ALM 12.5 user guide

# Version History & Table of Contents

## Version History

|  |  |  |
| --- | --- | --- |
| Version | Date | Changes |
| 0.1 | 14.06.2016 | This is the first version of the document |
| 0.2 | 29.06.2016 | Updates to the KPI calculation and threshold profiling section. |
| 0.3 | 7.7.2016 | Updates to the test plan and test execution plan sections with focus on how to best support the KPI calculations |
| 0.4 | 23.08.2016 | Removed client specific software references and made document more general. Split references to ALM and QC where appropriate |

|  |  |  |
| --- | --- | --- |
| Version | Date | Client |
| 0.4 | 23.8.2016 | Nordea, Sweden |
| 0.5 | 26.10.2016 | Slight changes for wider audience |
| 0.6 | 27.09.2017 | Further changes for Nordea |
| 0.7 | 28.09.2017 | Further updates based on comments |

## Table of contents

[Version History & Table of Contents 2](#_Toc494365481)

[Version History 2](#_Toc494365482)

[Table of contents 3](#_Toc494365483)

[Key to Indicators 5](#_Toc494365484)

[Summary 6](#_Toc494365485)

[Document Objectives 6](#_Toc494365486)

[Document Audience 7](#_Toc494365487)

[Fundamentals 8](#_Toc494365488)

[The Release 8](#_Toc494365489)

[Defining the right release 9](#_Toc494365490)

[The cycle 9](#_Toc494365491)

[The Scope 10](#_Toc494365492)

[Defining the right scope item – Size and features 11](#_Toc494365493)

[Scope Content 12](#_Toc494365494)

[Scope Item Contents 13](#_Toc494365495)

[Requirements and ALM 13](#_Toc494365496)

[Requirement Preparation – Final Technical Note 13](#_Toc494365497)

[Tests, Test Instances and Test Configurations. 14](#_Toc494365498)

[Tests and test instances in the example project 14](#_Toc494365499)

[Structuring the test plan to work with Scope Items and KPI’s 17](#_Toc494365500)

[Test Lab 18](#_Toc494365501)

[Structuring the test execution plan to work with scope items and KPIs 18](#_Toc494365502)

[Defects 21](#_Toc494365503)

[Recording Defects 21](#_Toc494365504)

[Including defects in release scope 21](#_Toc494365505)

[The Management Module 22](#_Toc494365506)

[Key Performance Indicators and Thresholds 23](#_Toc494365507)

[The release Score Card 24](#_Toc494365508)

[Libraries and Baselines 24](#_Toc494365509)

[Libraries 24](#_Toc494365510)

[Example 24](#_Toc494365511)

[Implementation Details 25](#_Toc494365512)

[Workflow 25](#_Toc494365513)

[Step 1: Determine Release Schedule 25](#_Toc494365514)

[Step 2: Identify and create Scope Items 25](#_Toc494365515)

[Step 3: Build and Load Requirements 25](#_Toc494365516)

[Step 4: Create Tests and Test Sets 25](#_Toc494365517)

[Step 5: Link tests, requirements and defects to scope items 25](#_Toc494365518)

[Step 6: Determine and build KPI’s and KPI 25](#_Toc494365519)

# Key to Indicators

|  |  |
| --- | --- |
|  | **This information is important for project managers.** |
|  | **This information is important for requirement managers and Business Analysts.** |
|  | **This information is important for testers and test managers.** |
|  | **This information important to the whole team.** |

# Summary



“One of the most common misconceptions when purchasing a test management or even a complete ALM tool is that simple installation and minimal configuration of the tool will facilitate successful use and implementation.” - HP Project Topology Best Practices Guide

This document explains how we will plan the software releases and structure data within HP Application Lifecycle Management (ALM) to obtain the maximum benefit from the substantial investment we have made and continue to make, through annual licensing costs, in the HP Application Lifecycle Management Software.

This document is based on two of HP’s own documents from the HP ALM Best Practices Series for ALM Practitioners. The first is “Project Planning and Tracking Best Practices” and the second is “Project Topology Best Practices” from which the quote at the beginning of this document is taken. Both these documents are available from the documentation library within ALM and are strongly recommended reading.

Where appropriate, HP examples have been changed to reference client software and some recommendations for the project setup may refer directly to client programs. **Significant parts of the HP documents are copied verbatim where appropriate**.

## Document Objectives

* Explain and establish the principle of test and release planning in much greater detail than we currently do. This applies to the ALM Project planning and tracking module however the same functions and principles can be applied to data extracted from QC installations.
* Be able to provide all required reporting and tracking from the start of the project.
* Avoid changes to the reporting mid-way and remove (as far as possible) documents that are open to interpretation.
* Base everything we report on reliable metrics.
* Leverage the significant investment in ALM by making use of the functionality we are not currently using.
* Where appropriate contribute to the professional development of the client testing community by building expertise in the tool.

## Document Audience

This document is intended for:

* Business Stakeholders who define release objectives and track release performance
* QA Managers who set QA objectives, are responsible for QA execution
* Development Managers who set development objectives, responsible for development
* Organization Executives who track release progress, may be involved in setting objectives, usually a scope
* Project Managers who manage the overall project including the release
* Testers from UAT and Business backgrounds
* Third Party On Shore coordinators and test resources with access to ALM

# Fundamentals



ALM has its own terminology and project planning structure built in and most if not all functionality refers in some way to this underlying terminology.

It is important to develop a shared understanding of what these terms mean and how the functionality is used to give us the best chance of a high-quality and pain-free delivery. Understanding and committing to using the functions that ALM provides will give the best experience of the tool. The HP documents define these terminology items as follows.

Release: a well-defined deliverable scheduled to be released for general use on a specific date. The activities leading to the readiness of a release for delivery are managed and tracked to ensure on-time and on-quality delivery.

Cycle: A defined period of time during which specified test sets are executed

Scope item: a tracked portion of the release scope. A scope item may be a business requirement, feature, theme, Change Request (CR) or a backlog item.

Milestone: a significant point on a release timeline for which objectives are set, managed, and tracked.

Key Performance Indicator (KPI): a performance measure on a release activity, used to evaluate status based on preset objectives and to analyze bottlenecks.

Threshold: a means for specifying release objectives. A threshold specifies an acceptable value for a KPI measurement on a specific date.

The subsequent sections will deal with each of these in turn and focus on how we should apply the concepts and functions to our project or program.

## The Release



Within ALM “The Release” is the highest level management object and everything flows from the release downwards. Likewise all objects within ALM, tests, requirements, test results and defects should be attributable to a release.

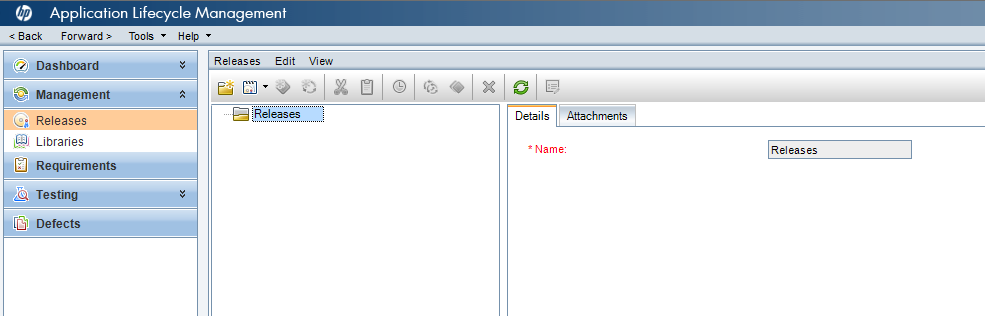


Figure 1: The empty ALM Release Management Module in the example project

### Defining the right release

A well-defined release takes schedule and scope into account. To define the right release for our needs, we should start by setting release start and end dates.

A release’s duration should span over the time that release activities are carried out and tracked within the organization.

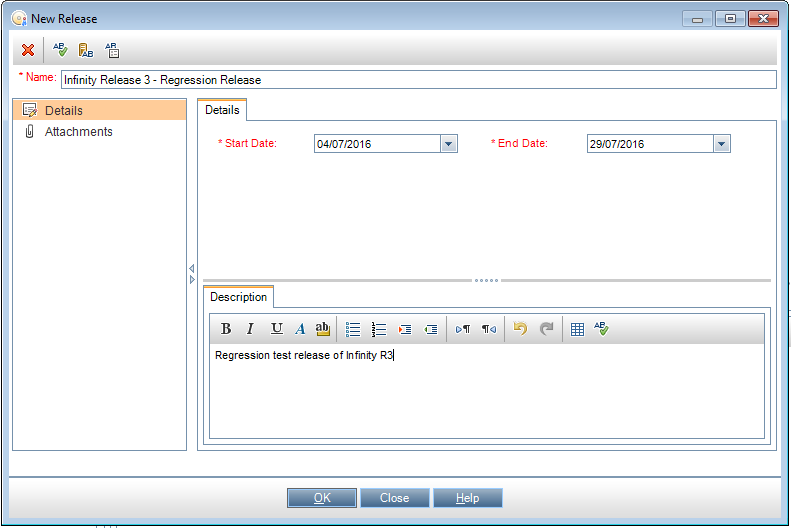


Figure 2: Creation of the first ‘ALM’ Release

For the purposes of the example project we will have two ALM releases. One for regression testing spanning 1st-29th July and one for additional functionality spanning 4th August – 2nd September.

The next section will deal with the definition of the scope of those releases and demonstrate how to populate the scope items.

## The cycle

Within a release there are “cycles” which can be used to break up the test activities into smaller units. A cycle may be two weeks long on an agile project, or two months long on a larger waterfall delivery.

A cycle is defined in the ALM manual as *“A set of development and quality assurance efforts performed to achieve a common goal based on the release timeline. After you define a release, you add cycles to the release. You can then assign requirements, defects, and test set folders to the cycles”*

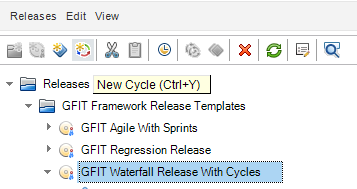


Figure 3: The Creation of a Cycle

## The Scope

A basic factor in every release is the definition of the release scope. In Project Planning and Tracking releases, the scope is defined by means of scope items. A scope item represents a unit within the release deliverable(s). It may be a business requirement, feature, theme, Change Request (CR) or a backlog item that is to be delivered as part of the release.



Within and Agile project the scope item could be a user story or epic.

Scope items are the units tracked in a Project Planning and Tracking release. Well-defined scope items are the basis for setting up tracking and analysis reports that will provide release stakeholders with the most value. Ask yourself “what am I likely to be asked is working?”

The name of a scope item is set by the user. It is a free text field and is best used to represent or encapsulate a core area of functionality that the business would recognize. “Enable User Administration” would be a good example of a scope item, “Enable end to end processing of Corporate Action DVCA Cash Dividend” would be another good descriptive title. Note that we are not implying in the name of the scope object anything about where or when it will be built or tested. It simply exists as a named entity that must be completed for the software to be considered functional.



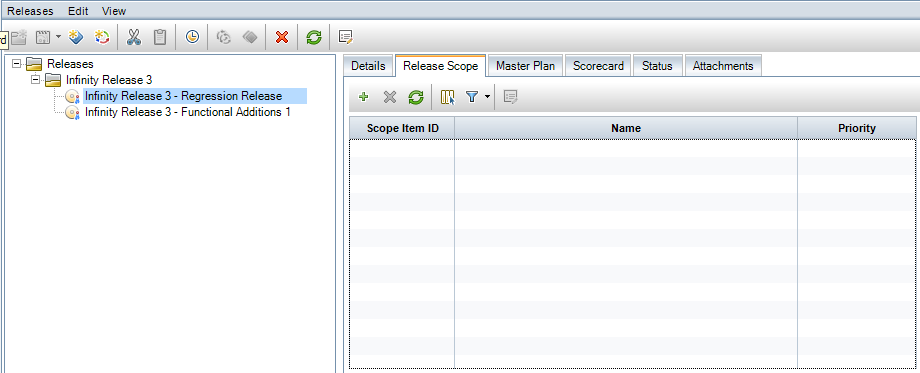


Figure 4: The two releases with no scope items added

Within the example project let’s add both these to the regression release

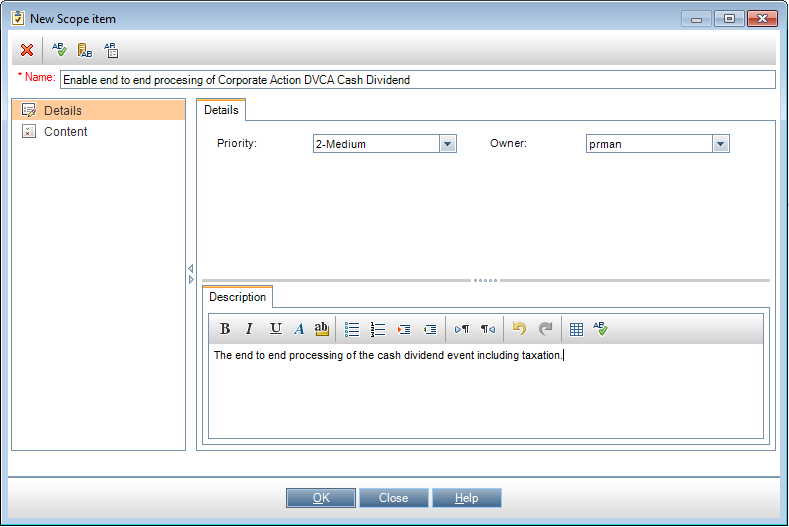


Figure 5: Adding a scope item

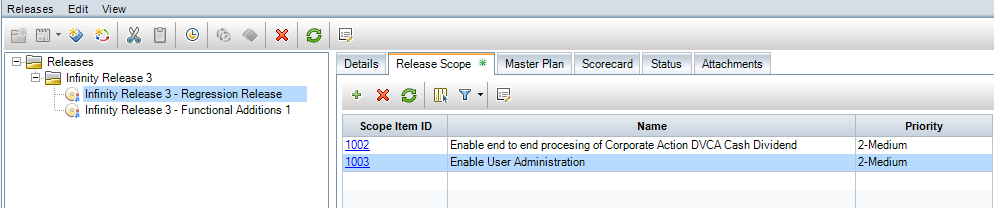


Figure 6: The Regression release with the two scope items added

Bear in mind that at this point we have no “content” to speak of. We have only created container objects in which we can put requirements, tests, defects and so forth but already we are able to see some structure within the project that we will be able to report on. New tabs are visible, “Master Plan”, “Scorecard” & “Status”. At this point these do not have any content.

### Defining the right scope item – Size and features

The extent of a scope item may vary. One scope item may be defined as a small CR that is developed and made ready for delivery in a matter of days and another one may represent an overall feature that is developed throughout the course of an 18-month release.

One may cover a requirement sub-tree containing hundreds of requirements and another may cover a much smaller number of requirements. It is up to the release manager to set up the scope items and to decide on their size.

So, what is the optimal size of a scope item?

* It should represent a unit within the release that is handled as a unit of work by all involved groups (manufacturing/development, QA, delivery, etc.). It is a release deliverable and is tracked as the release progresses.
* A release should contain up to 30 scope items in order for it to be manageable and in order to ease its tracking by the stakeholders. Note that the maximum number of scope items that can be entered in a release is limited by a site configuration parameter and may be modified by the system administrator.
* A scope item should not be too large; setting goals for its related activities, and then tracking activity execution and analyzing and mitigating activity problems, should be manageable tasks. See for information regarding scope items and tracking.

A scope item should include all delivery aspects of the delivery unit it represents. **It should not be split based on entity type or based on cycle or team**. Properly applying KPIs to the scope items (with and without optional filters) in milestones, allows for proper tracking of its related activities.

### Scope Content

With the container objects configured we can now start to look at the content to put within them. This is where we can start to look at requirements, tests, defects, change requests etc.



Of course, in practice project management will define the releases and scope items, laying out the delivery schedule, the business and Business Analysts will define the requirements explaining what is wanted and the test teams will design the tests to meet and prove those requirements on the software that is delivered.

It is important to realize, at this point, if not before, that ALM is not simply a testing tool. It is a complete technical program management tool and as such requires input from all of the above disciplines if it is to work well. Project managers, business resources and testers all need some degree of capability with ALM although it is strongly recommended that a center of expertise be built within the testing community.

There is a great deal that can be done to aid the loading of data from spreadsheets and it is further recommended that the requirement be generated externally, reviewed until agreed and then committed to the ALM system in a clean, one-time operation before generating tests.

# Scope Item Contents

A scope item consists of requirements, tests, test sets, and defects and this section will look at each of these entities in detail.

## Requirements and ALM



The basis for a scope item’s content is the set of requirements that define it; these specify the actual deliverable. **Managing the requirements tree, based on features, allows easy definition and maintenance of the scope items**.

Good requirement preparation takes time and understanding that the text will be used downstream helps BA’s understand why it is so important that the requirements are “clean” when they are loaded to ALM. There is a note on how best to achieve this at the end of this section.

Requirement descriptions should be concise, accurate and focus on the detail of the software feature to be tested.

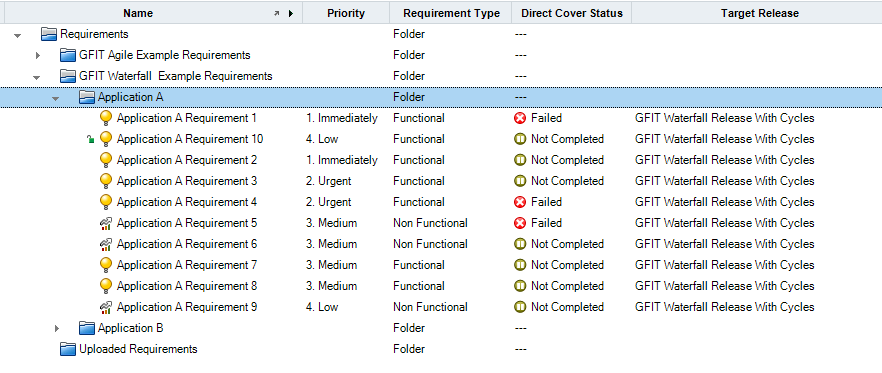


Figure 7: Requirements at a lower level

### Requirement Preparation – Final Technical Note

As a final note when preparing requirements for loading into the system please take care to avoid:



1. Merged cells
2. Tab characters
3. Duplicated numbers or text
4. Non printing characters
5. Leading and trailing spaces
6. Double spaces
7. Using Ampersands and “And” interchangeably

All of the above can cause problems with text processing down the line that are then hard to diagnose and resolve. There are Excel operations that can clean data for you. The functions “Trim” and “Clean” are both recommended. Their use and functionality can be checked in the excel help files but in short they remove trailing and leading spaces, non-printing characters such as tab and most of the potential issues listed.

## Tests, Test Instances and Test Configurations.

As in the case of requirements, tests and test instances may be included in a scope item by means of selecting sub roots and by means of filters. However, as opposed to requirements, tests and test instances may also be included in a scope item by means of linkage. One may indicate on a scope item that it includes the tests covering the scope item’s requirements. Similarly, one may indicate that a scope item includes the test instances included in the scope item’s tests.



It is recommended to manage test and test instance linkage in the ALM implementation and to utilize this when setting scope item content.

Test planning should always reflect the functionality not the release. Test Level can be used to distinguish between UAT and System tests but do not use the release to arrange folders in this area. Releases, as we have discussed are managed elsewhere. The testing tree should as far as possible reflect the requirements tree the test execution sets should include the test sets that are linked to releases and test cycles.

### Tests and test instances in the example project

In the example project I have created the tests in line with the requirements tree using the automatic conversion dialog. This is an important function because it automatically creates the test coverage and linkage between the requirement and the test. If linkage is not created automatically it can be added with the “Test Coverage” function on the requirements detail page.



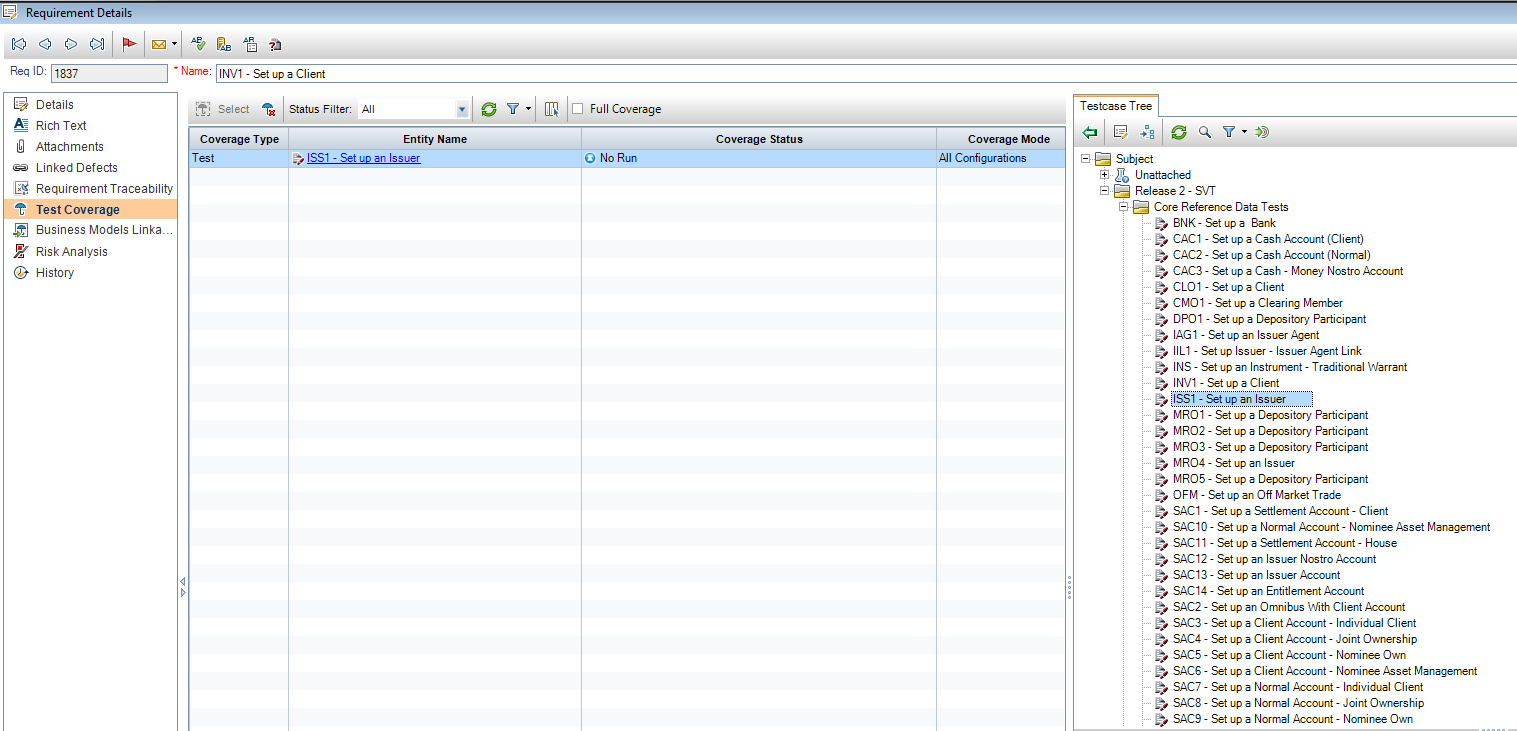


Figure 8: Manual Creation of Requirement coverage

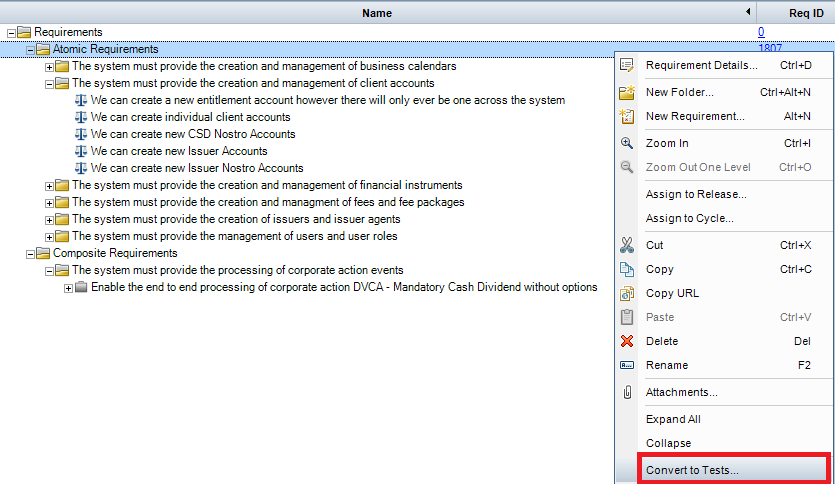


Figure 9: Converting requirements to tests

This will show the following dialog. Here I have selected the option “Convert lowest child requirements to tests”. This is generally preferable unless the requirements are expressed at a very, very low level in which case conversion to steps may be used.

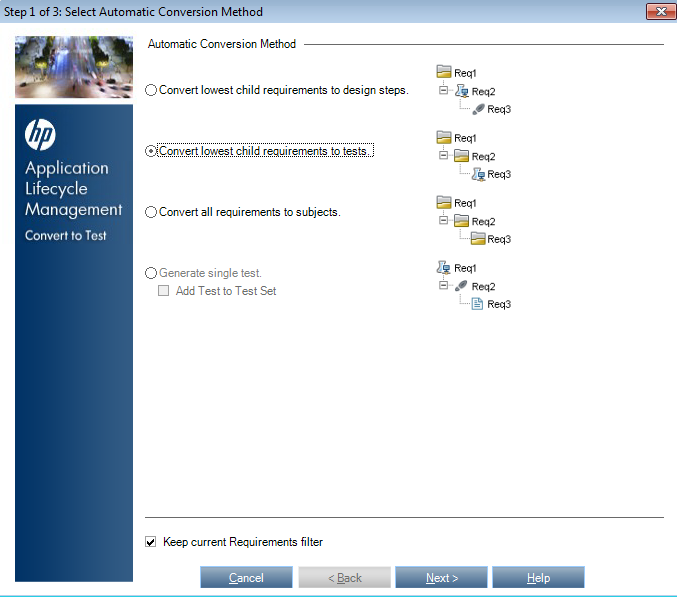


Figure 10: The "Convert to Test" dialog

Selecting “next” will confirm the set of requirements that are to be converted and the final screen shows where in the test planning tree they are to be created. At this point the test planning tree in the example project is empty and these tests will be created at the top level.

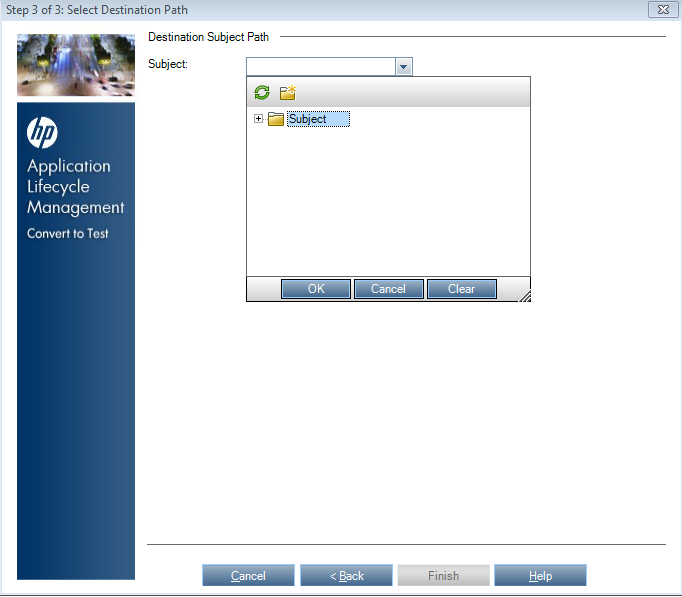


Figure 11: Select where to create the tests

Clicking “ok” & “finish” will create the tests and links from requirement to test.

*Please note that if you create the lowest level as a folder not a test then the linkage between test and requirement will not be created. If you then create individual tests in those folders you will have to add the linkage manually.*

### Structuring the test plan to work with Scope Items and KPI’s



To be added to a scope item a test must be either linked to a requirement, selected in the scope definition or within a test set.

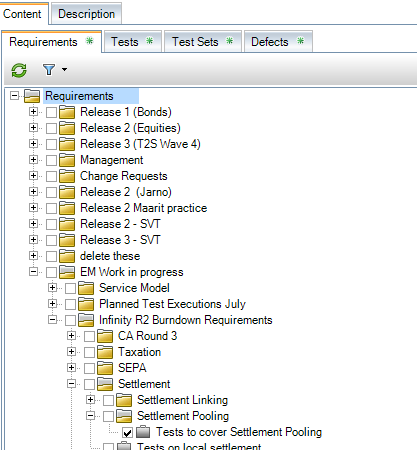


Figure 12: Scope item content definition

In Figure 13 we have a requirement called “Tests to cover settlement pooling” and if we look at the test coverage we can see that four tests are linked to this requirement, note, that in this example additional tests have been created to cover this requirement. This is a more complex image that you will get if you have one to one coverage as recommended by the “loading requirements” section above.

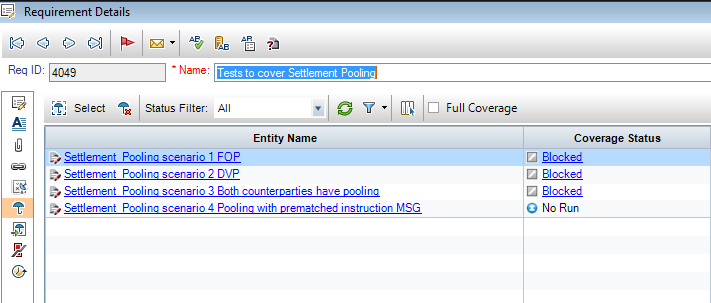


Figure 13: tests linked to requirement

If we then look at the scope item content and select “tests” then we have two options. Either to select “tests covering selected requirements” or “tests in selected folders”

### Structuring the test execution plan to work with scope items and KPIs

When building a release scope item we can also select test sets as valid scope content. Here in the scope content selection dialog we again have two choices as we did in figure 15 above. Again this is complex and confusing but has far reaching implications.

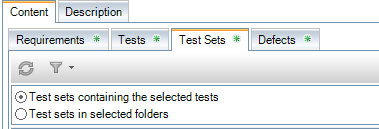


Figure 18: Test Sets as Scope Items

*If we select “test sets containing the selected tests” we must have some test folders selected. All tests sets that contain the specified tests, regardless of where they are in the execution plan, will be counted.*

If, alternatively we select “test sets in specified folders” then we have a new problem.

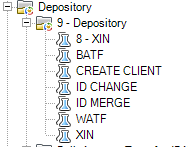


Figure 21: The new problem

Because we do not have the ability to select actual, specific test sets, only the folders in which they reside, if we have a folder like this one which contains test sets relating to multiple different areas, if we select that folder all those tests will be counted, so when you report on “Whole account transfer” you will also get client creation and ID Merges bundled into the numbers.

This is why it is essential to structure the test execution model in a way that can take advantage of KPI calculation and metric reporting.

In the test plan module the tests should be arranged purely by function

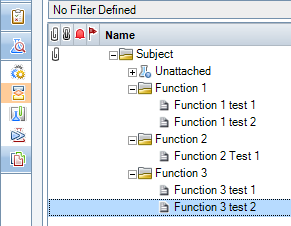


Figure 22: Test arrangement by function at planning level

Then in the execution module arrange them by release and then function

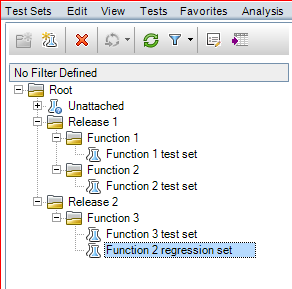


Figure 23: Arrange by release and then function at execution level

This means that the functions can be reported on effectively.

## Defects

### Recording Defects

A defect should be:

* Reproducible every time with a specific set of data
* Linked to a single test step
* Attributable to a specific requirement
* Linked to a specific scope item
* Clearly and accurately described to give the developer the best chance of fixing it on the first time of asking.

### Including defects in release scope

Defects are included in a scope item by means of filtering. It is recommended to use the Target Release and Target Cycle fields in your filter settings, to include defects in a scope item. Include defects in a scope item using the Target Release and Target Cycle fields in your filter settings.

Note that scope item content is usually dynamic. For example, entities may be added to scope items because they comply with a specified filter or because they are a descendant of a selected root.

There is a separate defect workflow document covering the defect workflow in more detail.

# The Management Module



What does all of the above get us? Beyond the academic benefit of knowing we are using the tool properly what material benefits will we gain?

The answer is that we will gain significantly more stable, more automated, more credible reporting that is available to everyone on demand, all the time. It will always be up to date, it will always be true.

By combining, from the project management side a view of the scope, the schedule and the key performance indicators, and from the testing side a view of tests, defects, executions and statuses we can build a comprehensive view of the program that should meet all reporting requirements.

The management module brings everything we have discussed above into one view. The best place to start to examine what this view can provide is on the master plan. This “master plan” is a detailed view of the “testing” component on a higher level plan that would include infrastructure setup, training etc.

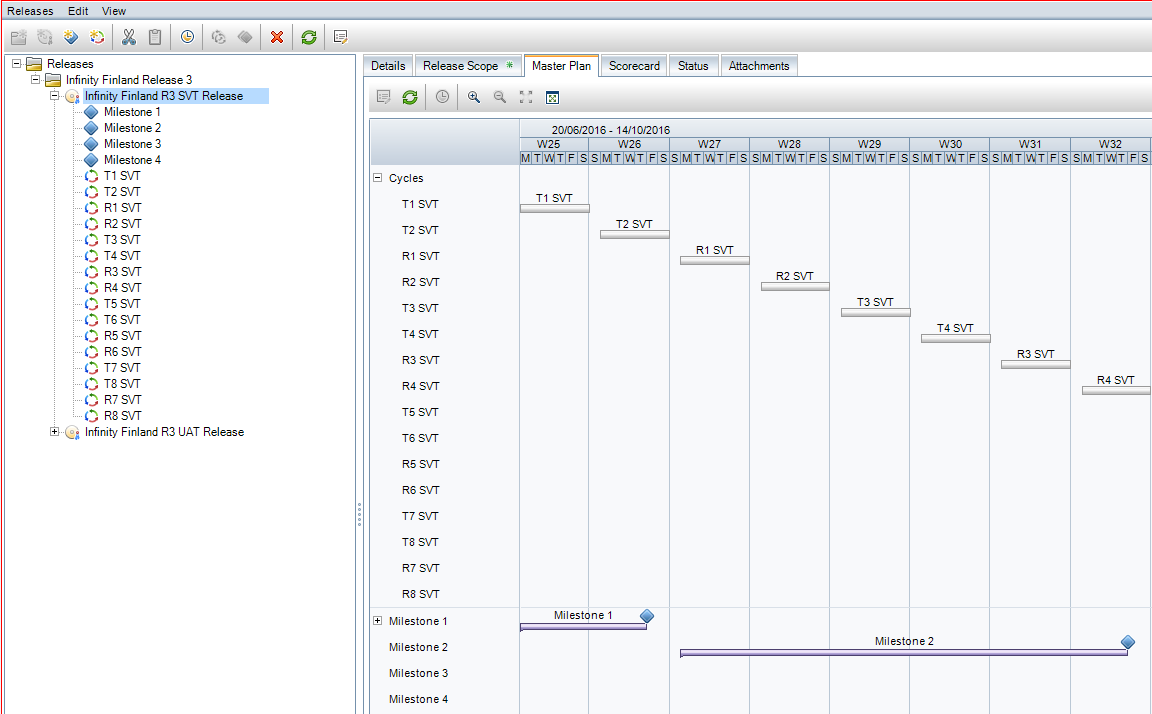


Figure 24: Release, Milestones and Cycles

This shows a Gantt-type chart with the individual test cycles, defined here as weekly cycles, over time, with the milestones.

The milestones are configured to include the scope items from the overall release. Here Milestone 1 is covering the scope item “Core reference data” and the KPI’s we want to measure are the authored tests, the automated tests, the passed requirements, passed tests and rejected defects. Available KPI’s are at the right and can be added; new KPIs can be created based on just about any reporting requirement.

## Key Performance Indicators and Thresholds

The RAG calculation of the KPI can be configured to change over time. It would not be right to report a RAG status of red on day 1 because we have not completed testing however as we move closer to the release date the thresholds should get tighter each week. The aim of the test manager is to keep the track of the KPI progress within the “green zone”.

Different KPIs require different thresholds to make sense. For example here we have a threshold profile for blocked test instances. This follows a downwards trend as we require the number of blocked instances to approach zero as the project progresses.

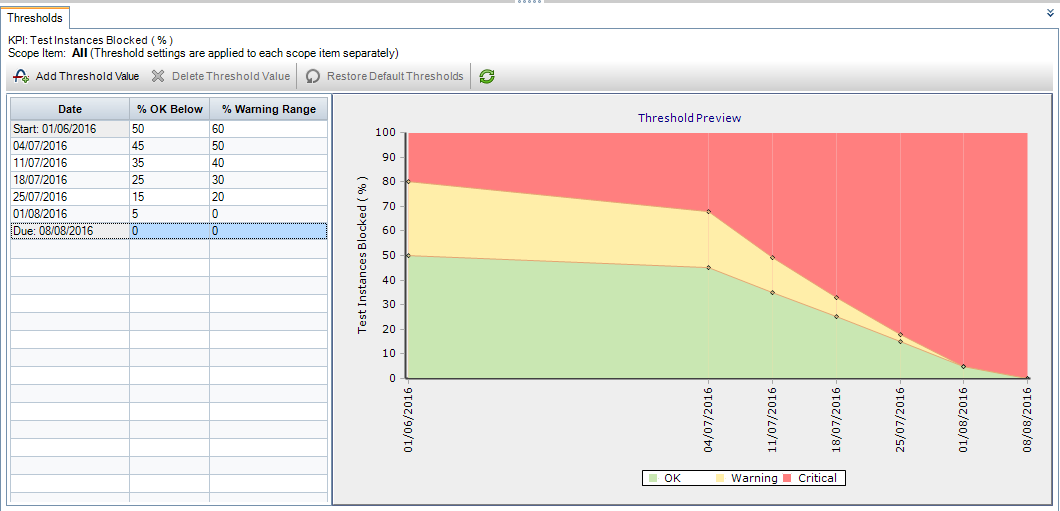


Figure 25: KPI thresholds on a downward trend

In this next figure we see the profile for a KPI like “failed tests” where we expect the number of failed tests to rise at the beginning of the test cycle before dropping sharply back down as defect fixes are delivered. Again the job of the test manager is to keep the metrics within the “green zone”

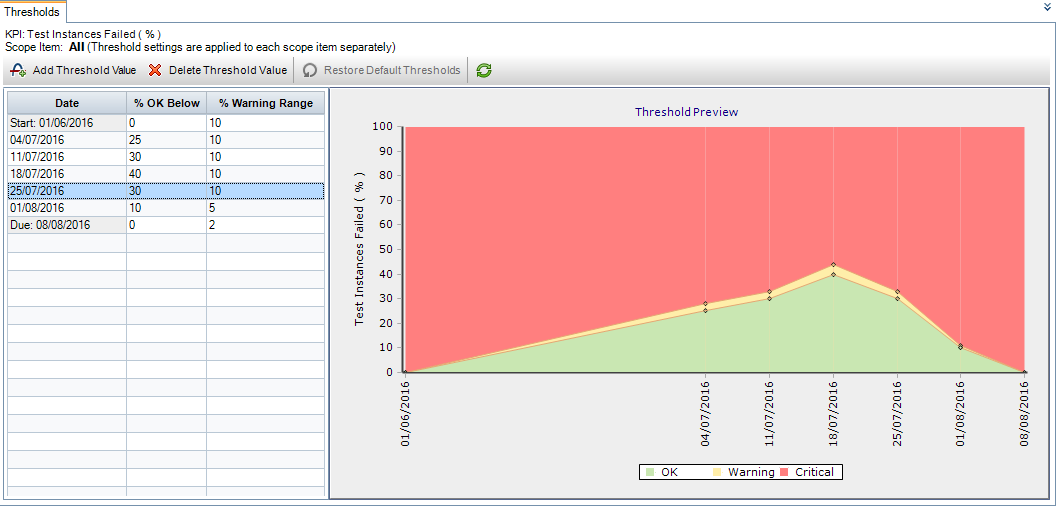


Figure 26: Thresholds in a bell like curve

## The release Score Card

The release scorecard brings together all the milestones and all the KPI’s on one highly configurable view. In this example one item has been added. As scope items are created and milestones are configured this grows into a single, consistent view of the program that dynamically reflects the priorities at the time. This object, when correctly supported by well-designed KPI’s and RAG profiles should be the principal source of reporting truth at any time in the program. This scorecard presents some of the default Key Performance Indicators that ship with ALM.

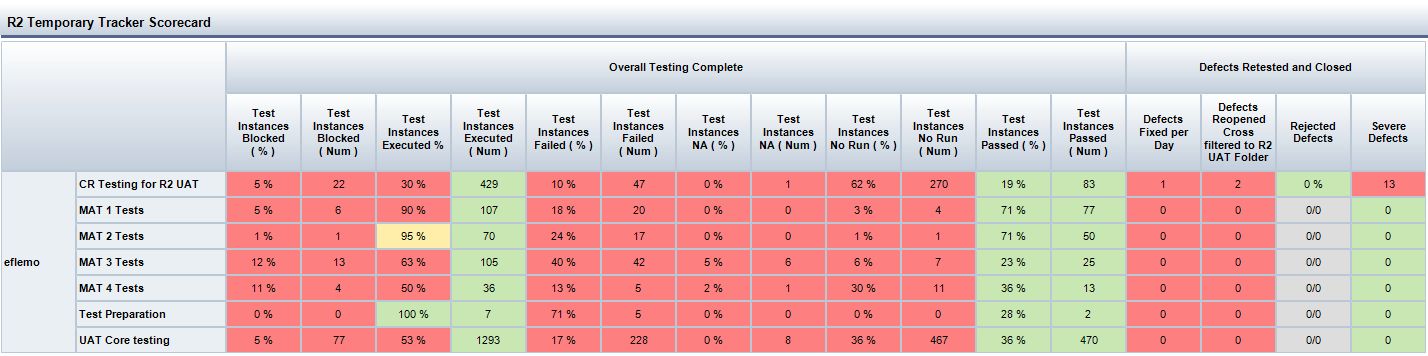


Figure 27: The release score card

# Libraries and Baselines

## Libraries

A library in ALM can contain a set of requirements and their associated tests. When a library is created you set a *baseline* that freezes the tests at that particular state in time.

### Example

Requirements and tests are captured in a library and base lined at version 1. Client now makes some changes to the tests to accommodate the Finnish market. Client Sweden can, at a later date, import the baseline version of the tests without taking the changes made specifically for Finland.

The two baselines can subsequently be compared to see the specific differences in tests and requirements present or absent from each version of the library.

# Implementation Details

## Workflow

### Step 1: Determine Release Schedule

The first step is to determine whether to have one release or multiple releases in the release module. It may be desirable over a 12 month program to have multiple releases, maybe quarterly with the full content delivered over four releases. Alternatively a single release with multiple milestones tracked independently could also work.

### Step 2: Identify and create Scope Items

The second step is the creation of scope items according to the guidelines on “what makes a good scope item”. To summarize it should be large enough to be a component or small system in its own right that will be described by a number of requirements that make up that item.

### Step 3: Build and Load Requirements

Step 3 as you may have guessed is to work out the requirements that describe each of the scope items and structure these in such a way that they will easily convert to test cases or test subject areas and test sets

### Step 4: Create Tests and Test Sets

Using the requirements and the “convert to tests” function we create a structure for the test cases we are going to use to test the system. We can define libraries of tests and import them from other projects. Libraries can be baselined at a set point in time

### Step 5: Link tests, requirements and defects to scope items

As I have mentioned it is hugely beneficial to link tests and requirements as this enables you to use the “tests that cover this requirement” selection in the scope item content pane.

### Step 6: Determine and build KPI’s and KPI

ALM calculates KPIs once daily at 00:00. These KPI calculations are based on whatever we want to track be that test preparation, defect counts, changes in defect status etc. Just about anything can be configured as a KPI but To meet the requirement of keeping the reporting consistent and stable we determine the KPIs at the start of the project. We can also configure the way in which the RAG status of these KPIs will change over the course of time. It is not appropriate to report a RAG status of red on the morning of day 1 because the tests are not 100% complete. See above on “Key Performance Indicator Creation”