Dimensions® RM 12.11.1

Integration Guide for ALM/Quality Center

Product version: 12.11.1

Publication date: March 2023

Contents

Preface

This document describes the Dimensions RM integration with ALM/Quality Center.

The integration synchronizes data between Dimensions RM and ALM, and automatically generates new ALM items in response to Dimensions RM events and new Dimensions RM objects in response to ALM events.

The heart of the integration is the Dimensions RM Synchronization Engine (Sync Engine), which runs as a Windows service. The Sync Engine reads an XML configuration file that describes how to map data between the two products. The synchronization occurs at a configurable regular interval.

For detailed information on ALM, refer to the manuals available with your ALM installation.

Objective

The purpose of this guide is to describe how to integrate Dimensions RM with ALM.

Audience

The audience for this manual is administrators who have extensive domain knowledge about the Dimensions RM and ALM data sources to be synchronized.

Manual Organization

Chapter/ Appendix	Description
1	Provides an introduction to the integration and the Sync Engine.
2	Describes how to set up Dimensions RM to integrate with ALM.
3	Describes how to set up ALM to integrate with Dimensions RM.
4	Describes how to configure the Sync Engine.
5	Provides information about running the Sync Engine.
6	Provides use cases that demonstrate the RM-ALM integration.
A	Describes events, triggers, and actions.
B	Lists the ALM datatypes.

Contacting Micro Focus Technical Support

Micro Focus provides technical support for all registered users of this product, including limited installation support for the first 30 days. If you need support after that time, contact Micro Focus Support at the following URL and follow the instructions:

http://supportline.microfocus.com

Language-specific technical support is available during local business hours. For all other hours, technical support is provided in English.

You can use the Micro Focus Support Web page to:

- Report problems and ask questions.
- Obtain up-to-date technical support information, including that shared by our customers via the Web, automatic e-mail notification, newsgroups, and regional user groups.
- Access a knowledge base, which contains how-to information and allows you to search on keywords for technical bulletins.
- Download updates and fix releases for your Micro Focus products.

License and Copyright Information for Third-Party Software

For license and copyright information of third-party software included in this release, check the file `Third_Party_Licenses.txt`, which can be found in the Dimensions RM installation directory, e.g. C:\Program Files (x86)\Micro Focus\Dimensions 12.11.1\RM.

Chapter 1
Introduction

What Is the RM-ALM/Quality Center Integration?

The RM-ALM/Quality Center integration enables information to be synchronized between Dimensions RM and HP ALM/Quality Center. Specifically, you can do any of the following:

- When new objects are added to a designated collection in Dimensions RM, trigger the creation of new items in ALM.
- When existing objects in Dimensions RM are updated, trigger updates in the corresponding items in ALM. In ALM 10, create new versions of the items if version control is enabled.
- When existing objects in Dimensions RM are marked as deleted, transition the corresponding ALM items to an inactive state and dissolve the link between the Dimensions RM objects and the ALM items.
- Trigger the creation of new objects in Dimensions RM when new items are created in ALM.
- Trigger updates in existing objects in Dimensions RM when the corresponding items in ALM are updated.
- Mark objects in Dimensions RM as deleted when the corresponding items in ALM become inactive or are deleted.
- Synchronize the relationship between objects, such as requirements, test cases, and defects between ALM and RM. When a relationship is added or modified in one tool, the relationship is then modified in the other tool.
- Synchronize changes to specific versions of objects in ALM to the corresponding versions of related objects in RM, and vice versa. The integration supports version control in ALM. If version control is enabled for the ALM project, when you synchronize, RM checks out the item, updates it, and then checks it in.

Who Should Use the RM-ALM/Quality Center Integration?

The RM-ALM integration is for Dimensions RM or ALM administrators who need to propagate changes in information between the two systems. For example, project managers using Dimensions RM to track requirements can edit those requirements in Dimensions RM, and the integration will send the information in those requirements to read-only fields within ALM items. Engineering can then view and use (but not edit) the information in those fields as they work on the project.

What Do I Have to do to Install and Use the RM-ALM Integration?

On the machine that runs Dimensions RM, a Windows service (the Sync Engine) for the RM-ALM integration is installed as an option (but not started) when Dimensions RM is installed.

NOTE The system on which Dimensions RM is installed must have access to the ALM database. See "Specifying the ALM Data Source" on page 27.

The ALM Connectivity tool enables HP ALM integration with third-party tools. Please ensure that ALM Connectivity is executed and the client registered before attempting to integrate RM with ALM.

To enable connectivity:

1. On the RM server running the SyncEngine, go to the HP ALM start page.
2. Under **Application Lifecycle Management**, click **Tools**.
3. From the **Tools** list, select **ALM Connectivity**.
4. Download and execute the **ALM Connectivity tool**.
5. Return to the **Tools** page, and register the client: **ALM Client Registration**.

Who Should Configure the RM-ALM Integration?

The RM-ALM integration is installed as an option with Dimensions RM, but the integration must be configured for each company's needs. The RM-ALM integration should be set up and administered by a user with administrative privileges.

How Often Does the RM-ALM Integration Synchronize the Data?

The RM-ALM integration can be set to synchronize data in intervals as short as a minute, or in much longer intervals of several hours or days. For best results, do not use an interval longer than one day. Shorter intervals keep data more up-to-date but create additional stress on system resources.

The synchronization interval is one of the settings that you adjust specifically for your site. You specify the interval in the XML configuration file. See "Specifying General Options" on page 26.

How Does the RM-ALM Integration Work?

The heart of the integration is the Dimensions RM Synchronization Engine (Sync Engine), which runs as a Windows service. Synchronization occurs at a configurable regular interval. The integration is controlled through a configuration file in XML format that describes how to map data between the two products. This file defines events, triggers, and actions. The integration watches for certain events, and triggers specific actions accordingly. For example, a named Dimensions RM collection can be watched for the addition of a new object to the collection. When that happens, the trigger for that event can be set to create a ALM item that is linked to the Dimensions RM object.

NOTE Because there is no separate graphical user interface for the integration, all behavior changes must be effected through the XML configuration file. By default, this file is named `config.xml`, and it resides in the `conf` subdirectory of the Dimensions RM installation.

API Usage and Limitations

All communication between RM and ALM is via the ALM API. Any direct calls to the ALM database are also made via the ALM API.

With a SAAS provider, if SYNCENGINE_QC_TABLE_TRIGGERS will not run, then creating or deleting links between objects in ALM will not create or delete links between objects in RM.

With a SAAS provider, if the SYNCENGINE_QC_SCHEMA will not run, no sychronization functions will work with the exception of create functionality.

Configuration File Usage

The configuration file controls the following conditions:

- The execution of actions in ALM based on "trigger" events in RTM, or the execution of actions in Dimensions RM based on "trigger" events in ALM. See "Specifying Events and Actions" on page 32.
- The set of classes in Dimensions RM that are monitored for changes.

What If I Am Also Using Another Dimensions RM Integration?

If you are using more than one Dimensions RM integration (for example, one RM-ALM integration and one RM-SBM integration, or two RM-ALM integrations involving different Dimensions RM instances), you must take these extra steps:

- Install another instance of the Sync Engine service for the second integration.
- Make sure that the two Sync Engine services have different names.
- Make sure that the two Sync Engine services use different configuration files.

Chapter 2

Use Cases

This chapter contains use cases for the Dimensions RM integration with ALM.

Scoping a Requirement

1. The requirement is created in Dimensions RM.
2. When the requirement is ready to be scoped or executed, it is added to a designated collection.
3. Based on the specific configuration and the configured class and collection, the Sync Engine retrieves the object data.
4. Based on the mapped attributes in the Sync Engine configuration, a corresponding requirement is created in ALM.
5. Periodically the Sync Engine checks for updates on the object and transfers the changes as configured in the update event specified in the configuration.

A requirement has been created in RTM and is now ready for scoping or execution. The requirement is added to the collection 'QUALITY CENTER'. This triggers the transition of the requirement object from Dimensions RM to ALM.

Requirement Change (RM -> ALM)

1. An existing requirement is changed in Dimensions RM.
2. Based on the specific configuration and the configured class, the Sync Engine retrieves the object data.
3. Based on the mapped attributes, the corresponding requirement in ALM is updated to match the requirement in Dimensions RM. If version control is enabled in ALM 10, a new version of the requirement is created.
4. Periodically, the Sync Engine checks for updates and transfers the changes as configured in the update event specified in the configuration file.

Requirement Deletion (RM -> ALM)

1. A requirement is deleted in Dimensions RM.
2. Based on the specific configuration and the configured class, the Sync Engine retrieves the object ID.
3. The corresponding requirement in ALM is marked as deleted or is deleted permanently. This depends on the configuration file.
4. Periodically, the Sync Engine checks for deletions within Dimensions RM and synchronizes the data with ALM

Defect Detection (ALM -> RM)

1. Running the tests in ALM yields a defect, because of a failed test.
2. A new defect is created in ALM and triggers a configured Sync Engine event.
3. The RM-ALM integration retrieves the data for the defect object.
4. When the configured conditions are met, a corresponding object is created in Dimensions RM.
5. Periodically, the Sync Engine checks for updates or deletes and transfers the changes to Dimensions RM. In case of delete events, the object in Dimensions RM will be marked as deleted.
6. A defect is detected in ALM. The transition rules specified in the Sync Engine configuration define a create event in Dimensions RM. This event creates a new object in the class QCDefect.

Test Creation (ALM -> RM)

1. A test is created in ALM.
2. A create event is triggered by the Sync Engine and the Sync Engine retrieves the object data based upon the specification in the configuration file.
3. A corresponding object in the Test class is created in Dimensions RM and filled with data.
4. Periodically, the Sync Engine checks for updates of the Test object, and in case of updates, the changes will be transitioned to Dimensions RM.

Test Update (ALM -> RM)

1. A test is changed in ALM.

2 The Sync Engine retrieves the updated test.

3 The change is transitioned to the corresponding Dimensions RM test based on the configuration of the update event in the configuration file.

4 Periodically, the Sync Engine checks for updates and transfers the changes as specified in the configuration file.

Test Deletion (ALM -> RM)

1 A test is deleted in ALM.

2 The Sync Engine detects the deletion of the test.

3 As specified in the configuration of the Sync Engine, the corresponding test object in Dimensions RM is marked as deleted.

Chapter 3

Setting Up Dimensions RM

Overview

The integration between Dimensions RM and ALM shares data by transferring data between objects in Dimensions RM and objects in ALM. The transfer is based on mappings between attributes in Dimensions RM and fields in ALM.

Adding Classes

You must add classes to Dimensions RM, which then hold the data transferred from ALM. The types and number of Dimensions RM classes depend on how you plan to store information in Dimensions RM. At a minimum, to ensure that relationships between items types in ALM and Dimensions RM are correctly synchronized between the applications, you must add classes to Dimensions RM for test cases and defects. Then, you must configure relationships correctly to ensure that relationship information is synchronized. Follow the steps in Chapter 5, "Configuring the Sync Engine" on page 23 to map the classes and relationships to ALM items and attributes.

Adding Attributes

For each Dimensions RM class, add the necessary attributes to correlate with the fields in ALM. For example, when mapping to the ALM Tests module, you must add the alphanumeric attribute "Test Name" to your Dimensions RM class. This attribute will map to Test Name, a required field in ALM.

Required Fields

The following fields are required:

- Requirement: "Name" must be unique
- Test: "Test Name" must be unique, "Subject" for location
- Testset: "Test Set" must be unique, "Test Set Folder" for location
- Defect: "Summary"

Chapter 4

Setting Up ALM

This chapter describes the workflow used to set up ALM to integrate with Dimensions RM. This includes setting up projects, fields, users, and creating a database schema for the Sync Engine.

Defining Projects and Fields

NOTE Before mapping attributes to fields, you must create Dimensions RM classes that correspond to ALM Requirements, Tests, TestSet and Defects modules. These classes must have the necessary attributes to map to ALM fields.

ALM classnames must be Requirement, Test, TestSet and Defect in the XML configuration file. ALM fields can be field name or field label.

Define or identify the project, and fields in the project that you want to receive values from linked objects in Dimensions RM. Consider the field types as you map the fields. ALM accepts values from the foreign data source as text and converts the value, if necessary. If an incoming text string is too long, ALM truncates it. It is not possible to map date fields; you can only map a date field to a text field. If you map ALM selection fields to Dimensions RM list attributes, the values must match exactly or be mapped using a <ValueMap …> element in the XML configuration file. For best results, make the fields receiving Dimensions RM data in the ALM database read-only so that browser users cannot modify those fields. They represent Dimensions RM data that normally should be changed only within Dimensions RM.

You do not have to map all fields; however, you must map the fields that are defined as required in ALM or mandatory in Dimensions RM. The Dimensions RM attribute, to which you map mandatory classes, should be set to mandatory. This prevents problems that would occur if you try to transfer an object from Dimensions RM with the attribute empty. You cannot map the internal ALM fields like Defect ID (for Defects) and ReqID (for Requirements). These values are used internally by ALM, and changing them could corrupt your ALM data. If an attribute in ALM contains HTML formatting, it will be deleted in Dimensions RM. Certain ALM fields refer to ALM users, such as the "Detected By" field in the default ALM Test module. The transferred value (from Dimensions RM) must be a valid user in ALM.

Setting Up ALM Users

You will need an administrative user account in ALM with project administration privileges. This user should be reserved just for the synchronization.

ALM Database Setup Steps

Creating the Interface Schema

Before you can run the Sync Engine with ALM, you must set up a database schema on your ALM database instance that stores information about the synchronized objects.

To set up this schema:

1 Open one of the following files in a text editor:

(Oracle) \<*Install location*>\conf\SYNCENGINE_QC_SCHEMA.sql

(SQL Server) \<*Install location*>\conf\SYNCENGINE_QC_SCHEMA_SQL.SQL

2 For Oracle, set the location of your tablespace datafile to the correct location and name for your environment. To do this, update the following path:

```
D:/ORACLE/ORADATA/RTM/Syncengineqc1.ora
```

3 For SQL Server, set the correct SQL Server installation location. To do this, update each instance of the following path:

```
C:\Program Files\Microsoft SQL
Server\MSSQL\Data\SYNCENGINE_QC_Data.MDF
```

4 For Oracle, change `RTMSYNC_INTERFACE_DB` to the correct dataschema name of your ALM project.

5 Save the file.

6 To execute the file:

- On Oracle, log in to the Oracle database as the DB administrator, then run the file.
- On SQL Server, open a command prompt and run the following command from the /conf directory, where the file resides:

  ```
  osql.exe" -E -i SYNCENGINE_QC_SCHEMA_SQL.SQL
  ```

  Or, if SQL Server has no built in authentication:

- `with osql.exe" -U -i SYNCENGINE_QC_SCHEMA_SQL.SQL`

7 You must then grant access to the Sync Engine schema to the ALM Oracle schema, as in the example below. This example is for a ALM Oracle schema called `DEFAULT_QCPROJ_DB`. The schema name format is <*domain*>_<*project*>_DB. In this example, `DEFAULT` refers to the domain, and `QCPROJ` refers to the project name. To find the project and domain names, see the ALM login dialog box.

```
GRANT DELETE ON  "SYNCENGINE_QC"."SYNC_XREF" TO "DEFAULT_QCPROJ_DB";
GRANT INSERT ON  "SYNCENGINE_QC"."SYNC_XREF" TO "DEFAULT_QCPROJ_DB
     ";
GRANT SELECT ON  "SYNCENGINE_QC"."SYNC_XREF" TO "DEFAULT_QCPROJ_DB
     ";
GRANT UPDATE ON  "SYNCENGINE_QC"."SYNC_XREF" TO "DEFAULT_QCPROJ_DB
     ";
```

Setting Up the Link Triggers

To enable the synchronization of link information, you must also run a SQL script called SYNCENGINE_QC_TABLE_TRIGGERS.SQL from the <*install location*>\conf directory.

To run these scripts and set up the link triggers:

1 Before running this script, edit it to replace **<*Schema Name*>** with the ALM project schema name, such as **DEFAULT_QCPROJ_DB**. The schema name format is **<*domain*>_<*project*>_DB**. In this example, **DEFAULT** refers to the domain, and **QCPROJ** refers to the project name. To find the project and domain names, see the ALM login dialog box.

2 Run the SYNCENGINE_QC_TABLE_TRIGGERS.SQL file. Follow the appropriate steps for either Oracle or SQL:

a On Oracle, log in to the Oracle database as the database administrator, then run the file. After running the file, to verify whether the Triggers are installed correctly, execute the following queries one by one and check that the output of the queries is 1:

```
select count(*) from all_triggers where
    trigger_name='SYNCENGINE_UPDT_TRG_LINK' and TABLE_NAME='LINK'
    and table_owner= '< SCHEMA_NAME>';
select count(*) from all_triggers where
    trigger_name='SYNCENGINE_UPDT_TRG_REQ_CVR' and
    TABLE_NAME='REQ_COVER' and table_owner= = '< SCHEMA_NAME>';
```

b On SQL Server, open a command prompt and run the following command from the /conf directory, where the file resides:

```
osql.exe" -E -i SYNCENGINE_QC_TABLE_TRIGGERS_SQL.SQL
```

Or, if SQL Server has no built in authentication:

```
osql.exe" -U -i SYNCENGINE_QC_TABLE_TRIGGERS_SQL.SQL
```

After creation to verify whether the Triggers are installed correctly, execute the following queries one by one and check whether the output of the following queries is 1.

```
use [<SCHEMA_NAME>]
select count(*) from sysobjects where xtype='TR' and name =
    'SYNCENGINE_UPDT_TRG_LINK';
select count(*) from sysobjects where xtype='TR' and name =
    'SYNCENGINE_UPDT_TRG_REQ_CVR';
```

Gathering Information about the ALM Data Model

Before you begin mapping fields and configuring synchronization, note all mandatory field names and their characters (such as whether they are integer, text, alphanumberic etc.). You will need this information, as well as corresponding field information in Dimensions RM, when you define the mappings in the config.xml file.

IMPORTANT! Note that each field to be synchronized must have history enabled by the ALM administrator.

Chapter 5

Configuring the Sync Engine

Installing the Integration

- The Sync Engine is automatically installed if you choose the **SyncEngine** feature in the Dimensions RM installer.
- You must have access to an ALM/Quality Center installation and a Dimensions RM installation. For information on installing these products, see the appropriate documentation for the product.

Installing the Local Client for ALM

The RM-ALM integration requires that the ALM client is installed on the local machine (where you plan to install the integration). The ALM client installs automatically the first time that you access ALM through your browser.

IMPORTANT! You must repeat the installation of the local client if you used it with any release before RM 12.8.

To install the Local Client, execute these steps:

1 Open Internet Explorer.

2 Enter the URL for your environment into the URL box of Internet Explorer.
Format: *http*://*server:port*/qcbin/start_a.htm

NOTE Use the URL as specified above and change the following:

- **http:** Change to `https` if you are using secure connections on your ALM server. If you are not using secure connections, use `http`.
- **server:** Change to the name of your ALM server
- **port:** Change to the port number of your ALM server

If you fail to install the ALM client, you cannot log into ALM when you launch the RM-ALM tool.

3 Locate the OLEView tool (OLE/COM Object Viewer) for Windows.

If OLEView is not installed on your system, download the Windows SDK and install it.

4 Right-click **`oleview.exe`** and select **Run as administrator** from the shortcut menu.

5 In the left pane, expand **Object Classes**, then expand **All Objects**.

6 Select **Mercury OTA Client**. The implemented interfaces are displayed in the Registry tab in the right pane.

7 Select the **Implementation** tab.

8 Select the **Use Surrogate Process** option. Wait until the DLL surrogate process is completed.

Upgrading from RM 2009 R2 or Earlier

If you used the sync engine in RM 2009 R2 or earlier, changes are required to the configuration file when upgrading to a newer version of RM.

Actions to create folders in ALM are no longer required and must be removed from the configuration file. The sync engine now automatically creates folders if they do not exist. The sync engine does not move existing ALM requirements from one folder to a new folder during an update action. It would create the requirements in the new folder and not delete the old ALM requirements.

NOTE The RTM Category in ALM is no longer required and can be removed.

ALM Specific Attributes

- RQ_TYPE_ID is used to determine the requirement type in ALM. It should NOT be used to force folder creation. RQ_TYPE_ID is new in ALM 10. It must have numeric values; a value map is suggested to list the names of the ALM types and connect them to numbers. RQ_Type is a deprecated attribute from ALM 9 and is no longer used. _RQ_TYPE_ID must be the last attribute in the action section.
- QC_FOLDER is valid only for ALM Requirement class. It is the name of the folder for the ALM requirements that include the full ALM path.
- QC_FOLDER_TYPE is the ALM system ID (Number) for the type. It allows creating a hierarchical structure as allowed in ALM as any type.

About the XML Configuration File

The first task that the Sync Engine performs each time that it is started is to read the configuration file (*RM_Install_Dir*\conf\config.xml by default). This file specifies the data sources with which the Sync Engine will communicate, the events that it will process, and the mappings of data between the specified data sources.

Whenever you modify the configuration file, you must restart the Sync Engine service in order for your changes to take effect.

The administrator must have extensive domain knowledge about the data sources to be synchronized in order to construct the field and object mappings and to identify triggering events.

Character Encoding and Text Editor Considerations

To modify the XML configuration file, use a plain-text editor that can save the file in the character encoding specified at the top of the file, typically UTF-8.

NOTE Microsoft® Notepad can save with UTF-8 encoding, but that is not the default, so be careful to select the correct encoding when you save the file.

For best results, use a text editor that specializes in markup languages such as HTML and XML. These editors usually provide many conveniences, including syntax highlighting and validation.

XML Entities - Escaping Characters

The XML format specification requires that some characters have to be replaced when adding them as values into an XML file.

Character	Entity	Plain Text	XML / Comment
"	&quot;	This is a "good" example.	This is a "good" example.
&	&amp;	Smith & Co.	Smith & Co.
’	&apos;	Jack’s house	Jack' house You only need to use ' if it is used within an attribute that uses apostrophes to surround the text.
<	&lt;	Price < $ 30.00	Price < $ 30.00
>	&gt;	Price > $ 30.00	Price > $ 30.00

Sample XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<sample>
<quotation>This is a &quot;good&quot; example.</quotation>
<ampersand>Smith &amp; Co.</ampersand>
<house value='Jack&apos;s house' />
<house value="Jack's house" />
<house>Jack's house</house>
<lessThan>Price &lt; $ 30.00</lessThan>
<greaterThan>Price &gt; $ 30.00</greaterThan>
</sample>
```

Specifying General Options

```
<Sync interval="15" sleep="15" maxcycles="1"/>
```

The `interval` attribute controls the amount of time between the beginning of one cycle and the beginning of the next cycle; that is, the Sync Engine will run every *<interval>* minutes.

The `sleep` attribute specifies the number of seconds between event-triggered actions. This enables you to exercise some control over system performance.

Note that `interval` is specified using minutes, but `sleep` is specified using seconds.

The `maxcycles` attribute determines the number of full synchronization cycles that are run at one time. When testing synchronization, it is best to set this to `1`.

Specifying Data Source Providers

About Data Source Providers

Data source providers are DLLs that serve as proxies for communicating with their respective data sources.

In the XML configuration file, the `<Provider ...>` elements specify the locations of the DLLs used by the Sync Engine to communicate with the Dimensions RM and ALM data sources. The `name` and `location` attributes are required, but the `description` attribute is optional and is used for logging purposes.

Data Source Definition Example

The following example illustrates very simply syntax for defining data sources:

```
<Provider name="RTM" location="syncrtm.dll" description="Dimensions RM
    Sync Proxy" />
<Provider name="QC" location="QCSyncProxy.dll" description=" ALM Sync
    Proxy" />
```

Continue to the following sections for more details.

Specifying the ALM Data Source

Summary

The `<DataSource ...>` elements contain connection information for a specific data source. This information is sent to the data source when the Sync Engine tries to connect. You must define the datasource for both ALM and Dimensions RM.

Example Syntax

The ALM data source specification should look something like the following:

```
<DataSource name="QCSource" provider="QC">
  <Param name="user" value="admin" />
  <Param name="password" value="xxxx"/>
  <Param name="host" value="TEST_RTM"/>
  <Param name="project" value="QCPROJ"/>
  <Param name="domain" value="DEFAULT"/>
  <Param name="qcbin" value="http://win-2k-vm:8085/QCBIN"/>
  <Param name="sql" value="T-SQL"/>
  <Param name="jvm" value="C:\\Program Files\\Micro Focus\\Dimensio
  ns 12.11.1\\Common Tools 1.8.6.0\\jre\\11.0\\bin\\client\\jvm.dll
  " /> <!-- Replace with the path in your environment. -->
  <Param name="classpath" value="-Djava.class.path=C:\\Program
  Files\\Micro Focus\\Dimensions 12.11.1\\RM\\bin\\SyncEngUtilsJava
```

```
        .jar;." /> <!-- Replace with the path in your environment. -->
        <Param name="class" value="com/serena/SyncUtils" />

    </DataSource>
```

The information is contained in `<Param ...>` elements, and though it is usually in the form of name/value pairs, it may be in whatever format the data source provider requires.

NOTE You can encrypt any parameter using the Sync Engine. Use the command-line option `-p` to retrieve a sample parameter with the value encrypted, and then paste that encrypted value into the configuration file. See "Controlling the Sync Engine from the Command Line" on page 48.

Guidelines

Review the following guidelines when defining these options

- `<DataSource name="QCSource" provider="QC">`: The DataSource can be set to any name, however this name needs to be set to the same value throughout the config.xml file.
- `<Param name="user" value="admin" />`: The user here is the ALM user account that the Sync Engine will use to access ALM. it is strongly recommended that you create a special ALM user account with API/Database access for use by the integration.
- `<Param name="password" value="xxxx"/>`: The password for the user account above.
- `<Param name="host" value="TEST_RTM"/>`: The name of the desired ALM domain as displayed on the ALM login page.
- `<Param name="domain" value="DEFAULT"/>`: The name of the ALM domain that contains the project to be synchronized.
- `<Param name="project" value="QCPROJ"/>`: The name of the ALM project to be synchronized.
- `<Param name="qcbin" value="http://win-2k-vm:8085/QCBIN"/>`: The URL for the ALM client. The port depends on the Web server setup for ALM.
- `<Param name="sql" value="T-SQL"/>`: This entry is mandatory for SQL Server databases.

IMPORTANT! Either the **host** or **domain** parameter can be used to specify the value of the ALM domain. If both are included, the last one to appear will be used.

Specifying the Dimensions RM Data Source

Summary

The Dimensions RM data source specifies the connection information for Dimensions RM.

Example Syntax

The Dimensions RM data source specification should look something like the following:

```
<DataSource name="RMSource" provider="RM">
   <Param name="user" value="sync_userid" />
   <Param name="password" value="xxxx" />
   <Param name="host" value="RM_DB_name" />
   <Param name="jvm" value="C:\\Program Files\\Micro Focus\\Dimensio
   ns 12.11.1\\Common Tools 1.8.6.0\\jre\\11.0\\bin\\client\\jvm.dll
   " /> <!-- Replace with the path in your environment. -->
   <Param name="classpath" value="-Djava.class.path=C:\\Program
   Files\\Micro Focus\\Dimensions 12.11.1\\RM\\bin\\SyncEngUtilsJava
   .jar;." /> <!-- Replace with the path in your environment. -->
   <Param name="class" value="com/serena/SyncUtils" />
</DataSource>
```

Guidelines

Review the following guidelines when defining these options

- `<DataSource name="RMSource" provider="RM">`: The DataSource can be set to any name, however the name must be consistent throughout the config.xml file.
- `<Param name="user" value="sync_userid" />`: The RM user account to be used by the synchronization. This user should be a member of the RM Administrators group, and it should have the same name as the ALM user. This may also be a domain user.
- `<Param name="password" value="xxxx" />`: The password for the above user.
- `<Param name="host" value="RM_DB_name" />`: The Oracle service name for the Dimensions RM database. This may be in the tnsnames.ora on the RM server.

Specifying Value Mappings

Summary

The Sync Engine refers to any `<ValueMap …>` elements to determine how to translate data between two data sources with different schemas. A value map is *bi-directional*, with one data source defined as "left" and the other as "right" (for example, `leftsource="RM" rightsource="QCSource"`). The `leftsource` attribute is required; the `rightsource` attribute is optional.

Each `<ValueMap …>` element contains one or more `<Map …>` child elements, which map actual values that you want to appear differently in the two databases.

NOTE Use value maps only to map users and to map ALM single-selection fields (not multi-selection fields) to Dimensions RM list attributes that are also single-selection.

Mapping Attribute Types

In some cases, ALM stores attribute information as the same type as Dimensions RM, but in a slightly different format. For example, the Requirement Type is stored in ALM as an integer. When you create a requirement in the ALM user interface, the types are presented to you in an alphabetical list. However, at the API level, ALM must store the

type as the associated integer. Because the Sync Engine uses the ALM API, you must map the textual requirement type name in Dimensions RM to the ALM requirement type integer.

To find the integer IDs for requirement types, review the Requirement Types table in ALM SiteAdmin. Note that these IDs are customizable and may vary across ALM installations.

Example Syntax

The following is an example of a <ValueMap> element to map data from Dimensions RM to ALM. Verify the full range of item types in ALM in order to determine what types of requirements you can map from. Note that *Folder* type requirements cannot be synchronized.

```
<ValueMap name="REQ_STATUS_RM2QC" leftsource="RMSource"
rightsource="QCSource">
   <Map left="Interface" right="101" />
   <Map left="" right="3" />
   <Map left="Security" right="102" />
   <Map left="Usability" right="103" />
   <Map left="Performance" right="105" />
</ValueMap>
```

Mapping a Set of Values to a Larger Set

If you need to map a set of values to another set of values that is larger, you need to handle the situation in which a value from the smaller set is mapped to multiple values in the larger set. To handle this problem, use the `primary` keyword to indicate which of the multiple possible values is the preferred choice:

```
<ValueMap name="UserIDs" leftSource="RM">
   <Map left="einsteina" right="alberteinstein" primary="true"/>
   <Map left="einsteina" right="albert_e"/>
   <Map left="einsteina" right="alberte"/>
</ValueMap>
```

Specifying the RM Category

When creating a requirement in Dimensions RM, you can also specify the category this requirement should be created in. The following methods are available:

- Specifying the RM Category by Full Path
- Specifying the RM Category by ID
- Specifying the RM Category by Attribute Value
- Specifying the RM Category by Value Map
- Specifying the RM Category by Category Name

Specifying the RM Category by Full Path

The full path of the category contains the instance name followed by a forward slash, e.g. RMDEMO/Support. For this example, the <Field> tag would be:
`<Field name="IN_CATEGORY" text="RMDEMO/Support"/>`

Specifying the RM Category by ID

The category ID is a numeric value. The root category (the instance name) has the value "0". For this example, the <Field> tag would be:
`<Field name="IN_CATEGORY" text="0"/>`

Specifying the RM Category by Attribute Value

When using the category from a ALM field, the value of that field must either be the full path, the ID or the category name. If the ALM field has the name "CATEGORY", the <Field> tag would be:
`<Field name="IN_CATEGORY" source="CATEGORY"/>`

Specifying the RM Category by Value Map

The following example specifies a value map which includes full path, category name and ID:

```
<ValueMap name="Category" leftsource="RMSource" rightsource="QCSource">

<Map left="RMDEMO/Support" right="1" />

<Map left="Maintainability" right="2" />

<Map left="3" right="3" />

</ValueMap>
```

Specifying the RM Category by Category Name

CAUTION! Using the category name alone is **not recommended**, as the name must be unique. If another category with the same name is created in a different path, the name is no longer unique, e.g. `MY_INSTANCE/Development/Data` and `MY_INSTANCE/Management/Data`.

If you want to use a **unique** category name, the <Field> tag would be:
`<Field name="IN_CATEGORY" text="Support"/>`

Specifying Events and Actions

Summary

The main body of the config.xml file stores a number of event clauses, defined in <event> elements. The event clauses include two portions:

- Event trigger specifications, stored in <trigger> elements, that determine the events that will trigger specific actions. Every event has a primary event type:
  - included (valid for Dimensions RM and ALM)
  - modified (valid for Dimensions RM and ALM)
  - deleted (valid for Dimensions RM and ALM)
  - excluded (valid for Dimensions RM only)
- Action specifications that define the action to carry out when the trigger event is detected.

This section covers important concepts about event triggers and actions. For additional reference material on supported element names, attributes, and values, see Appendix A, "Events, Triggers, and Actions" on page 53.

IMPORTANT! Do not change an event name once you have started to perform synchronizations. Event names are used as primary keys into the registry. If you change the name of an event, the synchronizations may not work properly.

Event names must be unique within the configuration file.

Converting Date Formats

Dimensions RM and ALM may use different date formats. If this is the case, a date conversion must take place before the date can be stored. For a date conversion, the following prerequisites must be met:

1 The DataSource must contain these entries:

```
<Param name="jvm" value="C:\\Program Files\\Micro Focus\\Dimensions
12.11.1\\Common Tools 1.8.6.0\\jre\\11.0\\bin\\client\\jvm.dll" />
<!-- Replace with the path in your environment. -->
<Param name="classpath" value="-Djava.class.path=C:\\Program
Files\\Micro Focus\\Dimensions 12.11.1\\RM\\bin\\SyncEngUtilsJava.ja
r;." /> <!-- Replace with the path in your environment. -->
<Param name="class" value="com/serena/SyncUtils" />
```

2 The field for which a date is required, must be configured as in this example (all on one line):

```
<Field name="QC_DATE_FIELD" source="Time Created"
text="@DATE: Source:dd-MMM-yyyy@HH:mm:ss Target:MM/dd/yyyy" />
```

NOTES

- **@DATE:**, **Source:**, and **Target:** are required.
- **@DATE:** Must be at the beginning of the **text** attribute and defines that a date conversion shall be executed.
- **Source:**
  Defines the source format (the date format of the attribute providing the date).
- **Target:**
  Defines the target format (the date format of the attribute receiving the date).

The following date format patterns are supported:

Pattern	Description	Example
d	Single-digit day (if possible)	▪ 1 ▪ 12
dd	Two-digit day	▪ 01 ▪ 12
E	Two-letter name of day	Mo
EEEE	Name of day	Monday
M	Single-digit month (if possible)	▪ 01 ▪ 12
MM	Two digit month	▪ 01 ▪ 12
MMM	Three-letter name of month	Jan
MMMM	Name of month	January
yy	Two-digit year	18
yyyy	Four-digit year	2018
H	Single-digit hour (if possible) in 24-hour format	▪ 1 ▪ 13
HH	Two-digit hour in 24-hour format	▪ 01 ▪ 13
h	Single-digit hour (if possible) in 12-hour format	▪ 1 ▪ 12
hh	Two-digit hour in 12-hour format	▪ 01 ▪ 12
a	AM/PM designator	▪ AM ▪ PM

Pattern	Description	Example
m	Single-digit minutes (if possible)	■ 1 ■ 59
mm	Two-digit minutes	■ 01 ■ 59
s	Single-digit seconds (if possible)	■ 1 ■ 59
ss	Two-digit seconds	■ 01 ■ 59

Dimensions RM Standard Date Conversion Table

The following table shows the conversion from Dimensions RM standard date formats to Sync Engine's date format.

Dimensions RM	Sync Engine
DD MONTH, RRRR	dd MMMM, yyyy
DD/MM/RRRR	dd/MM/yyyy
DD-MON-RRRR	dd-MMM-yyyy
DD-MON-RRRR@HH24:MI:SS	dd-MMM-yyyy@HH:mm:ss
MM/DD/RRRR	MM/dd/yyyy

Sample Event and Action Syntax

The following example synchronizes requirements and tests. The names of the Dimensions RM entities here (`TRS` and `System_Test_Case`) are specific to this example; they will vary from system to system.

IMPORTANT! The RQ_TYPE_ID must be the last field mapped in the `<Create>` action clause.

```
<Event name="RMCREATEQCREQ" datasource="RMSource"
description="RMCREATEQCREQ Create Event">
   <Trigger>
      <Condition>
         <Param event="included"/>
         <Param project="RMPROJ" />
         <Param name="collection" value="Interface"/>
         <Param name="class" value="TRS"/>
      </Condition>
   </Trigger>

   <Create name="CRRQ"  datasource="QCSource" description="CRRQ">
      <Param class="Requirement"/>
      <Param project="QCPROJ" />
```

```
        <Field name="RQ_REQ_NAME" source="TITLE" />
        <Field name="RTM Status" text ="Created" />
        <Field name="RQ_REQ_COMMENT" source="TEXT" />
        <Field name="RQ_DEV_COMMENTS" source="COMMENTS" />
        <Field name="RQ_USER_01" source="COMPLIANCE_STATUS" />
        <Field name="RQ_USER_06" source="IN_SCOPE_OF_TEST" />
        <Field name="RQ_TYPE_ID" source="NFR_TYPE"
        map="REQ_STATUS_RM2QC" />
    </Create>
</Event>
```

The event name RMCREATEQCREQ is the unique name of the event. This can be any name, however each event must have a unique name. Do not use special characters and limit the event names to no longer than 20 characters.

Defining Event Triggers

Summary

The <Trigger …> element contains one or more <Condition …> child elements. These condition blocks identify the cases when the event is triggered. Each condition block contains a set of query-like parameters (<Param …> elements), which are treated as if they were joined by the AND keyword in database query terminology. They must all be true for the condition to be true. The condition blocks themselves (if there are more than one) are treated as if they were joined by the OR keyword to determine whether an event has occurred. If any condition is true, the event has occurred. The parameters for the condition blocks must be the same as the parameters for the data sources.

Trigger Parameters

Triggers contain the following parameters:

- <Param event="included"/>: Specifies the type of event trigger. See Event types below for more information.
- <Param project="RMPROJ"/>: Specifies the Dimensions RM instance.
- <Param name="collection" value="Interface"/>: Specifies the collection in Dimensions RM to check for new requirements.
- <Param name="class" value="TRS"/>: Specifies the class of requirement to check in Dimensions RM.

Event types

The event type is mandatory and should be specified in the first condition block. ALM recognizes the following event types:

Event Type	ALM Description	Dimensions RM Description
included	An object of the specified class has been created since the last synchronization. This may be based on its timestamp.	An object of the specific class has been added to the specified collection since the last synchronization.
modified	An object of the specified class (which has previously been synchronized) has a modification date that is newer than the last synchronization.	An object of the specified class has a modification date that is newer than the last synchronization time.
deleted	An object of the specified class has been deleted since the last synchronization was run. **NOTE:** In ALM, when an item is deleted, its record is deleted from the primary table where it used to reside.	An object of the specified class has been deleted since the last synchronization was run.
excluded	Not applicable	Dimensions RM has the concept of a collection, from which objects can be included or excluded. ALM does not support the excluded event. Please refer to the deleted event.

NOTE The Sync Engine records the date and time for every event it detects in the Windows registry. Every event is stored in its own entry in the Windows registry.

Important Events Guidelines

Keep the following important guidelines in mind when defining events:

- All event names must be unique.
- Do not include special characters in event names
- Always place included events before modified events in the file.
- Always place modified events before excluded events in the file.
- Always place excluded events before deleted events.
- Deleted events must always be last.

Defining Actions

Summary

There are three types of actions: <Create …>, <Update …>, and <Delete …>. The actions specify the data source that is to accept the change triggered by an event in the other data source as well as the fields and parameters necessary to perform the action. The

action block also specifies a name and description for each action, which do not need to be unique, are both optional, and are used for logging purposes.

NOTE If a Dimensions RM object is locked when the Sync Engine tries to update it, the Sync Engine breaks the lock.

Action Parameters

Actions contain the following parameters:

- `<Param class="type_name"/>`: The name of the item type or class to be created, updated or deleted. For example, `<Param class="Requirement"/>`
- `<Param project="project_name"/>`: The name of the ALM project or Dimensions RM instance in which the item will be created. For example, to create, update, or delete an item in a ALM project called QCPROJ: `<Param project="QCPROJ"/>`
- The `<Delete...>` action for ALM elements use a special parameter:

  ```
  <Param name="Permanent" value="no"/>
  ```

  With this parameter, it is possible to decide if the ALM element will be deleted permanently (value = "yes") and cannot be recovered, or if only the mapping is executed and the object still exists (value = "no").

Defining Field Elements

Consider the following important guidelines when defining <Field> elements.

- Within action elements, `<Field …>` elements may have either a literal text value (specified with the `text` attribute), or a variable value (specified by the `source` attribute). They can also combine the two. If a field parameter has only a `text` attribute, it will be interpreted as a literal value for the field. For example, the following <field> element identifies the field based on the display name `RTM Status`, with the value `Created`:
  `<Field name="RTM Status" text ="Created"/>`
- When mapping Dimensions RM attributes, use the `source` attribute with the `<field>` element to define the source of the data. You can find these names by right-clicking the requirement class in Dimensions RM Class Definition, then selecting **Define** and double-clicking on each attribute that you want to map. For example, in the following `<field>` element, `TITLE` is the internal Dimensions RM name for the attribute:
  `<Field name="RQ_REQ_NAME" source="TITLE">`

  The Sync Engine replaces the placeholder value with the field value from the event object of the same name as that specified by the `source` attribute. If the event object does not specify the value, the virtual value will be empty.
- In the following example, both the `text` and `source` attributes are present:

  ```
  <Field name="RTM_ISSUE" text="RTM Issue {0}" source="PUID" />
  ```

  The use of the `{0}` token in the text attribute value creates a composite value (for example, `"RTM Issue MRKT_0009"`). The `{0}` token is replaced with the current value of the PUID field. If you use the `{0}` token in a `text` attribute and do not specify a `source` attribute, the token is replaced with nothing.
- In the following example, the mandatory `RQ_TYPE_ID` field in ALM is mapped to a text field in Dimensions RM called `NFR_TYPE`. The `RQ_TYPE_ID` field specifies the numeric

ID of a ALM requirement type. The `map` attribute specifies a value map that translates the Dimensions RM text string (the value of the `NFR_TYPE` field) to the ALM numeric ID.

```
<Field name="RQ_TYPE_ID" source="NFR_TYPE" map="REQ_STATUS_RM2QC"/>
```

Configuring Link Synchronization

You can use `<Param>` elements to define how links between different types of item (*tests* and *requirements* specifically) should be synchronized. This allows you to store, maintain, and synchronize information about item relationships in both Dimensions RM and ALM.

Summary of Link Events

Links are synchronized as part of the following events, granted that link synchronization is configured correctly:

- A *modified* event is triggered when a link is added between objects. When synchronizing from Dimensions RM to ALM, an update action then updates the objects with the link. The update action also removes links.
- An *included* event is triggered when a link is created at the time of an object being created. When synchronizing from Dimensions RM to ALM, a create action then creates the link.

Defining Link Synchronization

To configure this, you must complete the following steps:

1 To synchronize relationship information from ALM to Dimensions RM, add a `<Param>` element to the `<Create>` and `<Update>` elements with the `reqlink` or `tstlink` attribute set to the name of the relationship in Dimensions RM. Define the `reqlink` attribute to link requirements to tests or to defects. Define the `tstlink` attribute to link tests to defect.

For example, to ensure that the *to_be_tested* relationship is updated in Dimensions RM when you synchronize from ALM to Dimensions RM, include the `<Param>` line as in the example below:

```
<Update name="update" datasource="QC" description="Replace RM object
    data with ALM data">
   <Param reqlink="to_be_tested"/>
   <Field name="Description" text="Object was changed in RM" />
   <Field name="Name" source="Title" />
   <Field name="RTM State" source="State" />
</Update>
```

2 To synchronize relationship information from Dimensions RM to ALM, you must add `<Param>` elements to the `<Condition>` element, and to the `<Create>` and `<Update>` actions. You must define these as follows:

a In the `<Condition>` element for a trigger, define the `<Param>` tag as follows:
`<Param name="[reqlink|tstlink]" value="relationship"/>`

For example, if you want to synchronize the information on requirements links from a relationship type called *impact*:
`<Param name="reqlink" value="impact"/>`

b In the `<Create>` and `<Update>` actions, define the `<Param>` tag as follows:
`<Param [reqlink|tstlink]="relationship"/>`

For example, if you want to synchronize the information on requirements links from a relationship called *to_be_tested*:
<Param reqlink="to_be_tested"/>

Synchronizing Categories to Folders

You can synchronize Dimensions RM categories to folder structures in ALM. To synchronize categories to folders, you must add a field called QC_FOLDER to the create or update events.

To synchronize Dimensions RM categories to ALM folders:

1. Add the following <Field> element to the create or update action:
<Field name="QC_Folder" text="path"/>

2. To define a specific location in ALM where a requirement should be created or updated, set the text attribute to the full path. All requirements will be placed under a root folder called REQUIREMENTS. You do not need to include this in the text attribute.

   For example, to create or update the requirements under the path RM_REQUIREMENTS, set the <Field> element as follows:
<Field name="QC_Folder" text="RM_REQUIREMENTS"/>

3. Optionally, to create or update the requirement in a path based on the category structure in Dimensions RM, include a source attribute and set it to IN_CATEGORY:
<Field name="QC_Folder" source="IN_CATEGORY"/>

4. If you include the source attribute and set it to IN_CATEGORY, you must also include the following additional <Field> element to ensure that the category path is recreated as folders in ALM:
<Field name="QC_Folder_Type" text="1"/>

   Without this additional element, the categories are instead added as requirement type to the items in ALM.

   For example, the following two <field> elements in the create event will create new requirements under the REQUIREMENTS/RM_REQUIREMENTS folder in ALM:
<Field name="QC_Folder" text="RM_REQUIREMENTS"/>
<Field name="QC_Folder_Type" text="1"/>

Synchronizing Test Steps to Design Steps

You can synchronize Dimensions RM's Test Steps to ALM's Design Steps by adding the following <Field> element:
<Field name="TestSteps" source="TEST_STEPS" />

Validating the XML Configuration File

Summary

When you attempt to run the Sync Engine, the Sync Engine will validate the XML configuration file. Validation is the process of discovering whether the XML in the

document is valid; that is, whether it conforms to the schema named in the header section of the file. The process is similar to compiling a program and finding bugs.

TIP The rules for a particular type of sport determine how the sport is played—the duration of a typical contest, how many people can play at a time, and what is a legal way to score points. Similarly, an XML schema defines the structure and content of the XML document, and determines what is legal and what is not.

The Sync Engine reads this line of the XML configuration file:

```
<IntegrationConfiguration xmlns="http://www.serena.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="SyncEngine.xsd">
```

The file named `SyncEngine.xsd` is the XML schema file used for the integration. By default, this file is located in the `<RM_Install_Dir>\conf` directory. The XML parser in the RM-ALM integration will attempt to validate the XML configuration file against the XML schema.

Possible Errors

During the validation process, the XML parser may emit errors that are difficult to understand. Though not an exhaustive list of the error messages, the following explanations may help you understand where the errors are occurring.

`The key for identity constraint of element 'IntegrationConfiguration' is not found`

NOTE For this error, the XML processor for the Sync Engine always gives a line number for the end of the file (that is, for the end tag `</IntegrationConfiguration >`).You should not assume that the error is at the end of the file.

For this error, any of the following could be the cause of the problem:

- The data source referenced in a ProxyDef or action (for example, Create/Update/Delete) is not valid. This refers to the DataSource section at the beginning of the XML configuration file. Check this section to be sure that you are using the correct name for the data source.
- The leftsource/rightsource in a ValueMap is not valid. Check to make sure that the leftsource matches one of the two data sources. If you specified the rightsource, it must match as well.
- The provider referenced in a DataSource is not valid. Check the Provider section to be sure that you are using the correct name for the provider and for the DLL.
- The data source referenced in an Event is not valid. Check the DataSource section at the beginning of the XML configuration file to be sure that you are using the correct name for the data source.

Inspect the configuration file if this error message appears.

`Duplicate key value declared for identity constraint of element 'IntegrationConfiguration'.`

This error generally indicates one of the following:

- Duplicate DataSource names

- Duplicate Event names
- Duplicate names in a Provider tag
- Duplicate names in a ProxyDef tag
- Duplicate names in a ValueMap tag

The error message will give the line number and this should be a good indication of where the error is.

Hints

After you have finished editing the XML configuration file, you can try validating the file before attempting to use the file with the integration.

You may be able to get better information about any XML error by using an XML validation tool. These tools typically have more descriptive error messages and can help you identify the problem.

Sample XML Configuration for Synchronizing Test Cases

Summary

The following example configuration file illustrates how you might synchronize ALM test objects to Dimensions RM for create, update, and delete events. Because tests are secondary objects (children of requirements), a reqlink parameter in defined in the Trigger clause, and another in the Create action clause. These parameters are required in order to enable the creation and deletion of links between test cases and requirements. These attributes are only required for secondary objects such as tests and not for requirements.

NOTES

- The reqlink attributes refer to the same Dimensions RM link name but have a slightly different presentation.
- In this example, the delete event updates the Dimensions RM_System_Test_Case object rather than delete it. The Sync Engine can delete objects, however this is not always desirable from an auditing perspective.

Example

```
<!--_____________________________________________________________________-->
<!-- This section includes events for mapping Test in ALM to
     SYSTEM_TEST_CASE in Dimensions RM -->
<!--_____________________________________________________________________-->

<!-- The Beginning of the CREATE events (TEST to
     SYSTEM_TEST_CASE)section for synchronization from ALM to
     Dimensions RM -->

<Event name="Test_Case_Created_QC2RM" datasource="QCSource"
     description="Test_Case_Created_QC2RM Create Event">

   <Trigger>
     <Condition>
        <Param event="included"/>
        <Param project="QCPROJECT" />
        <Param name="class" value="Test"/>
        <Param name="reqlink" value="TRS_to_Sys_Test_Case"/>
     </Condition>
   </Trigger>

   <Create name="Create_RM_System_Test_Case"  datasource="RMSource"
       description="Creation of a Dimensions RM System_Test_Case from
       an ALM Test Creation">
     <Param class="System_Test_Case"/>
     <Param project="RMPROJ" />
     <Param collection="Interface"/>
     <Param reqlink="TRS_to_Sys_Test_Case"/>
     <Field name="TITLE" source="TS_NAME" />
     <Field name="TEXT" source="TS_DESCRIPTION" />
     <Field name="TYPE" source="TS_USER_03" />
     <Field name="POSITIVE_OR_NEGATIVE" source="TS_USER_05" />
     <Field name="PRE-REQUISITES" source="TS_USER_25" />
     <Field name="QC_TEST_ID" source="TS_TEST_ID" />
     <Field name="QC_TEST_STATUS" source="TS_STATUS" />
     <Field name="QC_EXECUTION_STATUS" source="TS_EXEC_STATUS" />
   </Create>
</Event>

<!--  The Beginning of the UPDATE events (TEST to SYSTEM_TEST_CASE)
     section to synchronize from ALM to Dimensions RM -->

<Event name="Test_Case_Updated_QC2RM" datasource="QCSource"
     description="Test_Case_Updated_QC2RM Update Event">
   <Trigger>
     <Condition>
        <Param event="modified"/>
        <Param project="QCPROJ" />
        <Param name="class" value="Test"/>
        <Param name="reqlink" value="TRS_to_Sys_Test_Case"/>
     </Condition>
   </Trigger>

   <Update name="Update_RM_System_Test_Result" datasource="RMSource"
```

```
        description="Update of a System_Test_Case from an ALM Test
        Update">
      <Param class="System_Test_Case"/>
      <Param project="RMPROJ" />
      <Param collection="Interface"/>
      <Param reqlink="TRS_to_Sys_Test_Case"/>
      <Field name="TITLE" source="TS_NAME" />
      <Field name="TEXT" source="TS_DESCRIPTION" />
      <Field name="TYPE" source="TS_USER_03" />
      <Field name="POSITIVE_OR_NEGATIVE" source="TS_USER_05" />
      <Field name="PRE-REQUISITES" source="TS_USER_25" />
      <Field name="QC_TEST_ID" source="TS_TEST_ID" />
      <Field name="QC_TEST_STATUS" source="TS_STATUS" />
      <Field name="QC_EXECUTION_STATUS" source="TS_EXEC_STATUS" />
   </Update>
</Event>

<!--  The End of the UPDATE events (TEST to SYSTEM_TEST_CASE) section
    synchronizing from ALM to Dimensions RM -->
<!--  The Beginning of the DELETE events (TEST to SYSTEM_TEST_CASE)
    section synchronizing from ALM to Dimensions RM -->

<Event name="DeleteLinks_QC2RM" datasource="QCSource"
    description="DeleteLinks_QC2RM Update Event">
   <Trigger>
     <Condition>
        <Param event="deleted"/>
        <Param name="class" value="Test"/>
        <Param project="QCPROJ" />
        <Param name="reqlink" value="TRS_to_Sys_Test_Case"/>
     </Condition>
   </Trigger>

   <Update name="UpdateDelete_Test_Links_in_RM" datasource="RMSource"
       description="UpdateDelete in RM from QC">
     <Param class="System_Test_Case"/>
     <Param project="RMPROJ" />
     <Param reqlink="TRS_to_Sys_Test_Case"/>
     <Field name="TITLE" source="TS_NAME" />
   </Update>
</Event>

<!--  The End of the DELETE events (TEST to SYSTEM_TEST_CASE) section
    synchronizing from ALM to Dimensions RM -->
```

Chapter 6

Using the Sync Engine

About the Sync Engine Service

The Sync Engine runs as a service called **Micro Focus SyncEngine**. Display Windows services to verify that the service is running, or to display properties for the service. When you display the service properties, you can see the path to the service executable on your system. This path also provides the information you need to locate the Sync Engine configuration XML file (config.xml).

Testing Connections

Before running the Sync Engine, make sure that the server to which the Sync Engine service is installed can connect to both ALM and Dimensions RM.

Testing the ALM Connection

To test the connection to ALM:

1. On the server to which the Sync Engine is installed, open a Web browser and enter the following URL:
   `http://QCservername:port/QCBIN`

   For example, if the server is called qualitycenter and the port number is 8080, open:
   `http://qualitycenter:8080/QCBIN`

   If you have not already downloaded and installed the ALM client components, they will be installed now.
2. If installation stops with a CAPICOM message, you may need to restart the browser and try again.
3. To further confirm that the connection works correctly, consider logging in to ALM and creating a requirements. You can also verify which fields are mandatory in this way.

Testing the Dimensions RM Connection

To test the connection to Dimensions RM:

1. From the Dimensions RM browser client, verify that you can log in using the account that is specified in the config.xml file. See Chapter 5, "Configuring the Sync Engine" on page 23.

Using Synchronization Logs

The Sync Engine provides log files containing information about its current state. It logs information about interaction with the data sources, the number of events processed, and timing information about the last polling cycle. Furthermore, the data source providers can return to the Sync Engine information that needs to be written to the log file.

Setting Logging Options

This file (<*RM_Install_Dir*>\conf\log4cpp.conf by default) controls some of the behavior of the RM-ALM integration:

- The level of detail (DEBUG, INFO, WARN, ERROR)
- The location of the log file
- The name of the log file
- The maximum size of the log file
- The maximum number of log files that will be maintained as a result of "rolling" to additional files after a log file reaches the maximum size limit

The log4cpp.conf file contains five categories:

- fileBrowser
- fileSyncEngine
- fileWebService
- fileDbDoctor
- fileAlfEventEmitter

You can alter the level of detail by replacing the default value (WARN) with one of the other possible values. DEBUG output includes all field values that are assigned or bypassed and why. INFO output includes ERROR output plus WARN output plus any state information, such as the beginning and ending of each event and action. ERROR output includes only conditions that result in one or more actions not being performed (such as server failure). WARN output includes ERROR output plus problems with database or XMLfile configuration, undefined fields, or value truncation.

You can alter the location of the log file by updating the log4cpp.conf file with the new location. You can alter the name of the log file. The layout of the output of the log file is controlled by the log4j.appender.file<*category*>.layout settings, which use standard log4j formatting. You should not change anything else in this file.

Logging Recommendation for Testing

When you are initially testing the synchronization, we recommendsetting the logging level to DEBUG.

Using Individual Logs for Several Instances

When running several Sync Engine instances, it is advised to log into different log files as this simplifies log analysis. This can be achieved by creating a separate log4cpp.conf file for each Sync Engine instance. Assuming you have 2 Sync Engine instances, you might have log4cppSync1.conf and log4cppSync2.conf log configuration files. When running Sync Engine (or creating a Sync Engine service), just specify the F option followed by the configuration file name (e.g. -F log4cppSync1.conf). Sync Engine will then use the configuration of the specified log configuration file.

Controlling the Sync Engine from the Command Line

The command-line usage for the Sync Engine is as follows:

```
syncengine [-f file] [-c] [-e level] [-E file] {-L priority] [-k
    {install|config|uninstall|start|stop|runservice}] [-n serviceName]
    [-v] [-m SAAS] [-h] [-p password] [-P eventName] [-r file]
```

Option	Description
`-c`	Checks (validates) the XML configuration file against the associated schema document. The schema document is specified in the second line of the XML configuration file. After validating the file and logging any errors, the Sync Engine quits without any further processing.
`-e level`	Specifies the error level to show: debug, info, warn, error.
`-E file`	Logs startup errors to a specified file.
`-f file`	Specifies an alternative configuration file.
`-F file`	Specifies an alternative log4net.cpp file. **IMPORTANT!** Do NOT use is option in combination with the `e` or `E` options (which specify log level or log file).
`-h`	Lists the available command-line options.
`-k install`	Installs the Sync Engine service with specified parameters.
`-k config`	Changes the parameters used by the installed Sync Engine service. These settings are permanent, the same as if they were specified with the `install` option.
`-k runservice`	Starts the Sync Engine service using the installed settings. **IMPORTANT!** Do NOT use this command if your intention is to run the Sync Engine as a process that will run once and then terminate.
`-k start`	Starts the Sync Engine service with temporary parameters--for a single run only. Once the service stops, the Sync Engine will return to using the installed settings. If no temporary parameters are specified, the effect is the same as using the `runservice` option. **IMPORTANT!** Do NOT use this command if your intention is to run the Sync Engine as a process that will run once and then terminate, or if you want to permanently change the parameters used by the Sync Engine service.
`-k stop`	Stops the Sync Engine service.
`-k uninstall`	Uninstalls the Sync Engine service
`-L priority`	Sets the Windows system priority for the Sync Engine service. *servicePriority* must be LOW, BELOWNORMAL, NORMAL, ABOVENORMAL, HIGH, or REALTIME. If you do not specify this option, the Sync Engine service runs under the default priority of BELOWNORMAL.
`-m SAAS`	Invokes the sync engine in SAAS (cloud) mode. To run in normal mode, omit this option.

Option	Description
-n *serviceName*	Sets the service name and specifies that the corresponding configuration file will be used.
-v	Displays the version number.
-p *password*	Encrypts the password text for use in the configration file. **NOTE** You must past the output from this command into the XML configuration file within the appropriate <DataSource> element.
-P *eventName*	Purges the queue containing data to be processed of all objects associated with the named event. Use this when orphaned objects (for example, from a disconnection) are beginning to affect performance. You can remove objects associated with multiple events by using the -P option more than once: "-P RTMIncluded -P RTMExcluded". If the event named isnot in the default config.xml file, use the -f option to specify the configuration file where the event is defined.
-r *file*	Logs runtime (operation) errors to a specified file.
-v	Displays the version number.

TIP The Sync Engine starts as a Windows service when you start it with the "-k start" option. To simplify the testing of your integration configuration, run the Sync Engine as a standalone executable by starting the Sync Engine without the "-k start" option.

Verifying Synchronization Results

You can use the log files to verify that Synchronization is functioning correctly, both to verify that synchronization completed successfully, and then to check to see what data has synchronized.

Verifying that Synchronization Completed Successfully

To verify synchronization results:

1. In Windows Explorer, open the directory where the Dimensions RM logs are stored, such as:

   <installation_location>\logs

2. Press the F5 key to refresh. As you refresh, you should see the syncEngine.log file increase in size.

3. Once the syncEngine.log file is no longer increasing in size, the Sync Engine service has likely completed. Display the Windows services, and verify that the service has stopped.

4 Once you have verified that the service has stopped, copy the syncEngine.log file to another location and open it in a text editor.

5 Scroll to the bottom.

6 The final entries should look something like the following:

```
04-22-10 11:32:01.761 INFO   SyncEngine           - ! Successfully
    closed connection to DataSource 'RMSource' (Micro Focus Test
    Sync Proxy).
04-22-10 11:32:01.761 DEBUG  SyncEngine.QCSource  - Start
    QualityCenter::disconnectQC()
04-22-10 11:32:01.793 DEBUG  SyncEngine.QCSource  - End
    QualityCenter::disconnectQC()
04-22-10 11:32:01.808 INFO   SyncEngine           - ! Successfully
    closed connection to DataSource 'QCSource' (Test Director Sync
    Proxy).
```

The "Successfully closed connection" message indicates that the processes has completed successfully. If this does not appear, then the synchronization may have failed due invalid XML, a data exception, or other issues. See "Troubleshooting" on page 51.

Validating Synchronization Results

To verify synchronization results:

1 Once synchronizatino is complete, option the syncEngine.log file in a text editor.

2 Scroll to the bottom of the file, then scroll up until you find the start of the date and time entries for the most recent synchronization. For example, if the date is November 10 2010, search for a line like the following:

```
03-31-10 11:07:52.706 ALERT  SyncEngine          - - The Sync Engine
    (2009.2.0.368) is starting...
```

3 There are a number of message that indicate the success of a synchronization, depending on the actions that the synchronization performed. Examples of these messages include:

- `Successfully retrieved 1 System_Test_Case objects for project`
- `Begin Create Action`
- `Adding a requirement`
- `Posting the requirement to QC: success`
- `Checking in the item: Success`
- `Creating link`
- `End QualityCenter:createLink() - Create Sucess.`

Verifying What Records Have Been Synchronized

A record of which requirements and tests have been synchronized is maintained in the syncengine_qc.sync_xref table in the ALM database. Your database administrator can connect using sqlplus, create a spool file and run a select query to generate a report.

Troubleshooting

General Troubleshooting Checklist

If you encounter errors during synchronization, make sure that all of the following have been completed:

- The syncengine_qc schema has been created, or the ALM project schema has not been granted access to the syncengine_qc schema. See Creating the Interface Schema20.
- The sync trigger has been created in the ALM project schema. See Setting Up the Link Triggers21.
- Users are configured correctly. For example, the ALM user is a project admin, the Dimensions RM user is an administrator, and the names are the same across the two systems.
- The latest updates have been installed to the Dimensions RM server, and you are working with supported versions of ALM and Oracle.
- For best results, ALM is running on a 64-bit Windows server.

Troubleshooting Specific Issues

syncengine Service Fails Immediately with No or Minimal Log

Possible cause Syntax error in the XML configuration file. To troubleshoot, load the config.xml into an XML aware editor that can validate syntax.

syncengine Service Fails After Partial Synchronization; Log Shows Failure During Create Action

Possible cause A data exception. Some of the attribute data in the Dimensions RM requirement may contain values that result in exceptions when translated by the Syncengine API, when trying to create the corresponding record in ALM. If possible, isolate the specific record. To do this, create a collection with just that record and re-run the synchronization and verify that the failure occurs again in the same place. You can attempt to correct the data in the issue, or ensure that this record is not in the collection when you run the complete synchronization and manually re-create it in ALM.

This error may result in requirements being created in ALM with the title of "new requirement" and with no value. These invalid extraneous requirements in turn prevent subsequent synchronizations from working. If you find any of these, check them out and delete them in ALM. If you are unable to check-out or delete them, wait a few minutes for ALM to release locks on them.

The Sync Engine Returns the Following Error on Create Action: (RTM Database Error (1): 1, ORA-00001: unique constraint (FRC_RM.PK_SYNC_XREF) violated)

Possible cause The requirement in question has already have been synchronized to ALM, but has been removed and then re-added to the collection being synchronized. Re-adding the requirement to the collection triggers the syncengine to attempt to create it in ALM. This might happen if, for example, you have moved all of your Dimensions RM requirements

into a new collection and are now attempting to synchronize the new collection. The Sync Engine therefore registers these as new requirements and the error above appears in the syncengine log. When this occurs, clean the create event out of the system registry to remove the failed synchronizations. The next synchronization will run against the new collection as expected.

Synchronization Does Not Create ALM Requirements

Possible cause Confirm that you are using the current release of Dimensions RM with all of the latest patches.

Synchronization Creates ALM Requirements but Does Not Update Them

Possible cause Confirm that you are using the current release of Dimensions RM with all of the latest patches.

Synchronization Runs but Reports Oracle Error: object not found

Possible cause There is an error in the instance name, collection name, or data attribute field names. Verify the names for all Dimensions RM and ALM variables, and make sure that the case matches exactly in all places.

Synchronization does not Replicate New Links from ALM to Dimensions RM

Possible cause The current versions of one (or both) of the RM objects to be linked are in a baseline. You cannot link baselined object versions. You must replace the baselined object versions with new versions that are not in a baseline.

Synchronization does note Delete Links in Dimensions RM

Possible cause The current versions of one (or both) of the RM objects are in a baseline. You cannot remove links between baselined object versions. You must replace the baselined object versions with new versions that are not in a baseline.

Synchronization does not Add or Delete Links Between Tests and Requirements in Dimensions RM

Possible cause Tests in ALM must be checked in in order for the Sync Engine to detect a change to links between a test and a requirement. Ensure that all tests are checked-in before running a synchronization. Also confirm that the Trigger SQL has been executed for the ALM project schema (see "Setting Up the Link Triggers" on page 21).

Appendix A

Events, Triggers, and Actions

Events

- Create
- Update
- Delete

```
<Event name="New defect" datasource="QC" description="Defect detected">
```

The `name` value is arbitrary but must be unique in the XML configuration file.

The `datasource` must be an Micro Focus ALM/Quality Center datasource as defined in the head of the configuration file, in case the event is triggered by ALM/Quality Center.

The `description` should provide enough information to trace the mechanism, but is only used for logging purposes.

Triggers

Each `event` contains a `trigger` that is triggered by a `condition`. This `condition` is specified by a set of parameters, which describe the type of `event` and the conditions that must be met to execute the action specified for this event.

```
<Trigger>
   <Condition>
         <Param event="created" />
         <Param name="class" value="Requirement" />
   </Condition>
</Trigger>
```

event Parameter

Exactly one `event` parameter is required in the first or only condition block for each event. If `event` is specified again in subsequent condition blocks (if they exist), then its value is ignored.

Possible values are:

- `Created`—Defines an event that is triggered by a new object being created in the specified class within Dimensions RM or ALM. In Dimensions RM, this happens when an object is added to a designated collection. The collection that triggers the creation event is configured in the Sync Engine configuration file. In ALM, the created events are triggered by the creation of a new object of the specified type.

- `Updated`—Defines an event where an object that has already been transitioned to Dimensions RM is changed in Dimensions RM or ALM. The event may be filtered by class and attribute conditions. In ALM 10, new versions of items are created in response to changes in corresponding items in Dimensions RM.
- `Deleted`—Defines an event that is triggered by the deletion of an object in Dimensions RM or ALM. The corresponding object will be marked as deleted.

name Parameter

The `name` parameter may be set to `class` or a valid field name.

If the `name` parameter is set to `class`, then the `value` attribute must be the name a ALM class (for example, Requirement, Test or Defect). See Appendix B, "ALM Datatypes" on page 57 for a list of available classes in ALM. The `value` attribute is case sensitive and should appear as it is shown in ALM. If that name contains the name of a field of the ALM class, then the value should contain a field value to constrain the triggering of the event. For example, setting the `name` to `Priority` and the `value` to `High` will trigger the defined action only for the objects with their Priority set to High. Any object with a different priority value will not trigger the action.

Actions

The integration supports three types of actions which may occur when the event triggering conditions are met:

- Create
- Update
- Delete

Create

A Create action creates new objects of a specified class when executed. The specified fields will be filled with either static data or be retrieved from the source object.

```
<Create name="create" datasource="QC" description="Submit new RM
    object">
      <Param class="Requirement" />
      <Field name="Name" source="Title" />
      <Field name="Description" text="Object from RM" />
</Create>
```

```
<Param class=" Requirement" />
```

The `class` parameter defines the target class in either Dimensions RM or ALM. In case the datasource is set to a Dimensions RM type source, this may be any available class in the specified instance for this event. In case of ALM set as the datasource, there is a fixed set of available classes in ALM. See Appendix B, "ALM Datatypes" on page 57 for a detailed description. Only a valid class of type **Requirement**, **Test**, **Test In Testset**, **Test Step**, **Run**, **Testset** or **Defect** can be specified.

```
<Field name="Description" text="Object from RM" />
```

The `name` attribute specifies the target field of the object in the specified class. This can be one of the fixed system attributes (for a detailed view of system attributes of the different class types in ALM refer to Appendix B, "ALM Datatypes" on page 57) or any valid user attribute. A valid user attribute is one that was defined in ALM

The `text` attribute will fill the mapped field with a static value.

```
<Field name="Description" text="Requirement RQ{0} from RM" source="ID"/>
```

The `text` attribute may contain a token `{0}` that will be replaced by the data from the defined source attribute. For example, for a requirement in Dimensions RM with 156 as its ID, the description field of the object in ALM will be set to "Requirement RQ156 from RTM".

```
<Field name="Name" source="Title" />
```

The `source` attribute specifies the field from the source object from which the data is retrieved.

IMPORTANT! For date fields, a conversion may be necessary. For additional information, see chapter "Converting Date Formats" on page 32.

Update

An update action will retrieve the data of a changed source object and synchronize the target object data for the specified fields. An update action may also replace the field value with a specified static value.

```
<Update name="update" datasource="QC" description="Replace RM object
    data with ALM data">
      <Field name="Description" text="Object was changed in RM" />
      <Field name="Name" source="Title" />
      <Field name="RTM State" source="State" />
</Update>
```

The `name` attribute specifies the target field of the object in the specified class. This can be one of the fixed system attributes (for a detailed view of system attributes of the different class types in ALM (refer to Appendix B, "ALM Datatypes" on page 57) or any valid user attribute. A valid user attribute is one that was defined in ALM.

```
<Field name="Description" text="Object was changed in RM" />
```

The `text` attribute will fill the mapped field with a static value for any instantiation of the class type.

```
<Field name="Description" text="Requirement RQ{0} from RM" source="ID"/>
```

The `text` attribute may contain a token `{0}` which will be replaced by the data from the defined source attribute. For example, for a requirement in Dimensions RM with 156 as its ID, the description field of the object in ALM will be set to 'Requirement RQ156 from RTM'.

```
<Field name="Text" source=" DESCRIPTION " />
```

The `source` attribute specifies the field from the source object from which the data is retrieved.

IMPORTANT! For date fields, a conversion may be necessary. For additional information, see chapter "Converting Date Formats" on page 32.

Delete

Depending on the parameter "permanent," the delete event will mark the object as deleted or delete the object permanently and destroy the linkage between the source object and the target object.

```
<Delete name="delete" datasource="QC" description="Delete or mark as
    deleted the RM object">
</Delete>
```

Appendix B

ALM Datatypes

Data types of attributes are noted in parentheses. Any field may be addressed using either the field label or the field name. Entries in red with an asterisk (*) cannot be mapped because they are set by ALM upon creation or updating.

Requirement

- Attachment / RQ_ATTACHMENT (String) *
- Author / RQ_REQ_AUTHOR (User List) – Will be automatically set to the configured user of ALM
- Creation Date / RQ_REQ_DATE (Date) *
- Creation Time / RQ_REQ_TIME (String) *
- Description / RQ_REQ_COMMENT (Memo)
- Direct Cover Status / RQ_REQ_STATUS (Lookup List)
- ITG Request Id / RQ_REQUEST_ID (Number) *
- Modified / RQ_VTS (Date) *
- Name / RQ_REQ_NAME (String)
- Priority / RQ_REQ_PRIORITY (Lookup List)
- Product / RQ_REQ_PRODUCT (Lookup List)
- ReqID / RQ_REQ_ID (Number) *
- Reviewed / RQ_REQ_REVIEWED (Lookup List)
- Type / RQ_REQ_TYPE (Lookup List)
- Arbitrary number of user defined attributes (Number/String/Lookup List/User List/ Date/Memo)

Test

- Creation Date / TS_CREATION_DATE (Date) *
- Description / TS_DESCRIPTION (Memo)
- Designer / TS_RESPONSIBLE (User List) *
- Estimated Dev Time / TS_ESTIMATE_DEVTIME (Number)
- Execution Status / TS_EXEC_STATUS (Lookup List)

- Modified / TS_VTS (Date) *
- Path / TS_PATH (String)
- Status / TS_STATUS (Lookup List)
- Subject / TS_SUBJECT (Lookup List)
- Template / TS_TEMPLATE (String)
- Test Name / TS_NAME (String)
- Type / TS_TYPE (Lookup List)
- Arbitrary number of user defined attributes (Number/String/Lookup List/User List/ Date/Memo)

Testset

- Close Date / CY_CLOSE_DATE (Date)
- Description / CY_COMMENT (Memo)
- ITG Request Id / CY_REQUEST_ID (Number)
- Modified / CY_VTS (Date) *
- Open Date / CY_OPEN_DATE (Date)
- Status / CY_STATUS (Lookup List)
- Test Set / CY_CYCLE (String)
- Test Set Folder / CY_FOLDER_ID (String)
- Arbitrary number of user defined attributes (Number/String/Lookup List/User List/ Date/Memo)

Defects

- Actual Fix Time / BG_ACTUAL_FIX_TIME (Number)
- Assigned to / BG_RESPONSIBLE (User List)
- Closed in Version / BG_CLOSING_VERSION (Lookup List)
- Closing Date / BG_CLOSING_DATE (Date)
- Comments / BG_DEV_COMMENTS (Memo)
- Defect ID / BG_BUG_ID (Number)
- Description / BG_DESCRIPTION (Memo)
- Detected By / BG_DETECTED_BY (User List)
- Detected in Version / BG_DETECTION_VERSION (Lookup List)
- Detected on Date / BG_DETECTION_DATE (Date)

- Estimated Fix Time / BG_ESTIMATED_FIX_TIME (Number)
- ITG Request Id / BG_REQUEST_ID (Number) *
- Modified / BG_VTS (Date) *
- Planned Closing Version / BG_PLANNED_CLOSING_VER (Lookup List)
- Priority / BG_PRIORITY (Lookup List)
- Project / BG_PROJECT (Lookup List)
- Reproducible / BG_REPRODUCIBLE (Lookup List)
- Severity / BG_SEVERITY (Lookup List)
- Status / BG_STATUS (Lookup List)
- Subject / BG_SUBJECT (Lookup List)
- Summary / BG_SUMMARY (String)
- Arbitrary number of user defined attributes (Number/String/Lookup List/User List/ Date/Memo)