Relationships and Traceability in PTC Integrity Lifecycle Manager

Author: **Scott Milton**

Table of Contents

1. Abstract

This document details the various mechanisms that provide "traceability" in Lifecycle Manager. For the purposes of this document, traceability is defined as the ability to link disparate items/objects/artifacts together to provide increased visibility and easier location of the related items. The exact types of objects linked, as well as the reasons for doing so differ on a case by case basis. The functionality detailed includes both the various types of relationship fields as well as the special functionality described as traces. The document is roughly divided into two sections; definitions and recommendations on how to configure the various relationship mechanisms, followed by how to view or use the various mechanisms.

2. Introduction

One of the primary uses of Lifecycle Manager Lifecycle Manager is to provide increased visibility of both how processes are defined and the various artifacts representing the current state of a process, throughout the organization.

As an example, Lifecycle Manager Lifecycle Manager is often configured to represent a Project lifecycle. This might involve a particular workflow for the Project, but also the decomposition of the Project into lower level Items, such as subdividing the Project into individual Features and then into Development Tasks. These Tasks are often linked to a Requirements Document, which itself contains links to individual Requirement items. Those Requirements are then linked to the source code/file artifacts that are the deliverable for a completed Task and the Test Cases that ensure sufficient test coverage of the requirement. The Test Cases in turn may need to be linked to any Defects found as a result of executing the Tests. This example could certainly be expanded to encompass other areas of the process, but it is clear that as the number of "objects" or "items" in an overall process continue to grow, they will clearly have multiple connections.

In a generic sense, these connections or links between items are often referred to as traceability. This traceability better leverages the assets across the organization; linking items provides easier access to information that is often important for many reasons, ensuring compliance, quality, completeness and accountability to name a few. Easy navigation between various artifacts that are traditionally in distinct domains, such as customer input, requirements, development, testing, etc., helps ensure a seamless stream of information between groups. It enables rooting out individual components or areas of the process that consistently cause problems. Proper traceability makes it much easier to enforce business logic, such as confirming that all requirements have functional specifications and test cases so that they are developed and tested before the product is released or project milestones are reached. Linkages of items also make it much easier to accurately gather and "roll up" metrics from the various artifacts and sub-processes involved in a project to a common point for improved accuracy, easier visibility and greater potential for comparison to other projects.

Lifecycle Manager provides a number of different traceability mechanisms to link these various artifacts and sub -processes together. At a macro level, these objects are typically broken down into "Workflows and Document" Items and "Configuration Management" artifacts (e.g., Source Code). Lifecycle Manager provides a clear and visible connection between development objects such as source code or documentation and their associated requirements. In general, the term "trace" is restricted to particular type of linkage in Lifecycle Manager, which will be defined below in the section entitled Relationship Fields versus Trace Relationship Fields.

Note: The rest of this document assumes you have at least a passing familiarity with Lifecycle Manager. For more information about Lifecycle Manager, please visit here. To schedule additional training for your location, please visit here.

3. Workflows and Documents Relationship Fields

Workflows and Document items are composed of a workflow, also known as a lifecycle or process, and fields of data. The workflow represents the state of the item and the fields contain both user defined data and system provided metadata about the item. The primary mechanism for linking items to each other are relationship fields, of which there are three distinct types:

A brief update on naming. The product formerly known as "PTC Integrity" is now named "PTC Integrity Lifecycle Manager", since PTC Integrity now refers to a family of software and systems engineering products. For brevity and clarity, this document uses "Lifecycle Manager" as an abbreviation for the full name, "PTC Integrity Lifecycle Manager".

- Standard Relationship Fields
- IBPLs
- QBRs

In an entity-relationship diagram (e.g., architecture diagram) that details an overall process these three types of fields would be used to represent the arrows connecting items together. Each has its own particular strengths, although there are some overlaps.

Note: Two other types of fields relevant to Tracing to Configuration Management artifacts, SI Project and Source Trace fields, will be defined later in a later section of this document, Tracing to Source Code.

- Editing an item and clicking on the "add related item" button (or the corresponding right click/context menu option). On the resulting dialog, entering the ID of the remote item(s) or running a query and selecting one or more items from the query results.
- Editing an item and clicking on the create related item button (or the corresponding context menu option). From the resulting dialog, choose from amongst the allowed types for the relationship field and populating the form, including any mandatory fields, for the selected item type.
- Dragging and Dropping one or more items on top of another item.
- A Project item relating to Sprints.
- A Sprint relating to User Stories.
- A Defect relating to duplicate Defects (or Change/Feature Requests).

In these examples, it is desirable for end users to show information about the related items, including the current state, owner and creator. In some cases, such as with duplication, it may require advanced querying for the relationship to be set and there may potentially be a large number of potential choices.

- To tightly restrict the specific items that can be related based on the data in either the remote, or current item. Unlike IBPLs, standard relationship fields typically only restrict by the types of items, not by any other fields of data on the items.
- To make it as simple as possible to relate existing items together (e.g., for users who would prefer not to have to run a query or type an ID).
- As a form of administrative partitioning. In this case of the effort needed to maintain the entries for a picklist can be delegated by an administrator by setting the permissions to create and retire items that back the IBPL as needed to various groups.
- A Project item relating to a Product.
- A Product relating to a Product Line.
- A Requirement relating to a Function of a Product.
- A Product Line showing all Active Products.
- A Product showing all Active Projects affecting it.
- A Product showing all Change Requests planned for it within the next 30 days.

In these examples, it is desirable for end users to show information about the related items, including the current state, owner and creator. It is also necessary for the related items to be automatically included or excluded from the relationship based upon certain rules, rather than manual intervention by end users (e.g., when items reach a certain state in their lifecycle or within a certain timeframe).

- To allow structured metadata to be defined for the Project or Subproject beyond the descriptions allowed directly in the Configuration Management viewsets.
- To define the location of associated configuration management artifacts, especially for an automated process. For example, an item representing a Build Request could allow users to select a specific checkpoint of source files to be built.
- Publish Metrics about the CM project?

When users are working in the Configuration Management viewsets, they can select any Configuration Management Project and view any items that have selected it in an SI project field visible on the Item; these are called associated items. Typically this will be accessed via the following menu: Project > View Information > Associated Items tab. From the Associated Items list you can easily browse or edit the items via the right click context menu.

- Viewing item details, including relationships: im viewissue --showrelationships<item id>
- Viewing item details, including change packages: im viewissue --showchangepackages<item id>
- Launching the graphical relationships view: im relationships -g <item id>
- Choosing the first row and first column could be read as, Input items have a peer trace called "Is Related To" to other peer items.
- Choosing the first row and third column could be read as, Input items have a downstream trace called "Decomposes To" to Requirement items.

A downstream trace will have a corresponding backwards or upstream trace e.g., Decomposes To/Decomposed From, Satisfied By/Satisfies, Validated By/Validates, Verified By/Verifies, etc.

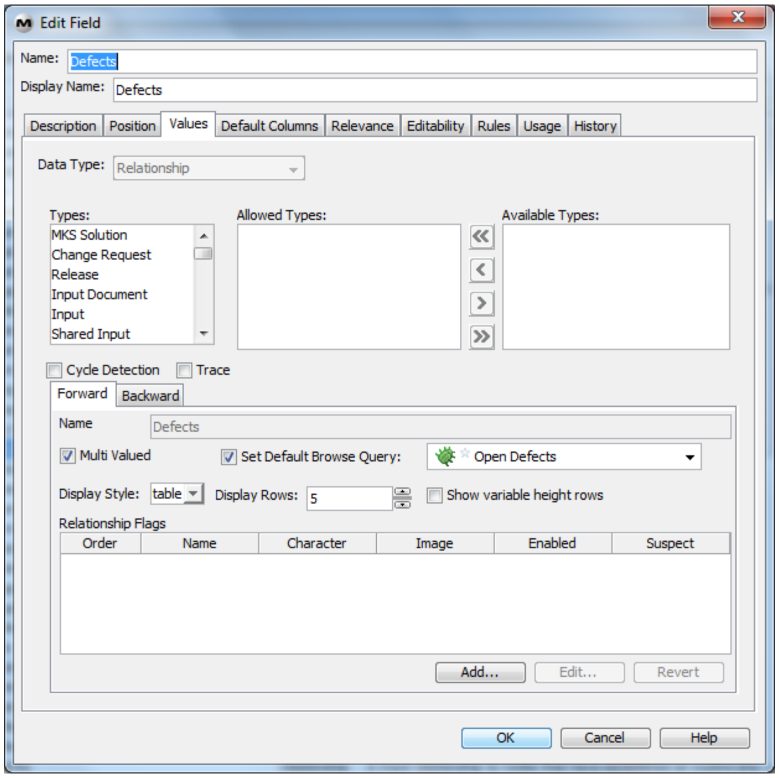
1. Open the administration client and go to "Workflows and Documents"
2. Select MKS Solution and Edit Type
3. Create a trace property with the following characteristics:
   a. Name: MKS.RQ.trace.<typename>
   b. Value: <Relationship field/trace to be created>:<typename for other end of trace>

- In the Admin Client, go to Workflows and Documents>Fields>Create Field (or edit field if you have field created, but have not named it as a Trace field)
  - In the "Suspect" column, marking this as true means core product will handle setting suspect.
  - If you mark "false" for suspect, this means you are either not setting suspect, or you have a trigger that will do the work for setting suspect.

3.1 Standard Relationship Field

The standard relationship field is created by an administrator in the admin client. It has both a forward and backwards component and either direction is made visible on one or more types of items. It is used to provide easy visibility and traversal of links between items. This type of field is typically rendered as a table where the possible related items that can be selected are other items in the database. The functionality provided by this field allows an administrator to fine tune which type of items can be linked together. The administrator also defines the default columns rendered in the table for all users, although end users can override this column as desired in their own UIs (both web and java client). Either direction of this field can be set to be single or multi valued by the administrator, allowing for one to one, one to many and many to many relationships. Although the administrator is defining the types that can be on either end of a relationship, end users typically set the actual values of this field manually, although triggers can be used to automate the process. End users will typically set the values of this relationship field in one of three ways:

This is the most common relationship field, and if there is no specific need for the capabilities of the other two types it should be used.



Name: Defects
Display Name: Defects

Data Type: Relationship

Types: MKS Solution, Change Request, Release, Input Document, Input, Shared Input
Allowed Types:
Available Types:

☐ Cycle Detection ☐ Trace

Forward | Backward

Name: Defects
☑ Multi Valued ☑ Set Default Browse Query: Open Defects
Display Style: table Display Rows: 5 ☐ Show variable height rows

Relationship Flags

Order	Name	Character	Image	Enabled	Suspect

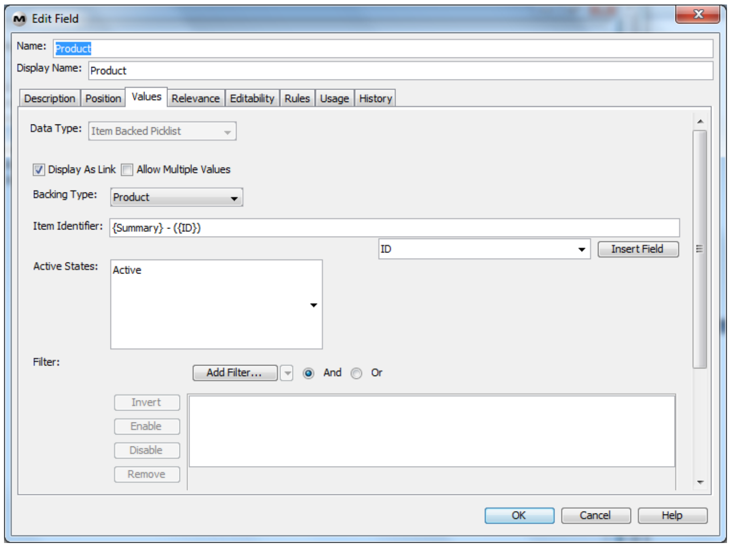
Fig A. Creating a Standard Relationship Field

Examples might include:

3.2 IBPL Field

The IBPL relationship field is created by an administrator in the admin client and made visible on one or more types of items. IBPL stands for Item Backed Picklist, and this type of field is rendered as a picklist where the possible values that can be selected are other items in the database. There is only a forwards direction to this type of field; the items selected do not themselves show any reference to the item(s) that selected them, unless a QBR is setup, which will be described in the next section. The functionality provided by the IBPL field allows an administrator to fine tune which possible items show up in the list by selecting their possible types and states (e.g., show all Active Releases), but can also be fine-tuned through the use of various rules (e.g., show all Active Releases that have the same value of the Product field that I have). This field can be set to be single or multi valued by an administrator. Although the administrator is defining the potential choices, end users typically set the value of this field manually, although triggers can be used to automate the process. The exact text displayed in the IBPL for each item, and whether or not it is rendered as a hyperlink or not is also under the control of the Administrator.

The IBPL relationship field is often used for three main reasons:



Name: Product
Display Name: Product
Data Type: Item Backed Picklist
[✓] Display As Link [ ] Allow Multiple Values
Backing Type: Product
Item Identifier: {Summary} - ({ID})
ID
Active States: Active
Filter: [✓ And] Or

Fig B. Creating an IBPL Field

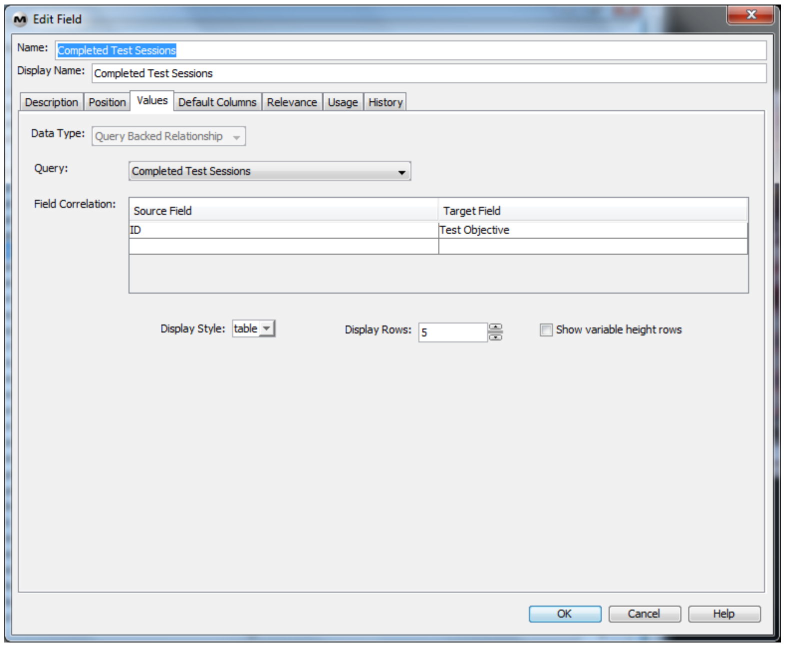
Examples might include:

In these examples, it is not particularly important to show a great deal of information about the related items. The relationships will generally not require any advanced querying to be set as there will be a small number of potential choices; it is more important to allow for quick and easy setting of the values.

3.3 QBR Field

The QBR field is created by an administrator in the admin client and made visible on one or more types of items. QBR stands for Query Backed Relationship, and this field superficially resembles a standard relationship field. However, this type of field has only a forwards component and the values of the field are not set by end users, instead they are derived by running a query that an administrator has defined. Like a standard relationship field, it is used to provide easy visibility and traversal of links between items. This type of field is typically rendered as a table where the related items are other items in the database that satisfy the defined query. The functionality provided by this field allows an administrator to fine tune a standard query with field correlations. Field correlations allow an administrator to add additional "virtual" filters to a standard query (e.g., show all items returned by the Active Defects Query) to further restrict the results to those that have field in common with the item that houses the QBR (e.g., show all items returned by the Active Defects Query that have the same value of the Product field that I have) The administrator also defines the default columns rendered in the table for all users, although end users can override this column as desired in their own UIs (both web and java client). This field is always considered multi value, although depending upon the query, in some cases one or no items might be displayed. This field type is never editable by end users.

The QBR field is generally only used when the system should be computing what is related at a current point of time. It often used in conjunction with an IBPL field. As an example, you may have types of items representing Products and Defects. The Defect may have an IBPL field allowing the choice of a single Product. The Product Item could have a QBR field showing all Defects that have selected that Product Item in their IBPL fields. It could also have multiple QBR fields that further break down the Defects that have selected the Product, for example showing all High Priority Defects, or All Defects opened in the last 30 days. A standard relationship field links items together and they stay linked, potentially forever, unless someone unlinks them. QBRs are computed dynamically each time an item is viewed, so the values of the field can change drastically between viewings depending upon the query the QBR is based on.



Name: Completed Test Sessions

Display Name: Completed Test Sessions

Data Type: Query Backed Relationship

Query: Completed Test Sessions

Field Correlation:

Source Field	Target Field
ID	Test Objective

Display Style: table

Display Rows: 5

☐ Show variable height rows

Fig C. Creating a QBR Field

Summary Table of Relationship Field options:

Field Type	Description	Best For	Not Recommended For
Relationship Field	Bidirectional relationship	User controlled relationships, drag and drop scenarios	Situations where the potential choices are highly restricted, or when the items which are related change based on a query
IBPL	Unidirectional Picklist of Items	Quickly selecting from a restricted list of choices	Situations where there are a large number of potential choices, when drag and drop to change relationship is desired
QBR	Unidirectional list of items retrieved from a query	The system running a query to find items that satisfy particular criteria, displaying items that have chosen the current one in via an IBPL	Situations where a user needs to control directly which items are related

4. The Document Model

Standard Relationship Fields can under some circumstances be a specialized type of relationship called a Trace Relationship. In order to understand Trace Relationships, it is necessary to have a cursory understanding of a special feature of Lifecycle Manager called the Document Model.

The Document Model allows a collection of related Lifecycle Manager items to be rendered in a special view that integrates the disparate items into a cohesive document, similar in appearance to a Microsoft Word document. The Document view is the interface you use to view and modify the parent Document Item as well as he individual items (e.g., Requirements, Test Cases, etc.) that make up their content in a cohesive, hierarchical, tree structure.

Documents are classified by their domain, typically Input, Requirements, Design and Test. Although each of the content Items in a document (e.g., individual Requirements, Test Cases, etc) can have many relationship fields of the three types defined above, specialized functionality exists for creating relationships between content items, both those in the same domain and in other domains, called Trace relationships. Like standard relationship fields, there is inherent directionality involved with traces. Generally, a trace to an item that is in an earlier domain, that is to say conceptually earlier in the overall process, is called an upstream trace (e.g., from a design to a requirement), and a trace to an item in a later domain is a downstream trace (e.g., from a requirement to a test case). Intra domain traces are generally called peer traces. Although the number of individual trace relationship types is not constrained, generally there is one relationship between each type of object in the system, for example: high level requirements, marketing requirements, test cases, components.

4.1 Relationship Fields versus Trace Relationship Fields

Trace relationships are a special subset of standard relation fields with additional functionality and defaults defined by the administrator. In most document based viewsets, there are menu items for creating, propagating and viewing traces that are preconfigured specifically for traces (e.g., menu items to view upstream traces, view downstream traces, view peer traces).

Depending upon which view the user is in and how the viewset has been configured, users can create traces (e.g. Requirements to Test Cases) in a variety of ways. For example, much like a standard relationship field by doing the "Add related" gesture from the item detail view, or the "Add related" button in the edit view. However, most viewsets specifically designed for the document model also include specialized gestures just for traces, such as the Relationships > Create Trace menu item, the Create Trace right click context menu item and allowing users to perform an Alt+drag and drop gesture from within the Document View, if the admin has configured the solution to do so. This can be a very convenient way to set up many traces quickly. Please refer to Appendix 3. Creating solution properties to allow drag and drop trace creation for details on how to configure this.

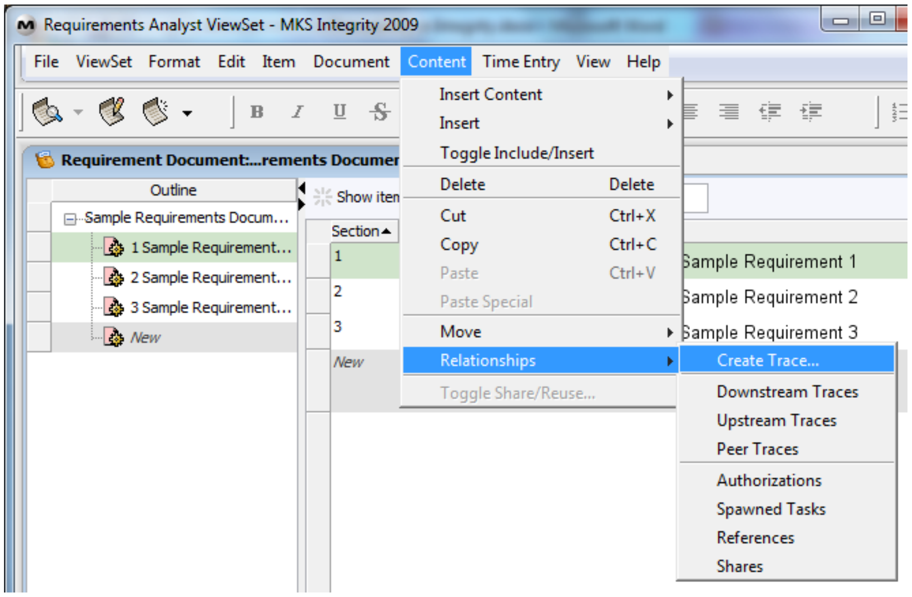


Fig D. Launching Create Trace from the menu

Note: You can also see how to launch the Downstream, Upstream and Peer Trace views above.

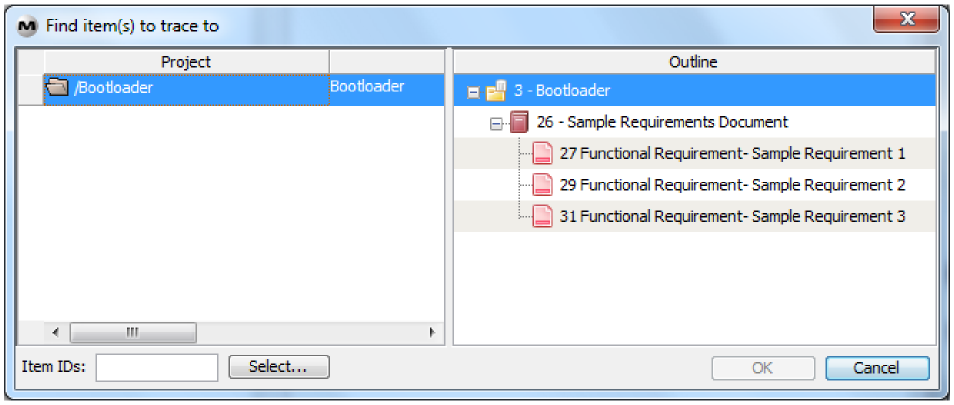
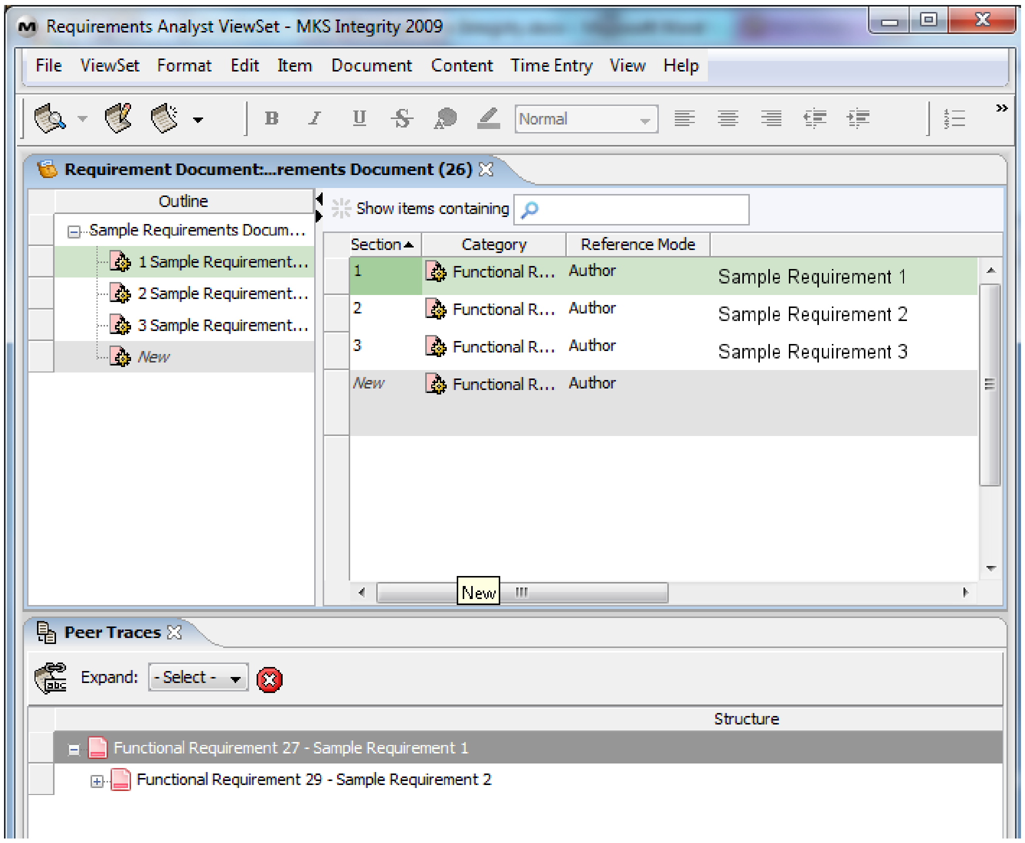

Find item(s) to trace to
Project
/Bootloader
Bootloader
Outline
3 - Bootloader
26 - Sample Requirements Document
27 Functional Requirement- Sample Requirement 1
29 Functional Requirement- Sample Requirement 2
31 Functional Requirement- Sample Requirement 3
Item IDs:
Select...
OK
Cancel

Fig E. Creating a Trace from the menu

Depending on how your administrator has configured your ViewSet, you can view trace relationships in between content items in a variety of ways; for example, upstream, downstream, or peer traces. These views are essentially a Hierarchical Relationship View automatically configured to filter appropriately based on the user's selection and context.



Sample Requirements Docum...
- 1 Sample Requirement...
- 2 Sample Requirement...
- 3 Sample Requirement...
- New

Section	Category	Reference Mode	
1	Functional R...	Author	Sample Requirement 1
2	Functional R...	Author	Sample Requirement 2
3	Functional R...	Author	Sample Requirement 3
New	Functional R...	Author	

- Functional Requirement 27 - Sample Requirement 1
  - Functional Requirement 29 - Sample Requirement 2

Fig F. Peer Trace view

4.2 Trace Propagation

To propagate traces means to copy trace relationships from one item to another. An example of this might be the situation where there is a Requirements Document (RD1) that has various traces to a Test Suite (TS1). When a new Release is being planned, both the Requirements Document and Test Suite are branched, creating a new Requirements Document (RD2) and Test Suite (TS2) respectively. The trace propagation wizard guides the user in copying the original set of traces in the original pair of documents into the newly created pair of branched documents.

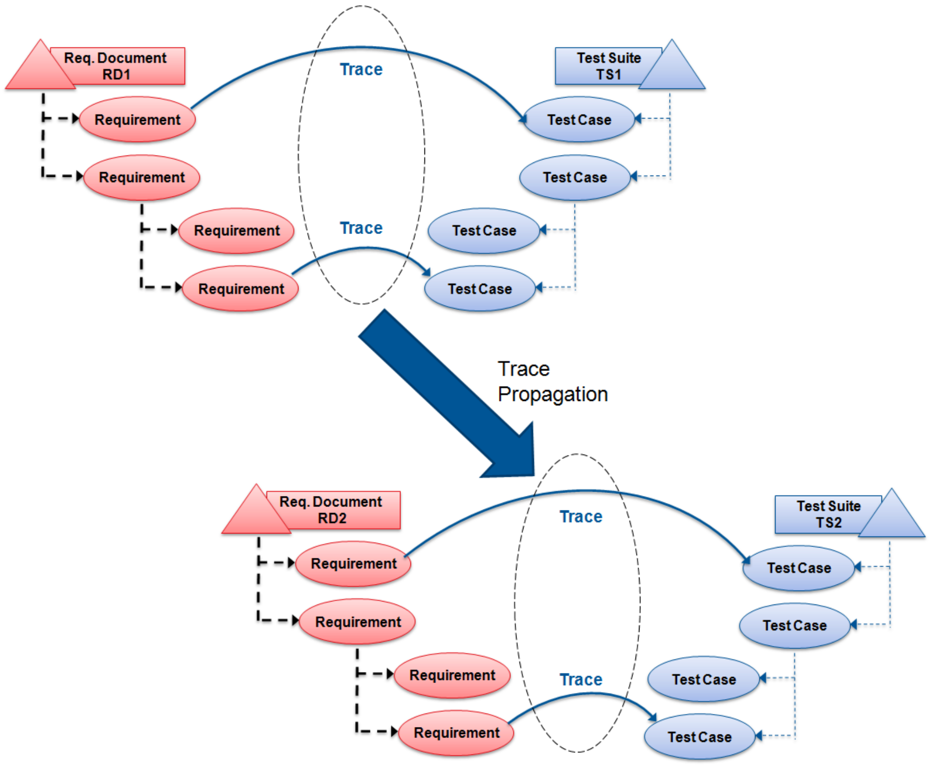
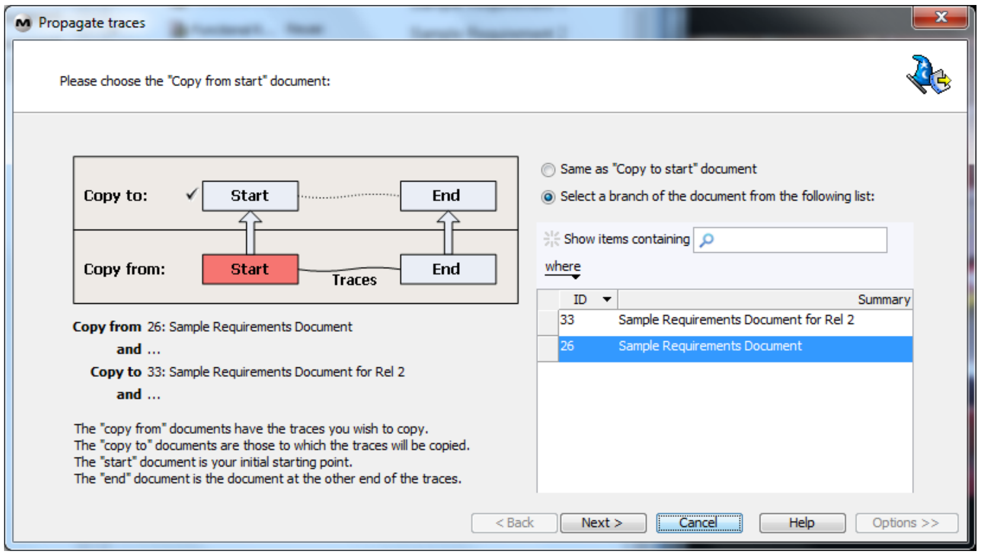


Fig G. Trace Propagation between two pairs of documents



Copy from 26: Sample Requirements Document
and ...
Copy to 33: Sample Requirements Document for Rel 2
and ...

The "copy from" documents have the traces you wish to copy.
The "copy to" documents are those to which the traces will be copied.
The "start" document is your initial starting point.
The "end" document is the document at the other end of the traces.

( ) Same as "Copy to start" document
(●) Select a branch of the document from the following list:

Show items containing
where

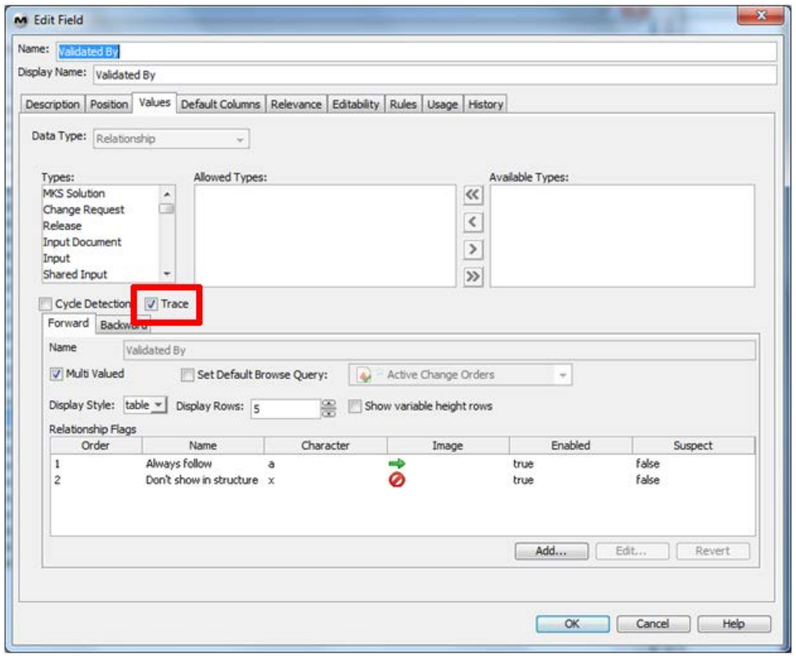
ID	Summary
33	Sample Requirements Document for Rel 2
26	Sample Requirements Document

Fig H. Trace Propagation Wizard

Note: The fields that are copied when you branch a document are determined by your administrator. PTC recommends that trace relationship fields are not automatically copied during branching. This enables you to control which trace relationships get copied to the branched document using the propagate traces wizard.

4.3 Configuring Additional Traces

For an administrator to configure additional traces, please refer to Appendix 3. Creating a Trace Relationship field and Appendix 3. Creating solution properties to allow drag and drop trace creation.



Edit Field

Name: Validated By

Display Name: Validated By

Description | Position | Values | Default Columns | Relevance | Editability | Rules | Usage | History

Data Type: Relationship

Types:
- MKS Solution
- Change Request
- Release
- Input Document
- Input
- Shared Input

Allowed Types:

Available Types:

☐ Cycle Detection ☑ Trace

Forward | Backward

Name: Validated By

☑ Multi Valued ☐ Set Default Browse Query: Active Change Orders

Display Style: table Display Rows: 5 ☐ Show variable height rows

Relationship Flags

Order	Name	Character	Image	Enabled	Suspect
1	Always follow	a	→	true	false
2	Don't show in structure	x	⊘	true	false

Add... Edit... Revert

OK Cancel Help

Fig I. Creating a Trace Relationship Field

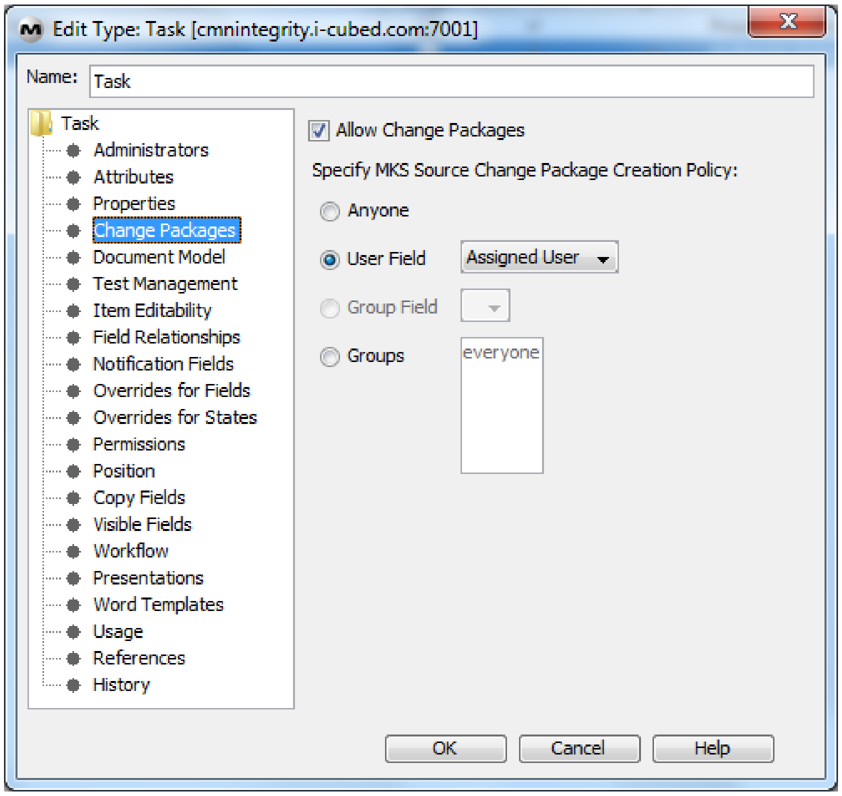
5. Tracing to Source Code

5.1 Change Packages

The primary method for tracing Workflow and Document Items to Configuration Management Artifacts is called a Change Package, often referred to as a cp. The Change Package is essentially one or more "containers" that are opened on an Item. As a user makes changes to the Configuration Management Repository, they are prompted to select an open cp to add these changes to. As they do so, a record of all the changes (adds, drops, updates, etc.) being made for the task are captured and stored in the package.

Administrators define which types of items are allowed to have Change Packages as well as who can add operations to them. They also define which states in the item's workflow allow an Open Change Package to exist. Typically this is used to enforce that repository changes are not made before approvals are completed, and that development is complete before testing or reviews begin. These features, along with other repository permissions, effectively allow Administrators to define who can make changes to the code repository and when (e.g., you must have a Task Item assigned to you in the Approved to make code changes).

Note: There are a number of other reasons for using Change Packages, such as enforcing Code Reviews and applying changes to other paths of development, but these are beyond the scope of this document. Please refer to the Lifecycle Manager Administration Guide for further details.


Edit Type: Task [cmnintegrity.i-cubed.com:7001]

Name: Task

[✓] Allow Change Packages

Specify MKS Source Change Package Creation Policy:

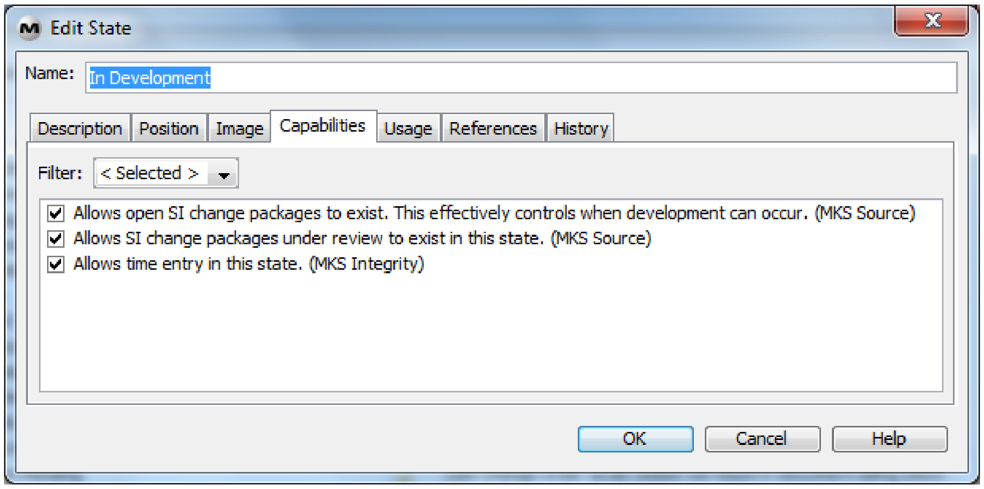
Anyone

[✓ User Field] Assigned User

Group Field

Groups: everyone

Fig J. Configuring Change Package Options on a Type


Edit State

Name: In Development

Filter: < Selected >

[✓] Allows open SI change packages to exist. This effectively controls when development can occur. (MKS Source)
[✓] Allows SI change packages under review to exist in this state. (MKS Source)
[✓] Allows time entry in this state. (MKS Integrity)

Fig K. Configuring Change Package Capabilities on a State

5.2 Source Trace Fields

The Source trace field is created by an administrator in the admin client and made visible on one or more types of items. Multiple Source Trace fields can be created in an implementation, for example, an administrator could create an "Implementing Source" trace field and a "Verifies Source" trace field and make them visible on the Requirement and Test Case item types respectively. These fields allow end users to link artifacts (i.e., members) in a Lifecycle Manager Configuration Management repository (i.e., source code control) to the item. Source Traces can be to any Lifecycle Manager item type, but are primarily useful for Requirements.

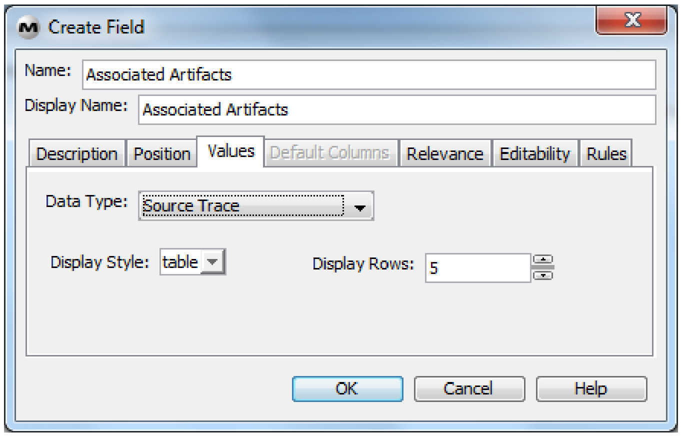

Name: Associated Artifacts
Display Name: Associated Artifacts
Data Type: Source Trace
Display Style: table
Display Rows: 5

Fig L. Creating a Source Trace Field

Note: A single item can have more than one source trace field, for example a test case could have traces to the source code that is covered by the test and links to the test scripts that implement the test case.

To add the trace, users drag the source file from a Sandbox or Project view to a source trace field in the Item>Edit view. If using the CLI or API, traces can be made even deeper in the source file, to individual lines of code. The Source Traces are readily visible by look at the resulting table in the item. From the opposite side of the trace, when working within Source, the Source Trace Viewer enables users to see all the items that have traces to the same member revision within a project context. Users access this view from a member revision, or an individual trace on an item. Depending on how the view is configured, users may perform actions on both the member revision and the items that have traces to it.

Source Traces are recommended for use in a functional safety environment. By providing a direct link between members under source configuration management and individual requirements, specifications, and test cases this feature helps ensure the software produced is authored and verified as consistent with the stated requirements, which is a key criteria of such standards as ISO 26262 and IEC 61508.

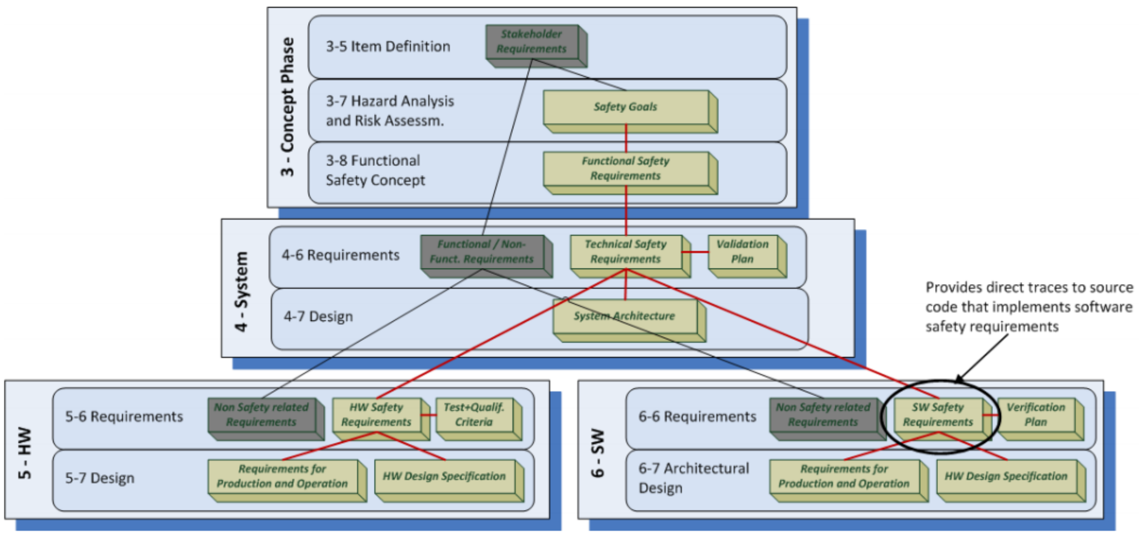

3 - Concept Phase
3-5 Item Definition
Stakeholder Requirements
3-7 Hazard Analysis and Risk Assessm.
Safety Goals
3-8 Functional Safety Concept
Functional Safety Requirements
4 - System
4-6 Requirements
Functional / Non-Funct. Requirements
Technical Safety Requirements
Validation Plan
4-7 Design
System Architecture
Provides direct traces to source code that implements software safety requirements
5 - HW
5-6 Requirements
Non Safety related Requirements
HW Safety Requirements
Test+Qualif. Criteria
5-7 Design
Requirements for Production and Operation
HW Design Specification
6 - SW
6-6 Requirements
Non Safety related Requirements
SW Safety Requirements
Verification Plan
6-7 Architectural Design
Requirements for Production and Operation
HW Design Specification

Fig M. Trace Relationships in ISO 26262

With respect to Change Management (8-8) in ISO 26262, source traces associated with Lifecycle Manager Items such as Change Requests can provide more detailed information on the action that is required, contributing to impact analysis. Traces from affected items, such as Requirements, can also be established to determine impact of the Change. (8, 8.4.3: Analysis of the change requests).

With respect to the "Additional Requirements for Management of Safety-Related Software" section of IEC61508 (3-6), Source Trace fields help to ensure requirements are satisfied and the completeness of the change. (3, 6.2.3a: Administrative and technical controls to manage changes and ensure requirements are satisfied).

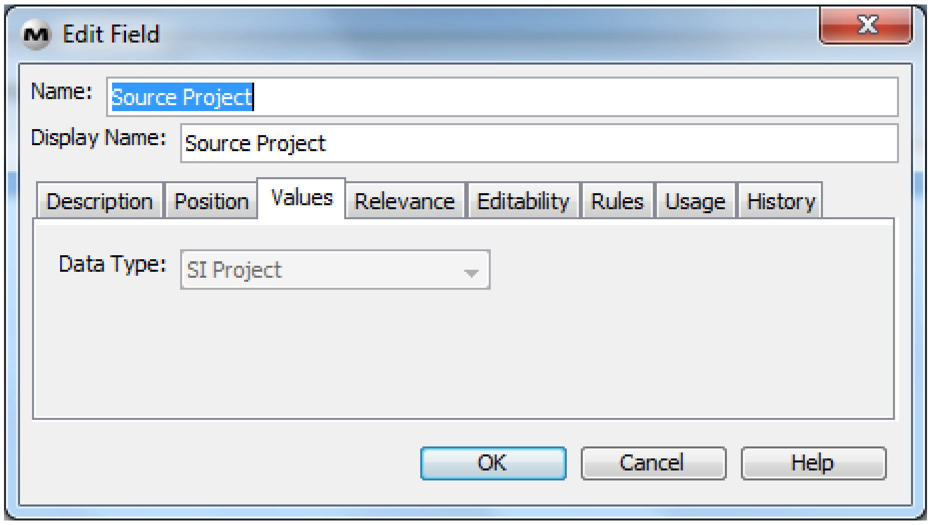
The Source Trace functionality is a relatively new feature of Lifecycle Manager and has not yet achieved widespread adoption; however customers seeking to achieve compliance with ISO 26262 have shown great interest in the feature and are actively adopting the functionality. The functionality is actively being included in future releases of the Functional Safety Solution Template, for further details, please refer to the functional safety documentation available on the Lifecycle Manager Customer Community (www.mks.com/community).

There are future plans to enhance the functionality of Source Trace fields, such as including support Source Trace Propagation in the aforementioned Trace Propagation wizard and tracing directly to lines of code in the GUI, as opposed to just in the CLI. These features are not yet scheduled for a particular release.

5.3 Source Project Fields

The Source project (or SI Project) field is created by an administrator in the admin client and made visible on one or more types of items. It allows end users to browse through an Lifecycle Manager Configuration Management repository and choose a single Configuration Management Project or Subproject. Users can also optionally specify a specific checkpoint revision or development path.

The SI Project field functionality is often used for the following reasons:

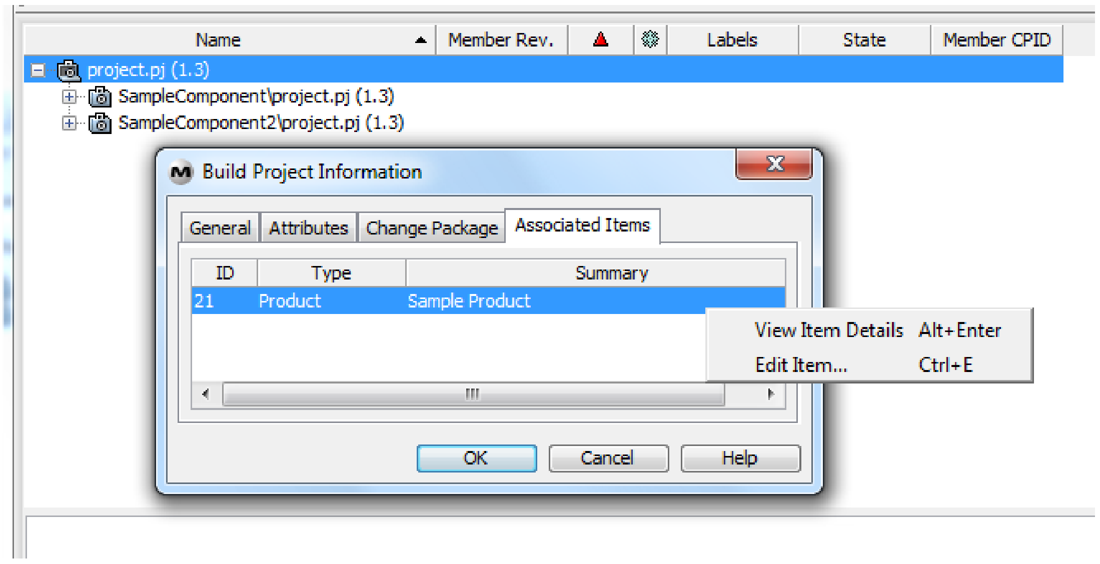

Edit Field

Name: Source Project

Display Name: Source Project

Data Type: SI Project

Fig N. Creating a SI Project field


Name	Member Rev.			Labels	State	Member CPID
project.pj (1.3)						
SampleComponent\project.pj (1.3)						
SampleComponent2\project.pj (1.3)						

Build Project Information

ID	Type	Summary
21	Product	Sample Product

Fig O. Viewing the Associated Items to a Project

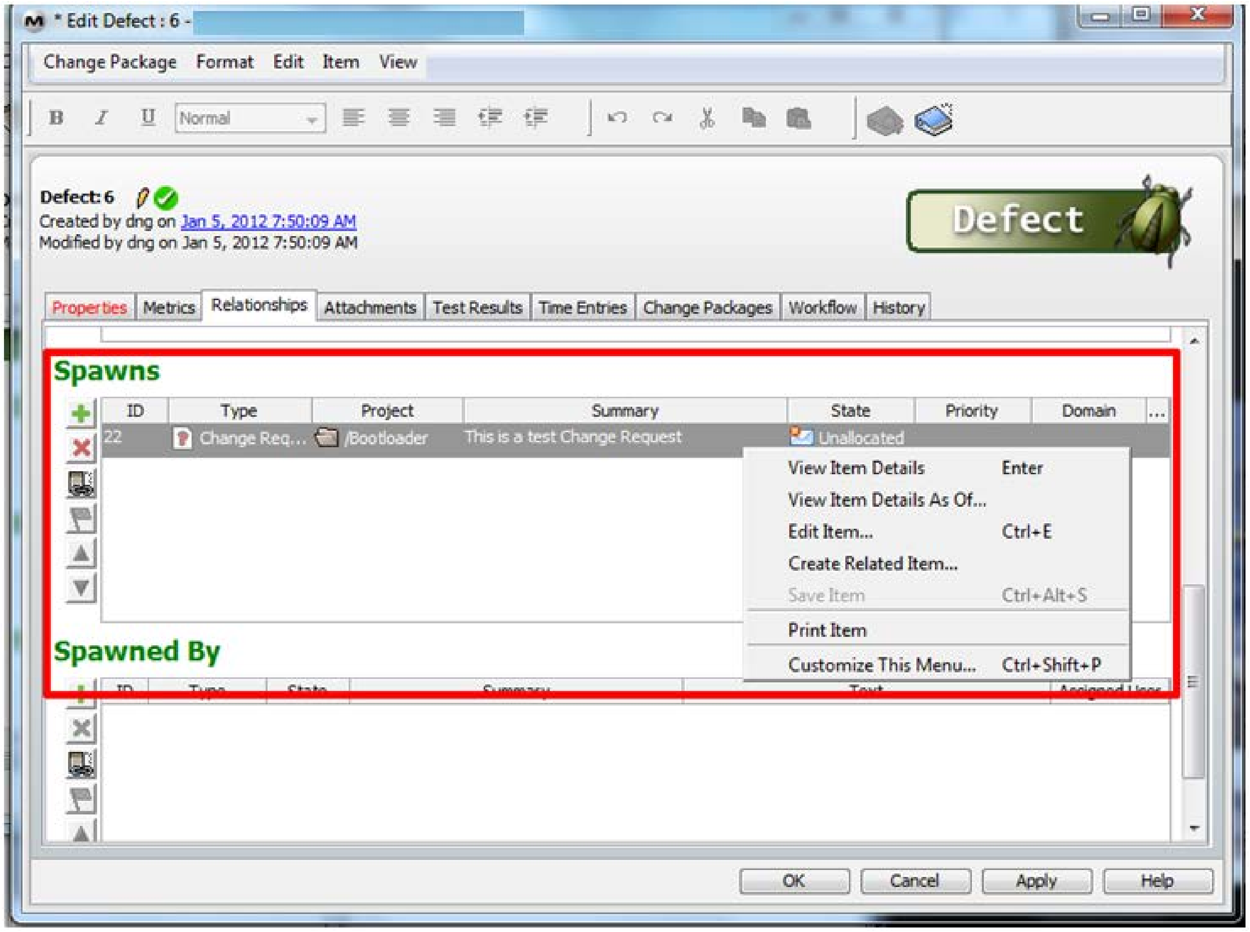
Source Type	Description	Best For	Not Recommended For
Change Package	Container of source code operations performed for resolving a specific item (e.g., add, drop, checkout, checkin, etc.)	Creating a focused audit trail of operations, automated application of changes to alternative development paths, Controlling when development occurs	Linking to a project, when multiple items need to link to the exact same
Source Project field	Link from a workflow item to a particular source project or source project checkpoint	Linking a workflow item to a particular source project, the item can be used to hold metadata or source metrics about the project	Auditability, linking to an individual file
Source Trace Field	Link from a workflow item to a particular file or file revision under source control	Linking a workflow item to a particular file or file revision, makes it easy to trace to a file that satisfies an item or provides additional details about the item instead of just an attachment	Auditability, linking to a directory of files

Summary Table of Tracing to Source options

6. Viewing Relationships

6.1 Item View

One of the most common ways to view relationships is to simply open the Item and view the field displayed. Depending upon the type of field, and how it is configured various information about the related objects will be displayed. In most cases, it will be easy to open (i.e., walk to) the related objects, be they Items or Configuration Management artifacts.



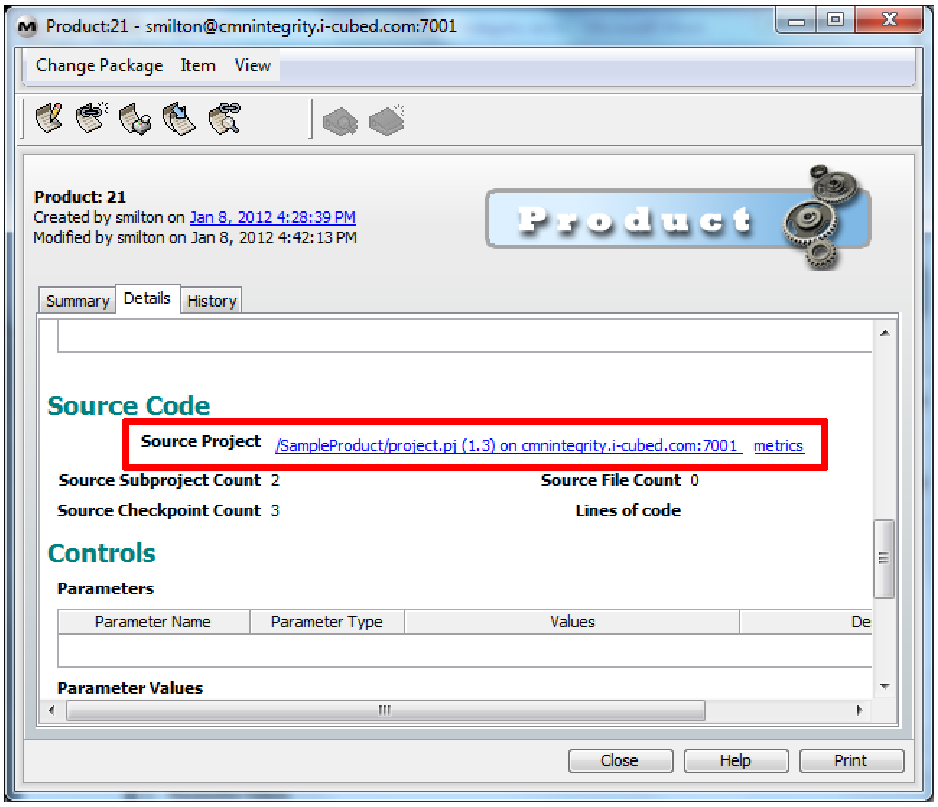
Defect: 6
Created by dng on Jan 5, 2012 7:50:09 AM
Modified by dng on Jan 5, 2012 7:50:09 AM

Spawns

ID	Type	Project	Summary	State	Priority	Domain
22	Change Req...	/Bootloader	This is a test Change Request	Unallocated		

Spawned By

Fig P. Viewing a Standard Relationship Field



Product: 21
Created by smilton on Jan 8, 2012 4:28:39 PM
Modified by smilton on Jan 8, 2012 4:42:13 PM

Source Code

Source Project /SampleProduct/project.pj (1.3) on cmnintegrity.i-cubed.com:7001 metrics

Source Subproject Count 2
Source File Count 0
Source Checkpoint Count 3
Lines of code

Controls

Parameters

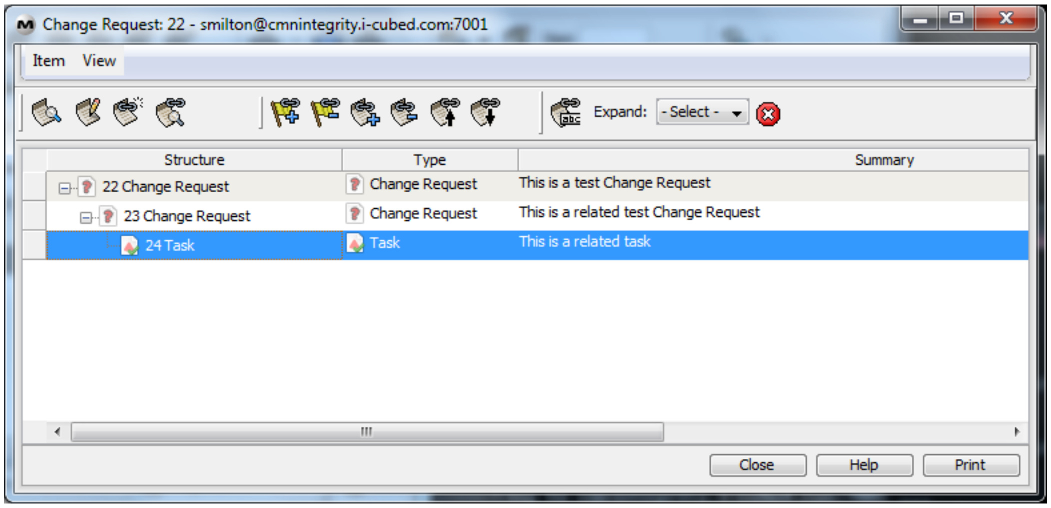
Parameter Name	Parameter Type	Values	De

Parameter Values

Fig Q. Viewing a SI Project field

6.2 Hierarchical Relationship View

The Relationships view displays all the relationships for one or more items in a tree hierarchy. You can configure the Relationships view to traverse selected relationship fields and relationship levels, as well as display specific data fields for the items in the view.



Structure	Type	Summary
22 Change Request	Change Request	This is a test Change Request
23 Change Request	Change Request	This is a related test Change Request
24 Task	Task	This is a related task

Fig R. Hierarchical Relationship View

6.3 Reports

Lifecycle Manager includes a robust reporting engine with a number of sample report templates. End users select from the available templates to create actual reports. There is full support available for traversing multiple levels of relationships in a single report, displaying information about each items. Change Package information is also available for reports, but at the current time is limited only to the change packages directly on the items being reported on.

Note: Individual report templates, sometimes referred to as report types or report recipes may or may not include designated change packages and relationship sections depending upon the choices made by the report template author.

6.4 Command Line

Relationship information is available from the command line interface and the various APIs. Some example commands:

7. Appendix 1. Sample Traces

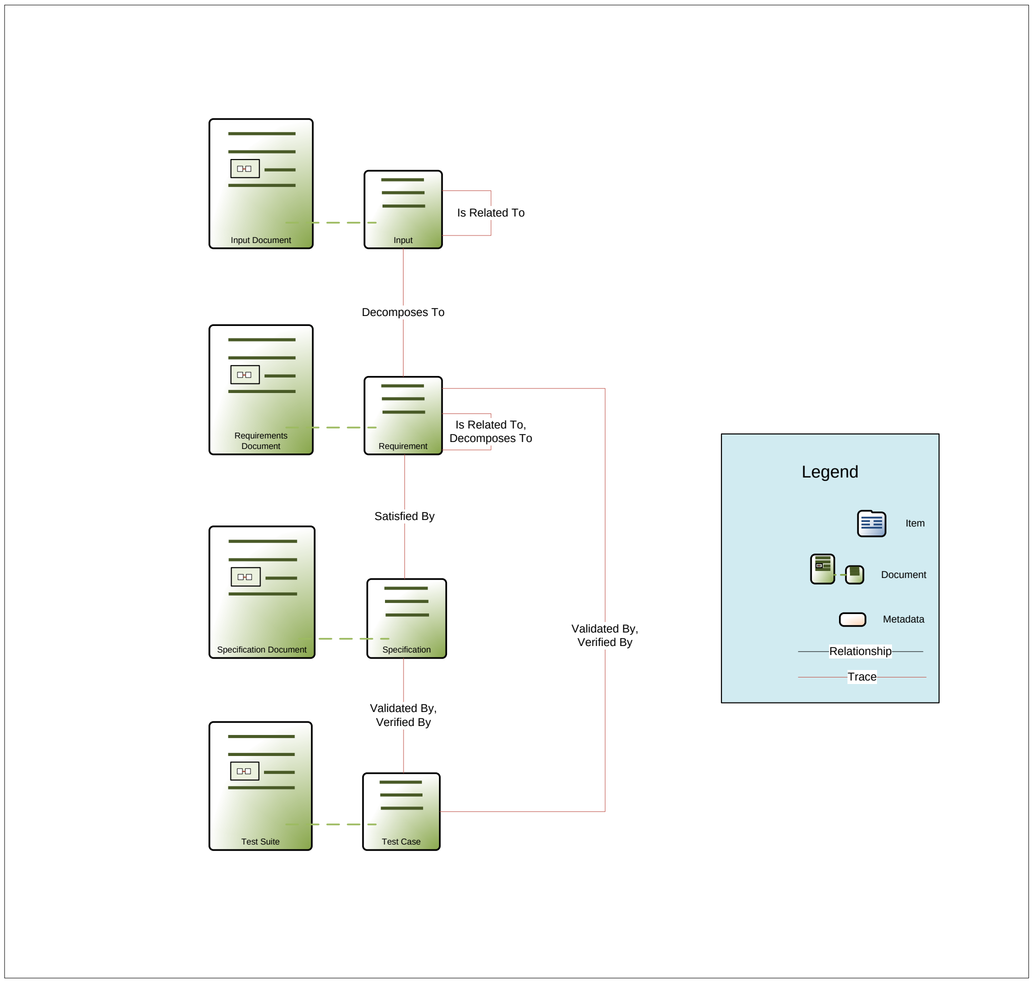
An example of the specific traces between artifact types can be found in the following table:

	Input	Model Element	Requirement	Specification	Test Case
Input	Peer Trace: Is Related To		Downstream: Decomposes To		
Model Element					Downstream: Validated By, Verified By
Requirement		Downstream: Modeled By	Peer Trace: Decomposes To, Is Related To	Downstream: Satisfied By	Downstream: Validated By, Verified By
Specification		Downstream: Modeled By			Downstream: Validated By, Verified By
Test Case					Peer Trace: Is Related To

To use this table, first select a row from the left, and then scan to the right until the desired column is reached. So as some examples:

8. Appendix 2. Sample Trace Diagram

The specific traces detailed in Appendix 2. List of Recommended Traces can be found in the following diagram:



Traceability v2.vsd

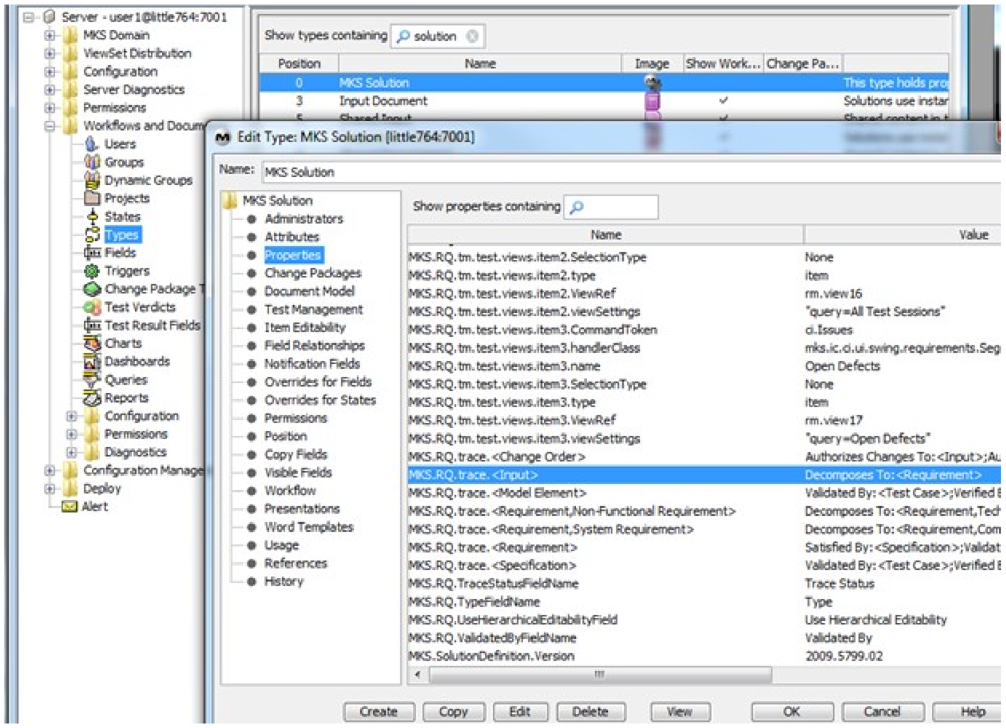
The embedded Visio document can be extracted from here:

9. Appendix 3. Creating solution properties to allow drag and drop trace creation

To set up solution properties, you must first have a solution type. The default ALM solution provides this type (named *MKS Solution*) automatically. It is possible that your custom solution will not have this type or the necessary properties needed to create traces so you must create this yourself.

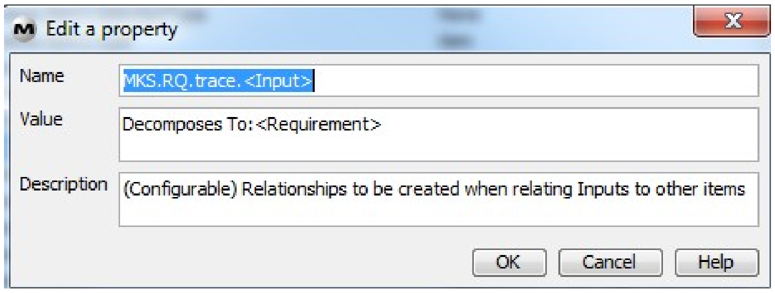
For more information, please contact your Lifecycle Manager representative to discuss setting up solutions to make sure that creating a solution type is appropriate for your situation and will not interfere with your existing system or future plans for solution installation in your environment (Only ***one*** solution type can reside on a single server).

To add (or modify existing) required properties,



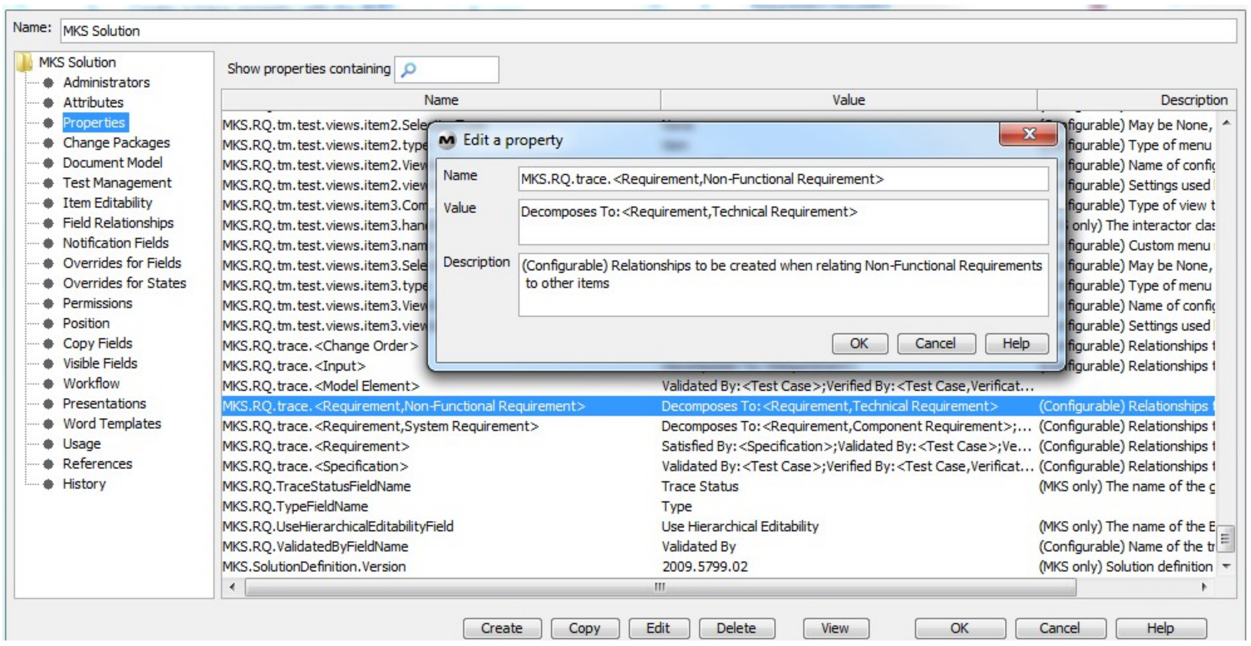
Server - user1@little764:7001
Types
Edit Type: MKS Solution [little764:7001]
Name: MKS Solution
Properties
Show properties containing
Create
Copy
Edit
Delete
View
OK
Cancel
Help

Name	Value
MKS.RQ.tm.test.views.item2.SelectionType	None
MKS.RQ.tm.test.views.item2.type	item
MKS.RQ.tm.test.views.item2.ViewRef	rm.view16
MKS.RQ.tm.test.views.item2.viewSettings	"query=All Test Sessions"
MKS.RQ.tm.test.views.item3.CommandToken	ci.Issues
MKS.RQ.tm.test.views.item3.handlerClass	mks.ic.ci.ui.swing.requirements.Seg
MKS.RQ.tm.test.views.item3.name	Open Defects
MKS.RQ.tm.test.views.item3.SelectionType	None
MKS.RQ.tm.test.views.item3.type	item
MKS.RQ.tm.test.views.item3.ViewRef	rm.view17
MKS.RQ.tm.test.views.item3.viewSettings	"query=Open Defects"
MKS.RQ.trace.<Change Order>	Authorizes Changes To:<Input>;Au
MKS.RQ.trace.<Input>	Decomposes To:<Requirement>
MKS.RQ.trace.<Model Element>	Validated By:<Test Case>;Verified E
MKS.RQ.trace.<Requirement,Non-Functional Requirement>	Decomposes To:<Requirement,Tech
MKS.RQ.trace.<Requirement,System Requirement>	Decomposes To:<Requirement,Com
MKS.RQ.trace.<Requirement>	Satisfied By:<Specification>;Validat
MKS.RQ.trace.<Specification>	Validated By:<Test Case>;Verified E
MKS.RQ.TraceStatusFieldName	Trace Status
MKS.RQ.TypeFieldName	Type
MKS.RQ.UseHierarchicalEditabilityField	Use Hierarchical Editability
MKS.RQ.ValidatedByFieldName	Validated By
MKS.SolutionDefinition.Version	2009.5799.02



Name	MKS.RQ.trace.<Input>
Value	Decomposes To:<Requirement>
Description	(Configurable) Relationships to be created when relating Inputs to other items

Traces can also be created between more specific content items, such as Requirements only of a certain category. See below for an example:



Name: MKS Solution

MKS Solution
- Administrators
- Attributes
- Properties
- Change Packages
- Document Model
- Test Management
- Item Editability
- Field Relationships
- Notification Fields
- Overrides for Fields
- Overrides for States
- Permissions
- Position
- Copy Fields
- Visible Fields
- Workflow
- Presentations
- Word Templates
- Usage
- References
- History

Show properties containing

Name	Value	Description
MKS.RQ.tm.test.views.item2.Sele		figurable) May be None,
MKS.RQ.tm.test.views.item2.type		figurable) Type of menu
MKS.RQ.tm.test.views.item2.View		figurable) Name of config
MKS.RQ.tm.test.views.item2.view		figurable) Settings used
MKS.RQ.tm.test.views.item3.Com		figurable) Type of view t
MKS.RQ.tm.test.views.item3.han		only) The interactor clas
MKS.RQ.tm.test.views.item3.nam		figurable) Custom menu
MKS.RQ.tm.test.views.item3.Sele		figurable) May be None,
MKS.RQ.tm.test.views.item3.type		figurable) Type of menu
MKS.RQ.tm.test.views.item3.View		figurable) Name of config
MKS.RQ.tm.test.views.item3.view		figurable) Settings used
MKS.RQ.trace.<Change Order>		figurable) Relationships t
MKS.RQ.trace.<Input>		figurable) Relationships t
MKS.RQ.trace.<Model Element>	Validated By:<Test Case>;Verified By:<Test Case,Verificat...	
MKS.RQ.trace.<Requirement,Non-Functional Requirement>	Decomposes To:<Requirement,Technical Requirement>	(Configurable) Relationships t
MKS.RQ.trace.<Requirement,System Requirement>	Decomposes To:<Requirement,Component Requirement>;...	(Configurable) Relationships t
MKS.RQ.trace.<Requirement>	Satisfied By:<Specification>;Validated By:<Test Case>;Ve...	(Configurable) Relationships t
MKS.RQ.trace.<Specification>	Validated By:<Test Case>;Verified By:<Test Case,Verificat...	(Configurable) Relationships t
MKS.RQ.TraceStatusFieldName	Trace Status	(MKS only) The name of the g
MKS.RQ.TypeFieldName	Type	
MKS.RQ.UseHierarchicalEditabilityField	Use Hierarchical Editability	(MKS only) The name of the E
MKS.RQ.ValidatedByFieldName	Validated By	(Configurable) Name of the tr
MKS.SolutionDefinition.Version	2009.5799.02	(MKS only) Solution definition

Edit a property

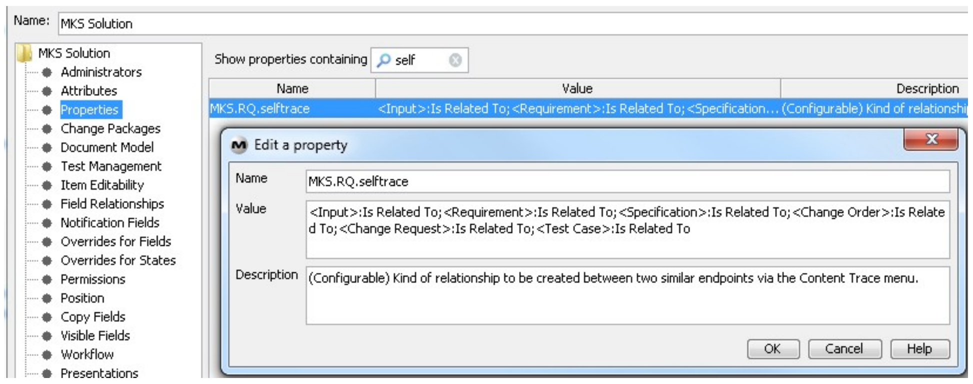
Name: MKS.RQ.trace.<Requirement,Non-Functional Requirement>

Value: Decomposes To:<Requirement,Technical Requirement>

Description: (Configurable) Relationships to be created when relating Non-Functional Requirements to other items

When two items are joined with the [Alt+] drag and drop, the solution properties are scanned for a "Best Fit". The more granular property (ie: type and category) setting that fits will "win".

You can also set a "selftrace" property. The syntax is a bit different.



Name: MKS Solution

MKS Solution
- Administrators
- Attributes
- Properties
- Change Packages
- Document Model
- Test Management
- Item Editability
- Field Relationships
- Notification Fields
- Overrides for Fields
- Overrides for States
- Permissions
- Position
- Copy Fields
- Visible Fields
- Workflow
- Presentations

Show properties containing: self

Name	Value	Description
MKS.RQ.selftrace	<Input>:Is Related To;<Requirement>:Is Related To;<Specification...	(Configurable) Kind of relationshi

Edit a property

Name: MKS.RQ.selftrace

Value: <Input>:Is Related To;<Requirement>:Is Related To;<Specification>:Is Related To;<Change Order>:Is Related To;<Change Request>:Is Related To;<Test Case>:Is Related To

Description: (Configurable) Kind of relationship to be created between two similar endpoints via the Content Trace menu.

Note: The most up to date version of these steps can always be accessed online here.

10. Appendix 4. Creating a Trace Relationship field

As an administrator, you can define which relationship fields act as trace fields between document content items. To define a relationship field as a Trace field, do the following:

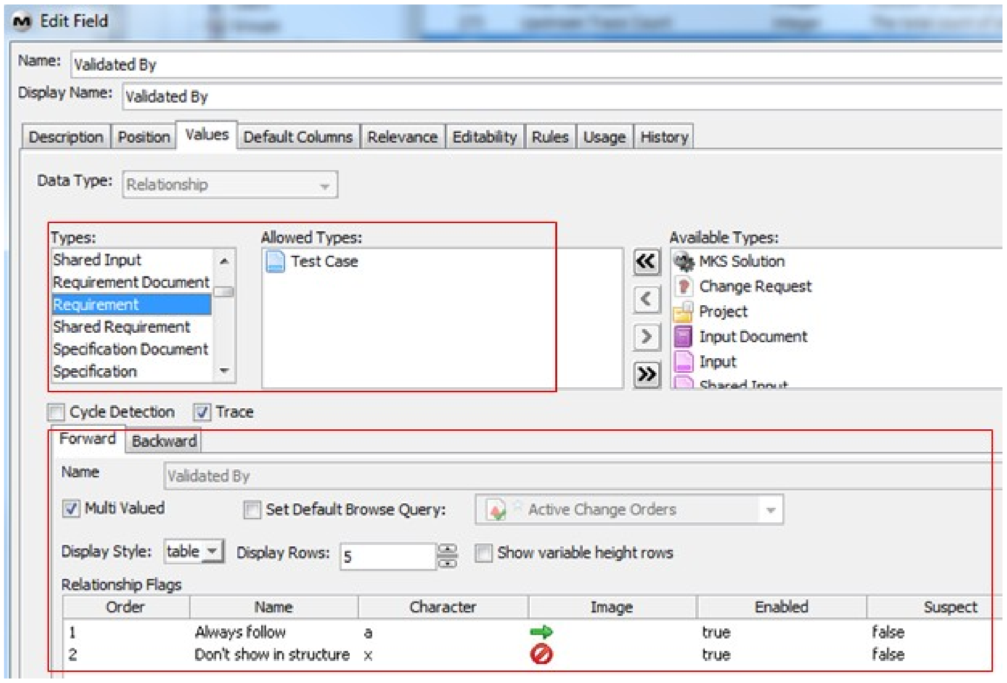
Note: When thinking of traces, it is important to consider direction. By direction, this means "downstream" or "upstream".

In Lifecycle Manager, the downstream trace (such as from a Requirement to a Test Case) is represented by the "Forward" relationship field of the relationship pair. As an example, let's look at the "Validated By/Validates" trace relationship field.

The "Forward" or "Downstream" field is the Validated By field as requirements are validated by test cases. Areas to notice about the field definition:

- The "Trace" setting: ☑ Trace **This is the important part of making the relationship field behave as a trace field.
- The Allowed Types: Requirement > Test Case
- The Forward Tab that defines whether the field can be multi valued, the display settings, and the relationship flags.
- There are usually no additional relationship flags added to the defaults here.

See below:



Edit Field

Name: Validated By

Display Name: Validated By

Description | Position | Values | Default Columns | Relevance | Editability | Rules | Usage | History

Data Type: Relationship

Types: Shared Input, Requirement Document, Requirement, Shared Requirement, Specification Document, Specification

Allowed Types: Test Case

Available Types: MKS Solution, Change Request, Project, Input Document, Input

☐ Cycle Detection ☑ Trace

Forward | Backward

Name: Validated By

☑ Multi Valued ☐ Set Default Browse Query: Active Change Orders

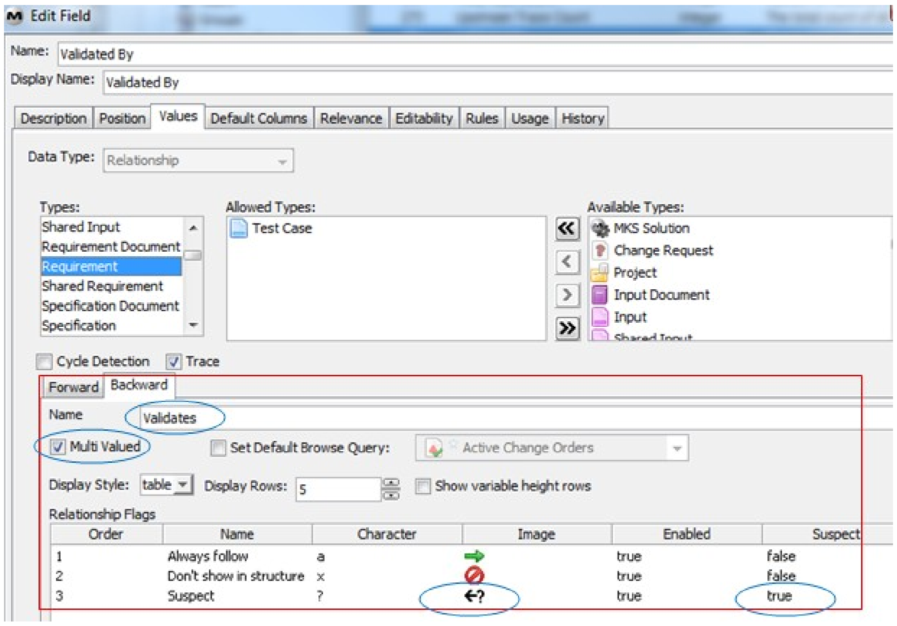
Display Style: table Display Rows: 5 ☐ Show variable height rows

Relationship Flags

Order	Name	Character	Image	Enabled	Suspect
1	Always follow	a	➡	true	false
2	Don't show in structure	x	⊘	true	false

- The Backward Tab that allows you to set values for the Upstream field (in this case, the Validates field)
- This tab only controls the values set in that tab. All other values above in the field definition are still regarding the forward field.
- So in this tab, you will set the name of the backwards field, identify if multi valued, set the display information, and define additional relationship flags.
- For the backward relationship field, you would likely set the suspect flag.

See below:



Edit Field

Name: Validated By

Display Name: Validated By

Description | Position | Values | Default Columns | Relevance | Editability | Rules | Usage | History

Data Type: Relationship

Types: Shared Input, Requirement Document, Requirement, Shared Requirement, Specification Document, Specification

Allowed Types: Test Case

Available Types: MKS Solution, Change Request, Project, Input Document, Input, Shared Input

☐ Cycle Detection ☑ Trace

Forward | Backward

Name: Validates

☑ Multi Valued ☐ Set Default Browse Query: Active Change Orders

Display Style: table Display Rows: 5 ☐ Show variable height rows

Relationship Flags

Order	Name	Character	Image	Enabled	Suspect
1	Always follow	a	➡	true	false
2	Don't show in structure	x	🚫	true	false
3	Suspect	?	←?	true	true

Note: The most up to date version of these steps can always be accessed online here.