one of the following:

- **Change the Schema.** If you use the table, copy it to a different schema. If you do not use the table, delete it. Before taking either action, back up the schema and consult your database administrator. For details, see "Changing the Database User Schema" on page 149.

- **Use the Exception File.**

  > **Note:** If the project database is case sensitive, the table name must be the same in both the database and the exception file.

> **Note:** Not recommended: Instruct the upgrade to ignore this problem.

Missing Table

The verification process checks that all of the tables defined for the project schema actually exist (according to the tables of each Quality Center/ALM version).

Problem: If a table is missing, the verification process generates a **Missing Table** warning.

Solution: Do one of the following:

- See "Changing the Database User Schema" on page 149.

- Run the repair process to create the missing table. Although you can use the repair process to add these objects, we recommend that you contact HP Support to make sure that the missing objects are not just symptoms of a bigger problem.

Columns

Database columns can contain the following warnings:

Extra Column

The verification process checks that each table includes the required columns, as defined for the expected database user schema and version. The schema should not include extra columns. Extra columns in a table might cause upgrade failure or functionality problems.

Problem: If the verification process detects an extra column (that does not exist in the database user schema definitions) in one of the tables, it generates an **Extra Column** warning.

> **Note:** This problem requires manual repair. The repair process cannot fix it.

Solution: Do one of the following:

- **Change the Schema.** If you have an internal implementation that requires extra table columns, move the extra columns to a different table in a different schema. If you do not use a particular column, delete it. Before taking either action, back up your schema and consult your database administrator. For a more detailed explanation, see "Changing the Database User Schema" on page 149.

- **Use the Exception File.**

> **Note:** Not recommended: Instruct the upgrade to ignore this problem.

Column Size Mismatch

The verification process checks that all the table columns are defined as expected. This validation ensures that the column size matches the expected size as defined for each table column. This verification excludes user-defined fields, whose size can be customized through project customization.

Some column mismatch warnings are caused by internal changes made in Quality Center 10.00 that are fixed by the repair process automatically. For details, see "Internal Quality Center Changes" on page 142.

Problem A: Size is bigger than expected. If the column size is bigger than expected, decrease the column size to the required size manually. Because this operation can cause data loss, it is not performed automatically by repair process.

> **Note:** This problem requires manual repair. The repair process cannot fix it.

Solution A: Consult your database administrator to resolve this issue. For risks involved in changing the database user schema, see "Changing the Database User Schema" on page 149.

Problem B: Size is smaller than expected. If the column size is smaller than expected, the repair process fixes the problem automatically by increasing the column size to the expected size.

Solution B: Run the repair process to increase the current size to the required size.

Column Precision Mismatch

In an Oracle Database, "precision" is the term used to define the size of fields with the INTEGER type.

Problem: The verification process generates a warning if the precision defined for a certain column is smaller than expected.

Solution: Run the repair process to increase the current precision to the required precision.

Column Type Mismatch

Changing a column type causes the upgrade to fail, and can cause major functionality problems.

Problem: The verification process generates a **Column Type** warning if the column type has changed.

> **Note:** This problem requires manual repair. The repair process cannot fix it.

Solution: Consult your database administrator to resolve this issue. For risks involved in changing the database user schema, see "Changing the Database User Schema" on page 149.

Column Nullability Mismatch

One of the attributes that is defined for a column is whether it can accept null values. A null is the absence of a value in a column of a row. Nulls indicate missing, unknown, or inapplicable data. If you have defined a NOT NULL or PRIMARY KEY integrity constraint for a particular column, you cannot insert rows into the column without adding a value.

Problem: The verification process compares the required definitions for each column in the expected database user schema to the project database user schema. If it encounters differences in the column NULL attribute definition, it generates a **Column Nullable** warning.

Solution: Run the repair process. The repair process runs a query to modify the column attributes to the expected attributes.

If the column includes NULL values, the repair process cannot update the column attribute to NOT NULL (if this is the required attribute) for the column. Ask your database administrator how to remove the NULL values from the column. After removing the NULL values, run the repair process again. For details, see "Changing the Database User Schema" on page 149.

Identity Column

The IDENTITY property is one of the attributes defined for columns in Microsoft SQL server.

Problem: As part of the verification for the columns attributes, the verification process might find a column IDENTITY property that is not configured as expected.

> **Note:** This problem requires manual repair. The repair process cannot fix it.

Solution: Change the IDENTITY property of the column to the expected configuration (according to the output from the verification process report) manually. Consult your database administrator to resolve this issue. For details, see "Changing the Database User Schema" on page 149.

Missing Column

If a column is missing from a table, run the repair process or contact HP Support.

Problem: If the verification process finds that a column is missing from one of the tables, it generates a **Missing Column** warning.

Solution: Do one of the following:

- Run the repair process to fix the problem.
- See "Changing the Database User Schema" on page 149.

Indexes and Constraints

A database index is a data structure that improves the speed of operations in a table. You can create indexes using one or more columns, providing the basis for both rapid random lookups and efficient ordering of access to records. Database Constraints are constraints on the database that require relations to satisfy certain properties.

Database indexes and constraints can cause the following validation warnings:

Extra Index

The ALM schema should include only those indexes defined in the required schema configurations.

Problem: If the verification process finds an index that is not defined in the required schema configuration, it generates an **Extra Index** warning.

> **Note:** This problem requires manual repair. The repair process cannot fix it.

Solution: Remove the extra indexes manually. Consult with your database administrator to resolve this issue. For details, see "Changing the Database User Schema" on page 149.

Some **Extra Index** warnings are caused by internal changes made in Quality Center 10.00. These extra indexes are no longer used by ALM, and are removed by the repair process. For details, see "Internal Quality Center Changes" on page 142.

Extra Constraint

The ALM schema should include only those constraints defined in the required schema configurations.

Problem: If the verification process finds a constraint that is not defined in the required schema configuration, it generates an **Extra Constraint** warning.

> **Note:** This problem requires manual repair. The repair process cannot fix it.

Solution: Remove the extra constraint manually. Consult with your database administrator to resolve this issue. For details, see "Changing the Database User Schema" on page 149.

Index Uniqueness Mismatch

A unique index guarantees that the index key contains no duplicate values. As a result, every row in the table is unique. Specifying unique indexes on ALM data tables ensures data integrity of the defined columns. In addition, it provides helpful information that is used as a query optimizer.

Problem: If the index uniqueness attribute does not have the expected value, the verification process generates an **Index Uniqueness Mismatch** warning.

You cannot create a unique index, unique constraint, or PRIMARY KEY constraint if duplicate key values exist in the data. The verification process performs these data validations. If a table has duplicate values or IDs, based on the index definitions on that table, the verification process also displays the duplication in the verification report. In this case, the repair process automatically fixes the duplication problem before creating the unique index.

Solution: Run the repair process to fix the problem.

Index Clustered

In Microsoft SQL, index type can be classified as clustered or non-clustered. The verification process compares the required definitions for each index in the expected database user schema to the project database user schema.

Problem: If the verification process finds differences in the index clustered attribute definition, it generates an **Index Clustered** warning.

Solution: Run the repair process to fix the problem.

Missing Constraint

Constraints are rules that the database enforces to improve data integrity.

Problem: If the verification process finds a constraint that should be defined as missing, it generates a **Missing Constraint** warning.

Solution: Run the repair process to fix the problem.

Missing Index

The verification process checks that all the required indexes (as defined in the expected database user schema) exist in the projects database user schema.

Problem: If the verification process does not find all the required indexes in the projects database user schema, it generates a **Missing Index** warning.

Solution: Run the repair process to fix the problem.

Index Changed

The verification process checks that the indexes are defined according to the expected database user schema.

Problem: If the verification process finds an index that is not defined according to the expected database user schema, it generates an **Index Changed** warning.

This warning can indicate the following problems:

- Function in a function-based index is different than expected.
- Index is not defined on the expected columns.

Solution: Run the repair process to fix the problem. The repair process removes the index, and then recreates it, based on the required definitions for this index.

Index Order Changed

The verification process checks that the order of the columns in the index definition has not changed.

Problem: If the order of the columns in the index definition has changed, the verification process generates an **Index Order Changed** warning.

Solution: Run the repair process to fix the problem. The repair process removes the index, and then recreates it, based on the required definitions for this index.

Triggers

A database trigger is procedural code that is automatically executed in response to certain events on a particular table in a database.

Database triggers can contain the following warning:

Extra Trigger

Extra triggers can cause upgrade failures and functionality problems.

Problem: If the verification process finds an extra trigger, it generates an **Extra Trigger** warning.

> **Note:** This problem requires manual repair. The repair process cannot fix it.

Solution: Before upgrading, back up your database schema and remove the extra triggers manually.

Because extra triggers can cause upgrade failures, the upgrade process cannot ignore this warning by using the Exception file. For details, see "Changing the Database User Schema" on page 149.

Sequences

A sequence is an Oracle object that acts as a generator that provides a sequential series of numbers.

Database sequences can contain the following warnings:

Extra Sequence

ALM schemas should contain only the sequences that are defined in the schema configuration file.

Problem: If the verification process finds an extra sequence, it generates an **Extra Sequence** warning.

> **Note:** This problem requires manual repair. The repair process cannot fix it.

Solution: Do one of the following:

- **Change the Schema.** Move the sequence to a new database user schema. Before doing so, consult with your database administrator. For details, see "Changing the Database User Schema" on page 149.
- **Use the Exception File.**

> **Note:** Not recommended: Instruct the upgrade to ignore this problem.

Missing Sequence

Problem: If the verification process finds that one of the sequences that should be defined on the ALM schema is missing, it generates a **Missing Sequence** warning.

Solution: Do the following:

- Run the repair process to fix the problem.
- See "Changing the Database User Schema" on page 149.

Incorrect Sequences

Problem: Sometimes the Oracle object sequence numbers become incorrect, for example, if an export of the database is done on a live activated project, in which users are still modifying tables. If the verification process finds that Oracle sequences objects are not fully synchronized with ALM schema table IDs, the verification process generates an **Incorrect Oracle sequences found** warning.

Solution: Run the repair process to fix the problem.

Internal Quality Center Changes

For upgrade from Quality Center 9.2: As a result of internal changes in Quality Center 10.00, a set of updates needs to be applied to the schema as part of the preparation for the upgrade to ALM.

To apply the updates to the schema, perform the following processes:

Verification Process

If the verification process finds any internal differences, it generates warnings in the verification report. The repair process fixes them automatically.

The verification process checks for the following internal changes:

Type	Problem	Element	Comment
Column	Size mismatch	`COMMON_SETTINGS.CSET_NAME`	Expected column size is 240. Actual size is 70.
Column	Size mismatch	`REQ.RQ_REQ_PRIORITY`	Expected column size is 255. Actual size is 70.
Column	Size mismatch	`REQ.RQ_REQ_TYPE`	Expected column size is 255. Actual size is 70.
Column	Size mismatch	`REQ.RQ_REQ_AUTHOR`	Expected column size is 255. Actual size is 70.
Column	Size mismatch	`REQ.RQ_REQ_PRODUCT`	Expected column size is 255. Actual size is 70.
Column	Size mismatch	`REQ.RQ_REQ_REVIEWED`	Expected column size is 255. Actual size is 70.
Column	Size mismatch	`REQ.RQ_REQ_STATUS`	Expected column size is 255. Actual size is 70.

Type	Problem	Element	Comment
Index	Missing	ALL_LISTS.AL_ABS_PATH_COV_IDX	
Index	Missing	BUG.BG_COMPOUND_IDX	
Index	Missing	CYCLE.CY_FOLDER_IDX	
Index	Missing	REQ.RQ_REQ_STATUS_IDX	
Index	Missing	RUN.RN_CYCLE_IDX	
Index	Missing	STEP.ST_RUN_IDX	
Index	Missing	TEST.TS_SUBJECT_IDX	
Index	Extra	BUG.BG_DETECTED_BY_LWR_IDX	
Index	Extra	BUG.BG_STATUS_LWR_IDX	
Index	Extra	BUG.BG_PRIORITY_LWR_IDX	
Index	Extra	BUG.BG_RESPONSIBLE_LWR_IDX	
Index	Index changed	REQ_COVER.RC_ENTITY_ID_IDX	
Index	Index changed	RUN.RN_TEST_ID_IDX	
Index	Index changed	RUN.RN_TESTCYCLE_IDX	
Function-based indexes - relevant only for SQL server.	Extra index	COMMON_SETTINGS.CS_COVER_LWR_IDX	
Function-based indexes - relevant only for SQL server.	Extra index	HOSTS.HOSTS_LWR_IDX	
Function-based indexes - relevant only for SQL server.	Extra index	HOSTS_IN_GROUP. HG_COVER_LWR_IDX	

Type	Problem	Element	Comment
Function-based indexes - relevant only for SQL server.	Extra index	`HOST_GROUP. GH_LWR_IDX`	
Function-based indexes - relevant only for SQL server.	Extra index	`USERS.US_USERS_LWR_IDX`	

Repair Process

The repair process fixes these internal differences in the following way:

- **Column Size.** Increases the size of columns to the required size.
- **Index Definition.** Removes extra indexes. It also recreates missing indexes and indexes that were defined differently.
- **Extra Function-based Indexes.** Microsoft SQL Server only. Removes obsolete function-based indexes.

Before beginning the upgrade, run the repair process on each project.

Data Validation

Duplicate Values

Some fields (or a combination of fields) must be unique in given tables. This constraint is enforced by the creation of a unique index on these fields. For example, the combination of fields TS_SUBJECT and TS_NAME, which represent the ID of the test's parent folder and test name, must be unique. It is not possible to create two tests with the same name under the same folder. In rare cases, a corrupted database contains duplicate values in these fields.

Problem: The verification process checks that all unique indexes exist (and therefore enforce unique values). If the verification process finds duplicate values, it does not permit the upgrade to run on the project.

The verification report specifies the fields in which there are duplications and number of duplicate values found, as shown below.

Duplicate Values
Looks for records in selected tables that have duplicate field values. Values must be unique.
The Repair tool automatically handles duplicate values.

#	Table	Columns	# Duplicate items

Solution: Automatic Repair. Run the repair process to automatically handle the duplicate values. The repair process renames the duplicate values to resolve the problem.

Duplicate IDs

Most tables have a unique primary key, usually a unique single column. If there are duplicate values in this field, the primary key is not created.

For example, in a table called test, the column TS_TEST_ID represents the test ID, which is unique. In rare cases, a corrupted database contains duplicate IDs.

Problem: The verification process checks that all IDs in a table are unique. If it finds duplicate IDs, it does not permit the upgrade to run on the project.

The verification report specifies the fields in which there are duplicate items and values, as shown below.

Duplicate IDs

Looks for records in selected tables that have duplicate ID field values.
The Repair tool automatically deletes the duplicate records.

#	Table	Column	# Duplicate Items
1	TEST	TS_TEST_ID	2

Solution: Automatic Repair. The repair process automatically deletes one of the records with a duplicate ID.

> **Caution:**
>
> This option assumes that the entire record is duplicated, and that the duplicated record is not accessible from the ALM user interface. Because there can be exceptions, we recommend that you use this option only after verifying manually that this record deletion will not cause data loss.

Tree Inconsistencies

The verification process checks four different entity trees (hierarchical representation of entities):

- Test Plan tree
- Business Components tree
- Requirement tree
- Test Lab tree

The verification process checks that the data in the tree tables is correct.

> **Caution:** Do not manually fix any problems related to tree data. The repair process fixes them automatically.

Problem: The verification process checks for the following types of problems:

- **Corrupted Path.** This is an internal ALM field that contains a string that represents the order of each node in the tree.
- **Wrong Number of Children.** This is an internal ALM field that contains the number of children for each node in the tree.
- **Orphan Records in Trees.** By definition, orphan records do not have parent records. As a result, you cannot access them through the ALM user interface.

Solution: Automatic Repair. Run the repair process to automatically fix any problems related to tree data.

> **Caution:** Before beginning the automatic repair, review each orphan record carefully. If the verification process finds an orphan record, it deletes it (and all its descendants) from the tree automatically.

Views

Database views can contain the following warning:

Extra Views

ALM schemas should contain only the views that are defined in the schema configuration file.

Problem: If the verification process detects extra views that were added manually to the schema, it displays an **Extra Views** warning. Adding extra views on top of the schema is not supported and could cause problems.

> **Note:** This problem requires manual repair. The repair process cannot fix it.

Solution: Do one of the following:

- **Change the Schema.** If you use the view, copy it to a different schema. If you do not use the view, delete it. Before taking either action, back up your schema and consult your database administrator. For details, see "Changing the Database User Schema" on page 149.
- **Use the Exception File.**

> **Note:** Not recommended: Instruct the upgrade to ignore this problem.

Orphaned Entities

The verification process checks for entity data that is missing corresponding parent data. For example, the following entities might be missing corresponding test configurations or test criteria:

- Test configuration coverage
- Criteria coverage
- Run criteria
- Runs
- Test instances

Caution: Do not manually fix any problems related to orphaned entities. The repair process fixes them automatically.

Problem: In version-controlled projects, deleting a test configuration or test criteria did not delete corresponding entities after checking in. This caused incorrect coverage calculation.

Solution: Automatic Repair. Run the repair process to automatically fix any problems related to orphaned entities created by this problem.

Missing Entities

The verification process checks for data that is missing. For example, the following entities might be missing:

- Test configurations
- Test criteria

Caution: Do not manually fix any problems related to missing entities. The repair process fixes them automatically.

Problem: The upgrade process can detect that certain entities are missing based on information that exists in related tables.

Solution: Automatic Repair. Run the repair process to automatically fix any problems related to missing entities created by this problem.

Missing Lists and/or List Values

The verification process checks that all of the fields of List type are associated with a list.

Problem: If a list and/or its values are missing, the verification process generates a warning about missing lists or missing list values.

Solution:

Run the repair process to create the missing list and/or its values.

Missing lists are re-created with the name: **AUTO_GENERATED_LIST_NAME_<*unique_ number*>**

After running the repair process, do the following in **Customization > Project Lists**:

- Rename any lists whose names are prefixed by **AUTO_GENERATED_LIST_NAME_**.
- If necessary, add any list values that are missing.

Tip: Although you can use the repair process to add these objects, we recommend that you contact HP Support to make sure that the missing objects are not just symptoms of a bigger problem.

Encrypted Values

Some fields are saved in the database in an encrypted state. Encryption is done using confidential data passphrases.

Note: This is an issue with Performance Center and Lab Management projects.

Problem: The verification process checks that all encrypted data can be decrypted with the current confidential data passphrases. If the verification process finds encrypted values that cannot be decrypted, the project is not upgraded.

The verification report specifies the fields that cannot be decrypted.

Solution: If verifying the LAB_PROJECT fails due to a problem with the Confidential Data Passphrase, do one of the following:

- Make sure that the same Confidential Data Passphrase is defined on the original server on which the LAB_PROJECT was located, as well as on the server to which it is being restored.
- Perform the following steps:

  a. In Site Administration: Before attempting to verify the LAB_PROJECT again, navigate to the **Lab Management** tab and clear all encrypted field values from the project by running the following queries:

- For a Microsoft SQL Database

```
update td.LAB_DIAGNOSTICS_SERVERS set DIAG_SVR_PASSWORD = ''

update td.LAB_AUT_HOSTS set AUTHOST_PASSWORD = ''

ALTER TABLE td.LAB_HOSTS DISABLE TRIGGER ALL

update td.LAB_HOSTS set HOST_PASSWORD = ''

ALTER TABLE td.LAB_HOSTS ENABLE TRIGGER ALL
```

- For an Oracle Database

```
update <schema name>.LAB_DIAGNOSTICS_SERVERS set DIAG_SVR_
PASSWORD = ' '

update <schema name>.LAB_AUT_HOSTS set AUTHOST_PASSWORD = ' '

update <schema name>.LAB_HOSTS set HOST_PASSWORD = ' '
```

b. Proceed with the verify, repair, and upgrade of your LAB_PROJECT.

c. Login to Lab Management and update the passwords of the AUT Hosts, Diagnostics Server and Standalone Unix Load Generators. For information on working in Lab Management, refer to the *HP ALM Lab Management Guide*.

Changing the Database User Schema

This section describes the problems that require manual repair (cannot be fixed automatically by the repair process), and recommends solutions for these problems. If you encounter any of the problems mentioned below, consult with your database administrator or contact HP Support for further guidelines to resolve these problems before upgrading.

The stability of the new database upgrade component depends on the database user schema validity. We recommend that you not use the Exception file to change the database user schema.

This section includes:

Missing Database Objects 150

Missing List Warning 150

Sequences Warning 150

Changed Database Objects 150

Extra Database Objects 151

Missing Database Objects

Missing database objects can be symptoms of a bigger problem.

Problem: Missing database objects (for example, tables and indexes) can yield unexpected and unwanted behavior.

Solution: Although you can use the repair process to add these objects, we recommend that you contact HP Support to make sure that the missing objects are not just symptoms of a bigger problem.

Missing List Warning

User-defined fields of List type must be associated with lists.

Problem: If a list is missing for a user-defined field, the verification process generates a **Missing List** warning.

Solution: Contact HP Support for instructions on changing the data type of the user-defined field from List to String in the SYSTEM_FIELD table.

> **Caution:** Contact HP Support before attempting to fix the problem manually.

Sequences Warning

An internal mechanism manages IDs and other system numerators. The table SEQUENCES holds the name of the table or other entity whose numeration is being tracked as well as its highest current value.

Problem: If one of the records is missing in this table, or if one of the values is incorrect, the verification process generates a **Sequences** warning.

Solution: The repair process fixes the problem automatically.

> **Caution:** We strongly recommend that you not attempt to fix the problem manually.

Changed Database Objects

Any of the following cases is defined as a Changed Database Object:

- Data type of a column was changed
- Length of a column was changed
- Nullability of a column was changed
- Column is defined as identity although it should not be defined as such, or vice versa

Problem: A changed column data type can result in incorrect behavior on the server side.

Solution: To avoid this behavior, make sure that you have resolved all data type and length concerns before beginning the upgrade.

For every changed database object that is found, do the following:

1. Create a new column with the required attributes as originally defined by the ALM server.
2. Move the data from the old column to the new one.

   If you cannot move the data (for example, move strings to numeric columns, or move large data to smaller fields), contact HP Support.
3. Remove the old column.
4. Rename the new column to the original column name.

Extra Database Objects

ALM has various customization options. One option is to add user-defined fields (UDFs). You can add a UDF by using either the project customization user interface or through OTA (Open Test Architecture).

Problem: Any other addition to the database user schema (for example, defining extra objects on top of ALM schema) can result in a failure, such as the following:

- **Name Conflict.** If the later version happens to include a name that you added for a proprietary database object (for example, a table, view, or column), the two names will be in conflict.
- **Copy and Synchronize Failure.** If the database user schema contains extra or missing database objects, some ALM mechanisms for copying and synchronizing might fail.
- **Extra Triggers.** If the database contains extra triggers, some update operations might fail.

Solution:

For each extra database object that is found, perform the corresponding solution:

- **Move extra columns to newly created tables.**

  To make sure a new table has a one-to-one relationship with the original table, define the primary key of the new column in the new table with the value of the primary key of the original column in the original table.
- **Move extra tables to a different database user schema.**

  These extra tables include those tables created above. You might need to amend the proprietary application data access of these tables. You can still access these tables from within the ALM database connection by specifying the full name.

Examples:

- Oracle

```
<schema name>.<table name>
```

- SQL Server

```
<database name>.td.<table name>
```

To be able to see these tables, you must grant the necessary permissions for the database user schema.

- **Move extra views to a different database user schema.**

  Like extra tables, these views can be moved to a different database user schema. In addition, you must grant reading permissions to the newly created database user schema on the database user schema objects.

- **Remove referential integrity between customer database objects and ALM database objects.**

  This removal includes no data loss.

- **Remove extra triggers before the upgrade, and, only if truly necessary, restore them after the upgrade.**

  No data loss is involved. The upgrade process includes data upgraders that perform some data manipulations (for example, removing duplicate values, fixing tree structures, and so on).

  Your triggers will not be invoked on these update events.

  As a result, you need to do the following:

  a. Ask HP Support for information about the data upgrader activity.

  b. Review the information about the data upgrader activity.

  c. Decide on which proprietary updates you need to perform.

- **Remove extra indexes.**

  You can log all indexes before the upgrade, and (only if necessary) restore them after the upgrade. No data loss is involved.

- **Oracle Database only: Move extra sequences to a newly created database user schema.**

To access the extra sequences from the database user schema, you must grant ALM the required permissions. When moving these sequences, set them to start with the number they reached at the time of the move.