



serena.com

SERENA[®] **DIMENSIONS[®] RM 10.1.1**

Integration Guide for
Serena[®] Dimensions[®] CM



Copyright © 2001–2007 Serena Software, Inc. All rights reserved.

This document, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by such license, no part of this publication may be reproduced, photocopied, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Serena. Any reproduction of such software product user documentation, regardless of whether the documentation is reproduced in whole or in part, must be accompanied by this copyright statement in its entirety, without modification.

This document contains proprietary and confidential information, and no reproduction or dissemination of any information contained herein is allowed without the express permission of Serena Software.

The content of this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Serena. Serena assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

Trademarks

Serena, TeamTrack, StarTool, PVCS, Collage, Comparex, Dimensions, RTM, Change Governance, and ChangeMan are registered trademarks of Serena Software, Inc. The Serena logo, Professional, Version Manager, Builder, Meritage, Command Center, Composer, Reviewer, Mariner, and Mover are trademarks of Serena Software, Inc.

All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

U.S. Government Rights

Any Software product acquired by Licensee under this Agreement for or on behalf of the U.S. Government, its agencies and instrumentalities is "commercial software" as defined by the FAR. Use, duplication, and disclosure by the U.S. Government is subject to the restrictions set forth in the license under which the Software was acquired. The manufacturer is Serena Software, Inc., 2755 Campus Drive, San Mateo, CA 94403.

Part number: MA-RMINT-002

Publication date: April 2007

Table of Contents

	Welcome	7
<i>Chapter 1</i>	Overview	9
	What Is the RM-CM Integration?	10
	What Are the Components of the RM-CM Integration?	10
	Who Should Use the RM-CM Integration?	10
	What Do I Have to Install to Use the RM-CM Integration?	11
	Which Machines Have to Install Both RM and CM?	11
	Who Should Set Up the RM-CM Integration?	12
	Do I Need An Additional RM or CM License?	12
	How Often Do RM and CM Exchange Data?	12
	How Does the RM-CM Integration Work?	12
	What If I Want to Use Multiple RM Projects?	13
	What If I Am Also Using Another RM Integration?	13
<i>Chapter 2</i>	Integration Concepts	15
	Introduction	16
	Behavior of the Basic Configuration	16
	Behavior When Data Goes from RM to CM	17
	Use Read-Only Fields in the Receiving Application	18
	Behavior When Data Goes from CM to RM	19
	Behavior of the Advanced Configuration	21
	Behavior When Dimensions Reads Info from RM	24
	Where Proxy Classes and Proxy Instances Are Created	25
	Diagrams Illustrating Behavior of the Advanced Configuration	25
	Comparison of the Basic and Advanced Configurations	26
	Other Considerations	28
	Which Projects Should Be Synchronized?	28
<i>Chapter 3</i>	Setting Up Dimensions CM and Dimensions RM.	29
	Introduction	30
	Setting Up Dimensions CM	30
	Create the Sync User.	30
	Add Report Views	30
	Create C:\temp Directory.	31
	Advanced Configuration: Setting Up a Proxy Class in Dimensions RM.	31
	Next Steps	35
<i>Chapter 4</i>	Configuring the Integration	37
	Introduction	38
	Editing the XML Configuration File: Getting Started	38
	Default Location of the XML Configuration File	38

Character Encoding and Text Editor Considerations	38
Suggested Work Environment.	39
Structure of the XML Configuration File	39
Editing the XML File for the Basic Configuration	39
XML Header	40
<IntegrationConfiguration>	40
General Options	40
Provider Info.	41
DataSource Info	41
Value Maps.	43
ProxyDefs.	44
Events	44
</IntegrationConfiguration>	50
Error Handling for Create, Update, and Delete Actions	50
Error Handling for Create Actions	50
Error Handling for Update Actions	50
Error Handling for Delete Actions	50
Editing the XML File for the Advanced Configuration.	51
Modifying Events to Use Proxies	51
Validating the XML Configuration File	52
Possible Errors	53
Hints	54
The Logging Configuration File	54
What You Can Change	54
Controlling Log File Size	55
Example Logging Configuration File	55
Next Steps	56
Chapter 5 Running the Integration	57
Introduction	58
Things You Should Do Before Running the Integration	58
Creating a Dimensions CM Request from a Dimensions RM Object	58
Creating Test Data	59
Starting the Sync Engine Service	59
Waiting for a Cycle to Complete	59
Checking the Destination Application	59
Checking the Log Files	60
Next Steps	62
Chapter 6 Sync Engine Command Prompt Usage and Special Cases . .	65
Introduction	66
Controlling the Sync Engine from a Command Prompt	66
Configuring the Sync Engine Service to Use a Specific XML Configuration File	67
Reinstalling the Sync Engine Service	67
Running with Multiple Sync Engines	68
Chapter 7 Use Cases	69
Introduction	70

Scoping a Requirement.	70
Example Details	70
Applying a Dimensions Lifecycle to Requirement Approval	71
Example Details	71
Implementing Requirements	71
Defect and Enhancement Management.	71
Example Details	71
Index.	73

Welcome

The *Serena Dimensions RM Integration Guide for Serena Dimensions* describes the process of using the Dimensions RM-Dimensions integration engine to synchronize data between Serena® Dimensions® RM and Serena® Dimensions® CM. The integration can automatically generate new Dimensions requests in response to Dimensions RM events and new Dimensions RM objects in response to Dimensions CM 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.

The audience for this manual is the administrators who will be installing and configuring the integration for Dimensions RM and Dimensions CM users.



NOTE The product name "Serena RTM" changed to "Serena Dimensions RM" in release 10.1. However, code samples in this guide still refer to "RTM." Note also that the term "change document" changed to "request."

For detailed information on Dimensions CM, refer to the manuals available with your Dimensions CM installation, of which the following are only a selection:

- Serena® Dimensions® CM Installation Guide*
- Introduction to Serena® Dimensions® CR*
- Serena® Dimensions® CM Process Modeling User's Guide*
- Serena® Dimensions® CM User's Guide*
- Serena® Dimensions® CM Command-Line Reference Guide*
- Serena® Dimensions® CM Integrated Products Guide*

For detailed information on Dimensions RM, refer to the following manuals available with your Dimensions RM installation:

- Serena® Dimensions® RM User's Guide*
- Serena® Dimensions® RM Administrator's Guide*
- Serena® Dimensions® RM Installation Guide*
- Serena® Dimensions® RM Command Line Parameters Quick Reference*

Serena also offers integrations with other software products. For detailed information, refer to the following manuals, available with your Dimensions RM installation:

- Serena® Dimensions® RM Integration Guide for IBM® Rational® Software Modeler*
- Serena® Dimensions® RM Integration Guide for Mercury Quality Center™*
- Serena® Dimensions® RM Integration Guide for Microsoft® Project*
- Serena® Dimensions® RM Integration Guide for Serena® TeamTrack®*
- Serena® Dimensions® RM Integration Guide for Telelogic DOORS®*

Chapter 1

Overview

What Is the RM-CM Integration?	10
What Are the Components of the RM-CM Integration?	10
Who Should Use the RM-CM Integration?	10
What Do I Have to Install to Use the RM-CM Integration?	11
Which Machines Have to Install Both RM and CM?	11
Who Should Set Up the RM-CM Integration?	12
Do I Need An Additional RM or CM License?	12
How Often Do RM and CM Exchange Data?	12
How Does the RM-CM Integration Work?	12
What If I Want to Use Multiple RM Projects?	13
What If I Am Also Using Another RM Integration?	13

What Is the RM-CM Integration?

The Dimensions RM-to-Dimensions CM integration is a feature of Dimensions RM that enables data to be exchanged between Dimensions RM and Dimensions CM. A user with administrative privileges to both Dimensions RM and Dimensions CM configures the integration to "watch" a specific collection of Dimensions RM objects. When administrator-designated events occur, you can trigger resulting actions. Specifically, you can do any of the following:

- When new objects are added to a specific Dimensions RM collection, trigger the creation of new requests in Dimensions CM.
- When existing objects in a specified Dimensions RM collection are updated, trigger updates in the corresponding requests in Dimensions CM.
- When existing objects in a specified Dimensions RM collection are deleted, change the status of the corresponding Dimensions CM requests (for example, to REJECTED).
- In a specified collection, trigger the creation of new Dimensions RM objects when new requests are created in Dimensions CM
- In a specified collection, trigger updates in existing Dimensions RM objects when the corresponding requests in Dimensions CM are updated.
- In a specified collection, delete Dimensions RM objects when the corresponding requests in Dimensions CM change to a specific status (for example, to REJECTED).

What Are the Components of the RM-CM Integration?

The Dimensions RM-Dimensions CM integration is effected through two main components: a Windows service, and an XML configuration file that determines the behavior of the integration.

The Windows service is known as the Sync Engine, because it synchronizes data between Dimensions RM and Dimensions CM.

The XML configuration file specifies which events the integration watches for and reacts to, and what happens when such an event is detected. This file is named `config.xml` by default, but can be renamed.

There is an additional configuration file that controls the flow of information to a log. This configuration file is named `log4cpp.conf` and should not be renamed.

Who Should Use the RM-CM Integration?

The Dimensions RM-Dimensions CM integration is suitable for all users of Dimensions RM or Dimensions CM who need to coordinate 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 sends the information in those requirements to read-only fields within Dimensions CM requests. 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 Install to Use the RM-CM Integration?

To run the Dimensions RM-Dimensions CM integration, one machine must have both Dimensions RM and the Dimensions CM desktop client installed. The Windows service for the Sync Engine is installed when Dimensions RM is installed, if you chose to install the Sync Engine component.

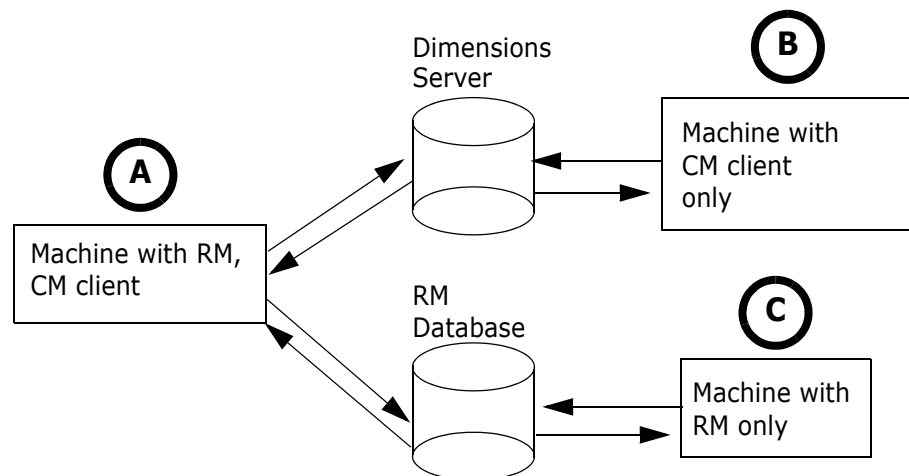
The administrator must set up the integration so that it can access the Dimensions RM and Dimensions CM data. End-users of Dimensions RM or Dimensions CM only need not install anything else.



NOTE The RM-CM integration supports only Dimensions 9.1.2 and later.

Which Machines Have to Install Both RM and CM?

Consider the following diagram.



- User A is on a machine that has installed Dimensions RM and a Dimensions CM client. This user can view objects in Dimensions RM, requests in Dimensions CM, and can use the RM-CM integration to exchange data in both systems.
- User B is on a machine that has installed a Dimensions CM client only. The requests created or edited by User B are visible to User A. Obviously, User B cannot directly change Dimensions RM objects, but if User A has set up the RM-CM integration to create or modify Dimensions RM objects in response to requests that User B creates or modifies, User B's changes will be visible to Dimensions RM users who browse the collections affected by the integration.
- User C is on a machine that has installed Dimensions RM only. The Dimensions RM objects created or edited by User C are visible to User A. In a similar fashion to User B, User C cannot directly change Dimensions CM requests, but if User A has set up the RM-CM integration to create or modify Dimensions CM requests in response to Dimensions RM objects that User C creates or modifies, User C's changes will be visible to Dimensions CM users who browse the requests affected by the integration.

Who Should Set Up the RM-CM Integration?

The RM-CM integration should be set up and administered by a user with administrative privileges. This user must be able to do the following:

- Create new users in Dimensions CM
- Create a new user in Dimensions CM with the role of CHANGE-MANAGER; this user is required by the integration
- Create new classes and relationships in Dimensions RM

Do I Need An Additional RM or CM License?

The machine running the Windows service (the Sync Engine) must have Dimensions RM and a Dimensions CM client installed, so that machine must have licenses for those two applications. No additional licenses are required.

How Often Do RM and CM Exchange Data?

The RM-CM integration can be set to exchange data in intervals as short as one minute, or in much longer intervals of several hours.

Shorter intervals mean that the documents are kept up-to-date. Longer intervals decrease the load on your systems. It is usual to set an interval of a few minutes, but let your own experience guide you.

The data exchange interval is one of the settings that the Dimensions RM or Dimensions CM administrator adjusts specifically for your site. You should not use an interval longer than one day.

How Does the RM-CM Integration Work?

The integration is controlled through a configuration file in XML format. 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 Dimensions CM request that is linked to the Dimensions RM object.

Note that the RM-CM integration does not automatically synchronize all existing requirements. Data is transferred from Dimensions RM to Dimensions CM as new or existing requirements are added to the collection.

There is no graphical user interface, so all behavior changes must be effected through this file. By default, this file is named *config.xml* and it resides in the *conf* subdirectory of the Dimensions RM installation.

The configuration file controls the following settings:

- The accounts to use when the integration connects to Dimensions RM or Dimensions CM
- The mapping of fields from one application to the other
- The definition of what conditions will "trigger" an action in either application

For those not familiar with XML, the thought of modifying the config.xml file might be worrisome. Fortunately, you only need to modify certain sections of the file. This is discussed in detail in [Chapter 4, "Configuring the Integration" on page 37](#).

What If I Want to Use Multiple RM Projects?

If you wish to exchange data between Dimensions CM and multiple Dimensions RM projects, you have to tell the integration which projects you wish to use, and the maximum number of projects that can be open at any given time. This is done during the editing of the configuration file. See [Chapter 4, "Configuring the Integration"](#).

What If I Am Also Using Another RM Integration?

If you are integrating Dimensions RM with TeamTrack as well as Dimensions CM, you must use multiple instances of the Sync Engine.

In this case, you must take these extra steps for each additional integration:

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

Also, if you have to uninstall Dimensions RM, you must first uninstall the other Sync Engine services manually, as they will not be uninstalled with the rest of Dimensions RM.

Chapter 2

Integration Concepts

Introduction	16
Behavior of the Basic Configuration	16
Behavior of the Advanced Configuration	21
Comparison of the Basic and Advanced Configurations	26
Other Considerations	28

Introduction

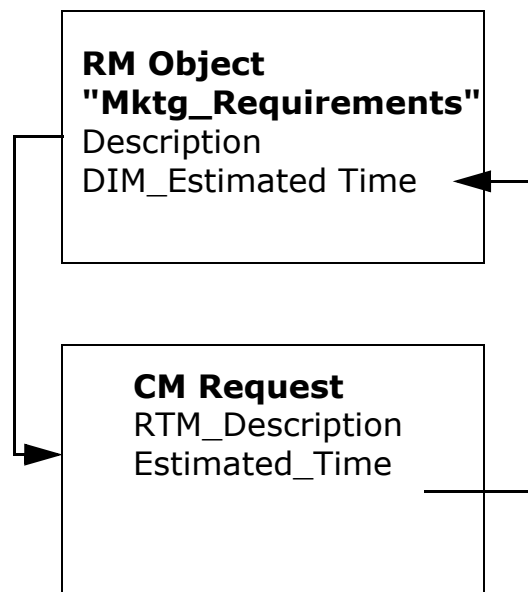
In the [Overview](#) chapter you learned about the basic idea of the RM-CM integration. This chapter covers the behavior of the RM-CM integration, which can be configured in two different ways. This chapter covers both the basic and the advanced ways of configuring the integration.

The basic configuration is suitable when each Dimensions RM object can be associated with a single Dimensions CM request.

The advanced configuration is suitable when a single Dimensions RM object should be associated with several Dimensions CM requests. For example, you might want to associate a single Dimensions RM requirement with Dimensions CM requests addressed to Engineering, Quality Assurance, and Documentation.

Behavior of the Basic Configuration

The simplest way to use the RM-CM integration is to configure it to pass the information in all named fields, one-to-one, from Dimensions RM to Dimensions CM, or vice versa.

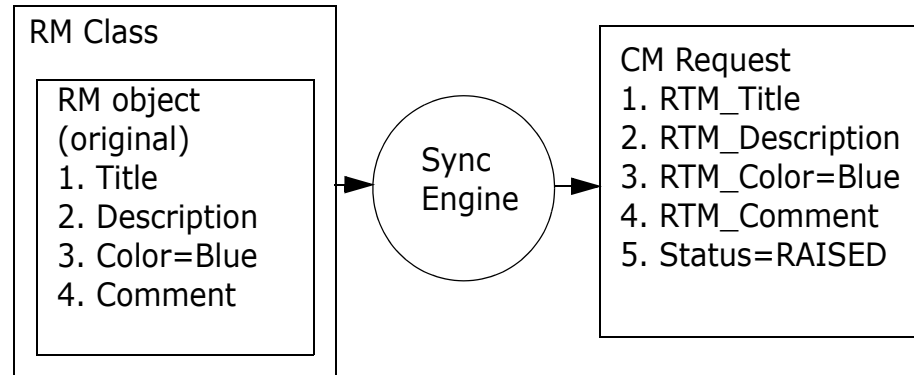


You name the fields whose information is to be exchanged in the configuration file (covered in [Chapter 4, "Configuring the Integration"](#)). The data in those fields is updated each time any of the following events is detected by the Sync Engine:

- Addition of a new Dimensions RM object to a collection; or the creation of a Dimensions CM request in a specified product and design part
- Update of an existing Dimensions RM object or Dimensions CM request
- Removal of an existing Dimensions RM object from the named Dimensions RM collection
- Change in status (for example, to REJECTED) of a Dimensions CM request

Behavior When Data Goes from RM to CM

The following diagram represents a Dimensions RM user creating a new object in Dimensions RM. The RM-DM integration, through the Sync Engine, detects the new Dimensions RM object, and causes Dimensions CM to create a Dimensions CM request containing the same information.

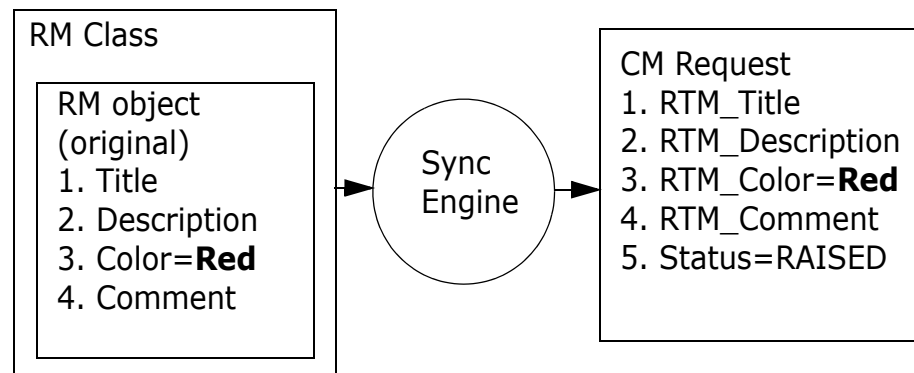


You can see that the information in the Dimensions RM object is passed to fields in the Dimensions CM request. The information in the Color field in Dimensions RM, for example, is passed to a field in the Dimensions CM request called RTM_Color. From now on, the Dimensions RM object and the Dimensions CM request are linked.

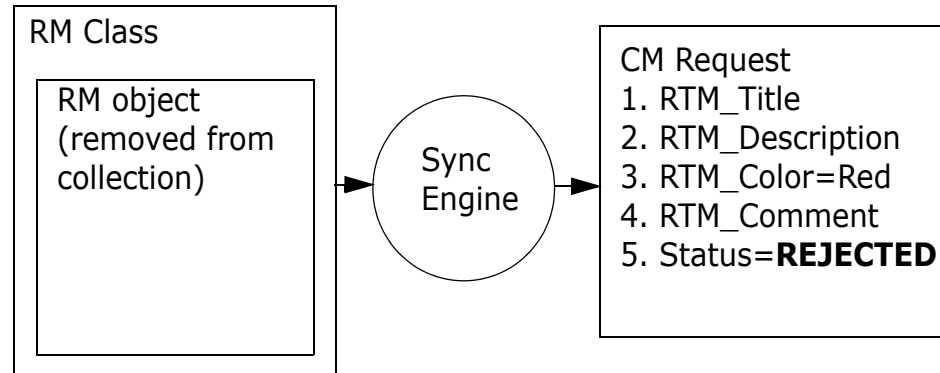


NOTE You should treat information sent to the other application as read-only. There is the possibility of data loss if the destination application, either Dimensions RM or Dimensions CM, writes into the fields that receive the information from the Sync Engine. This is discussed more fully in ["Use Read-Only Fields in the Receiving Application" on page 18](#).

Now a Dimensions RM user changes information in the Dimensions RM object. When the integration runs again, the change is detected, and the linked Dimensions CM request is updated:



Finally, the RTM object is removed from the collection. When the integration runs again, the change is detected, and the linked Dimensions CM request is given a status of **REJECTED**:



You could of course change the status to a different value. The XML configuration file allows you to specify which status the integration should set.

Use Read-Only Fields in the Receiving Application

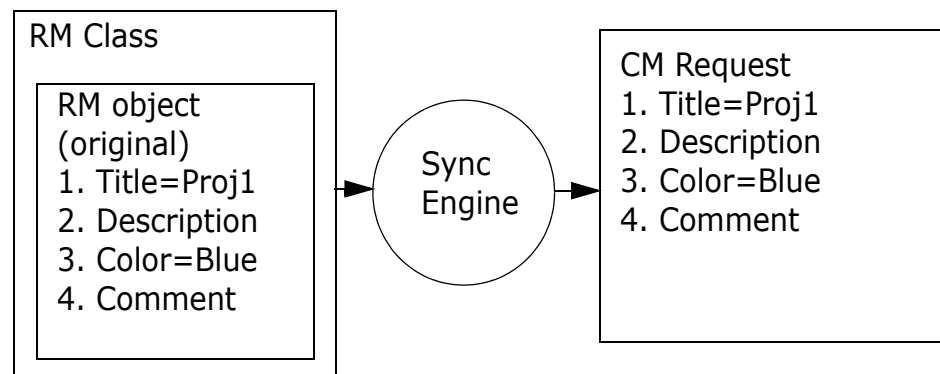
In the previous example, information in the Dimensions RM object was copied to the Dimensions CM request, and new fields using an "RTM_" prefix were created to contain the information from the Dimensions RM object.

You specify these fields in Dimensions CM, and in the XML configuration file. When you create these fields, you must create them as read-only, so that users cannot modify them.

You might ask, why not allow users to modify the Dimensions CM fields, so that the changes could be sent back to Dimensions RM and reviewed there? This is not recommended, and could lead to data loss.

Consider the following example:

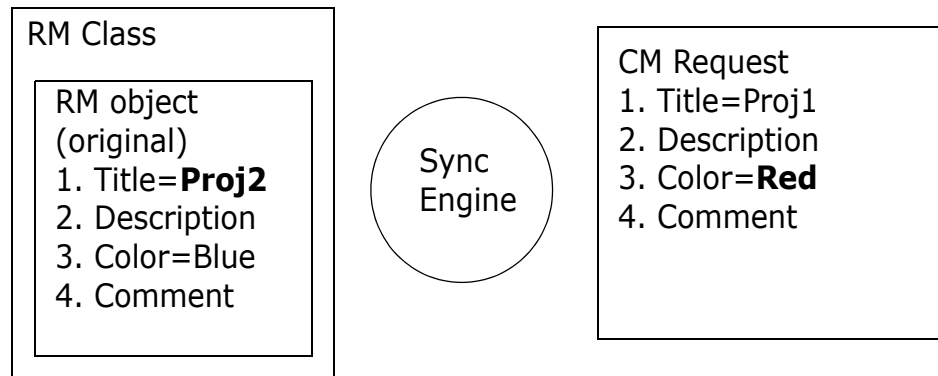
This configuration is **NOT** recommended.



The information from Dimensions RM is sent to Dimensions CM as described before. Dimensions CM users are given permission to change the title, description, color, and comment fields.

But if a Dimensions RM user changes the Dimensions RM object at the same time that a Dimensions CM user changes the Dimensions CM request, the Sync Engine is not able to merge the results:

The Sync Engine is not able to merge changes from both users.



In the above example, what you probably want is for the title to change to "Proj2" and the color to change to "Red", but what you get is either the Dimensions RM user's changes OR the Dimensions CM user's changes.

The integration is unable to merge changes from multiple users whether the originating application is Dimensions RM or Dimensions CM.



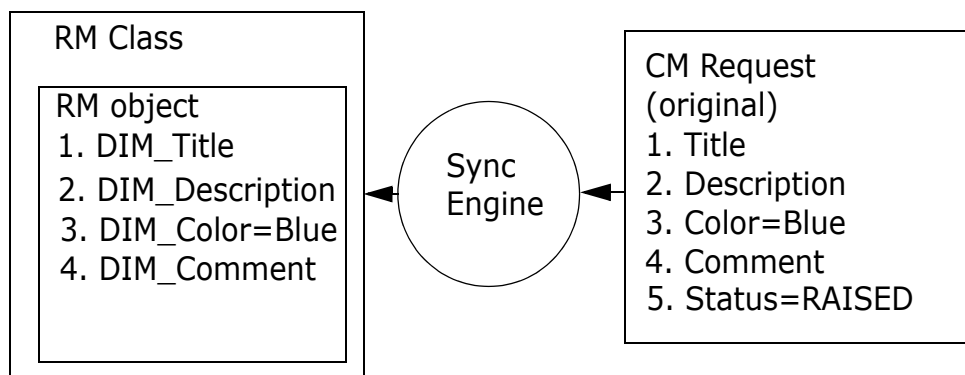
IMPORTANT! To summarize: Because of the possibility of data loss, use separate fields (for example, with an RTM_ or Dimensions_ prefix) to receive the data from the application sending the information, and make those fields read-only.

Behavior When Data Goes from CM to RM

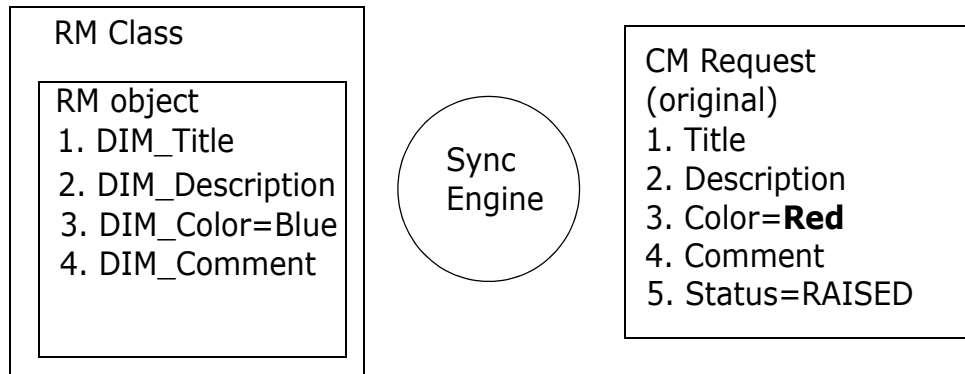
The behavior of the RM-CM integration is essentially the same whether data is sent from Dimensions CM to Dimensions RM, or in the reverse direction.

In the following illustration, a Dimensions CM user creates a new request. The RM-CM integration, through the Sync Engine, detects the new request, and causes Dimensions RM to create a new Dimensions RM object containing the same information:

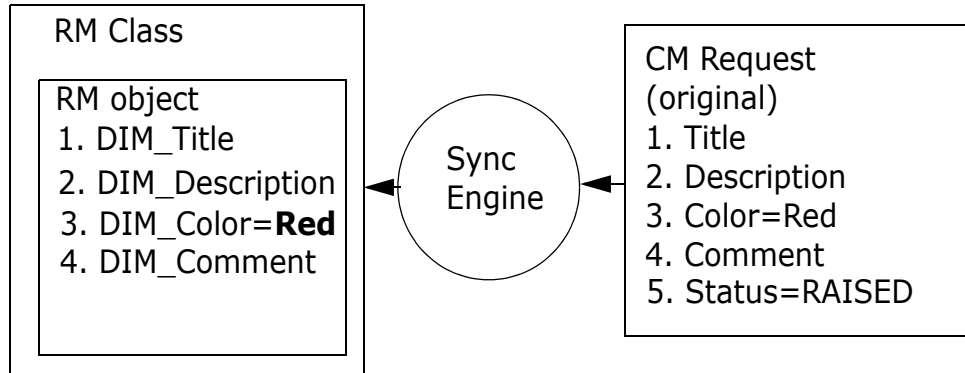
The Sync Engine is not able to merge changes from both users.



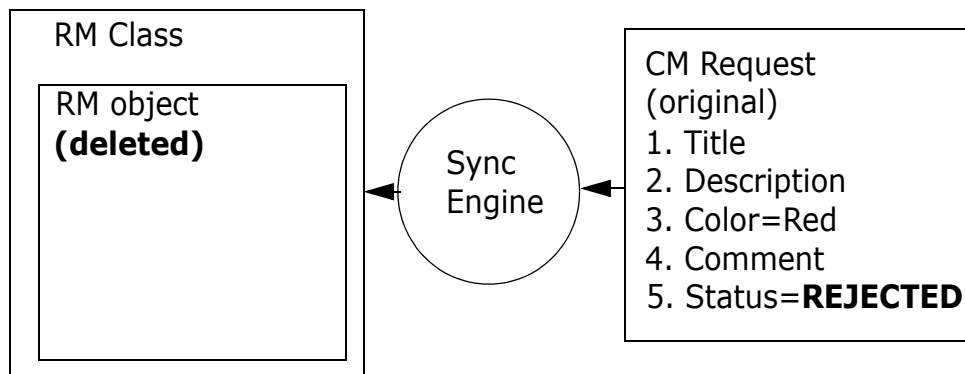
In the next step, a Dimensions CM user modifies one of the fields in the request so that Color=Red instead of Blue. Because the Sync Engine is not running at the moment, there are no changes to the object in Dimensions RM.



Now the Sync Engine runs again, and notices that the Dimensions CM request has been modified. The Sync Engine updates the linked Dimensions RM object:



Finally, if a Dimensions user changes the status of the original request to REJECTED, the Sync Engine detects that change, and deletes the Dimensions RM object (though it remains a member of the associated collection):

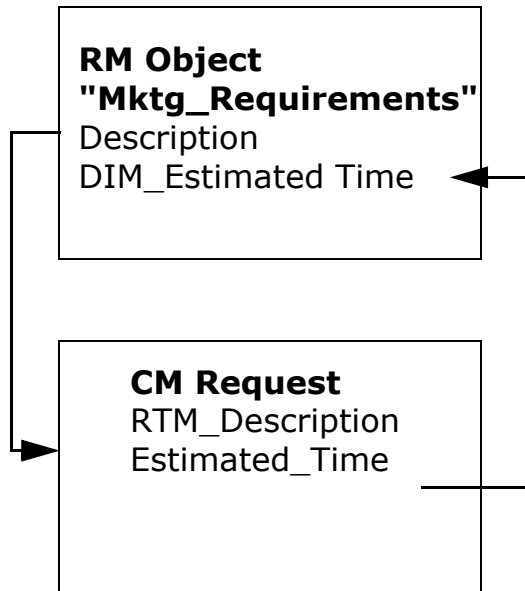


This is the essential RM-CM integration behavior and you can accomplish much of what the product was intended for using only the basic configuration. However, there are certain situations in which the advanced configuration is useful or even necessary.

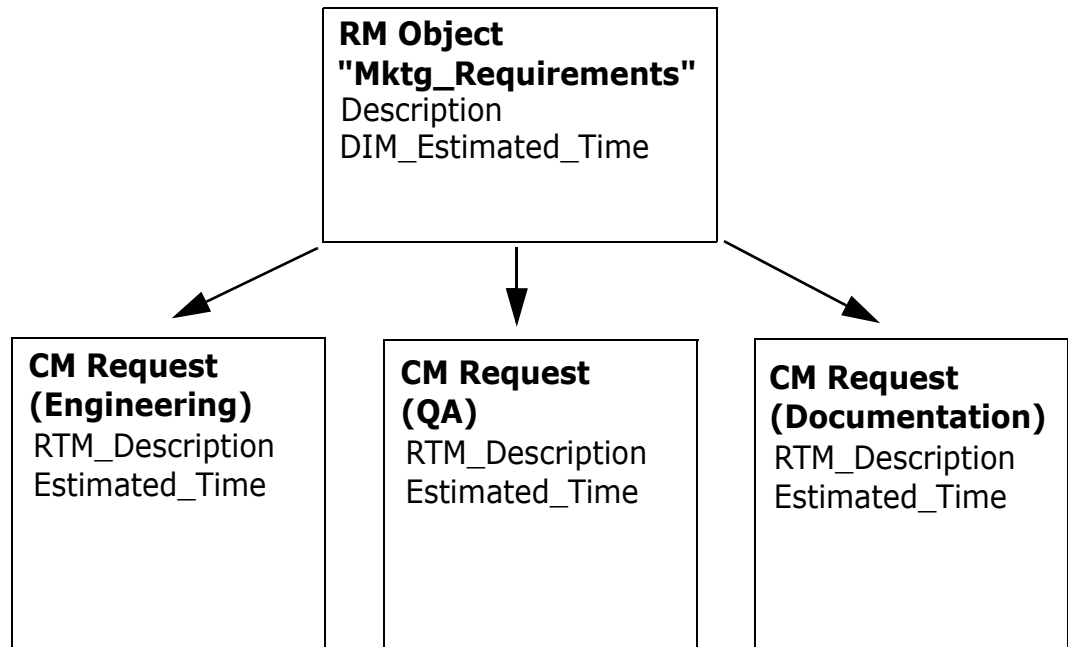
Behavior of the Advanced Configuration

The advanced RM-CM integration uses proxy classes to enable a one-to-many mapping of values. This is useful when you have multiple change documents created from a single Dimensions RM object.

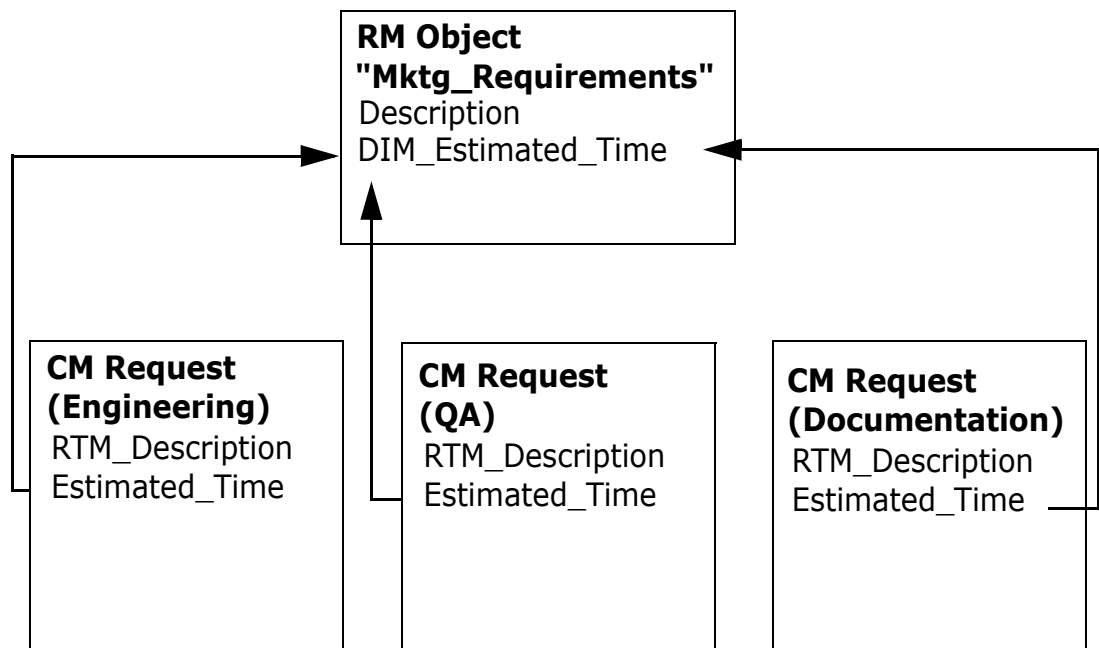
As you saw in ["Behavior of the Basic Configuration" on page 16](#), the RM-CM integration allows you to map Dimensions RM objects to Dimensions CM requests in a one-to-one configuration:



But let's say that a given set of requirements cause the creation of three different sets of Dimensions CM requests—one for Engineering, one for quality assurance (QA), and a third for Documentation.



A problem arises when Engineering, QA, and Documentation each try to send back information—the updates all go to the same Dimensions RM object.

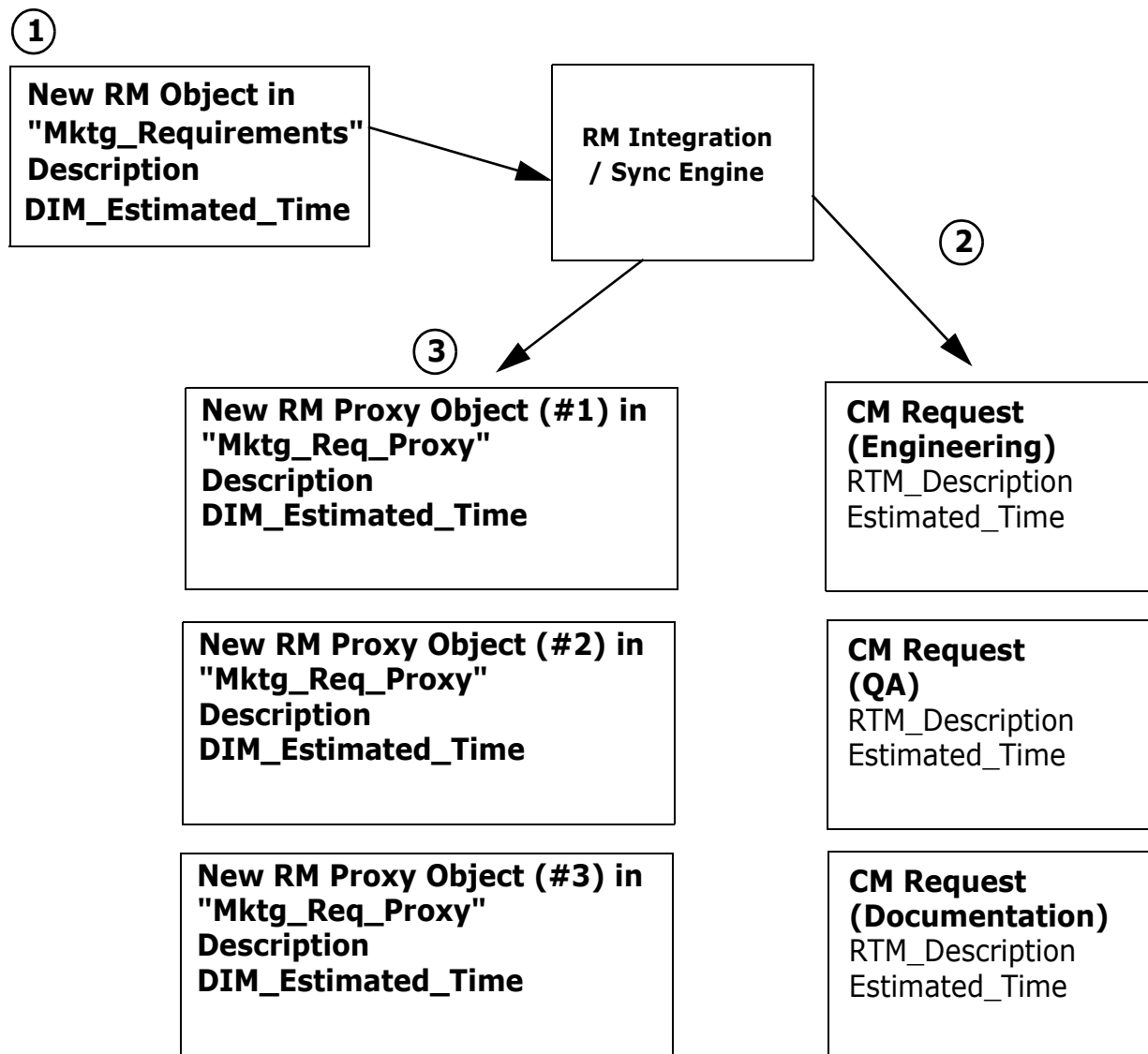


Specifically, each department sends back updates for Estimated_Time. In this scenario, the DIM_Estimated_Time retains the value of the most recent update. The earlier updates are lost.

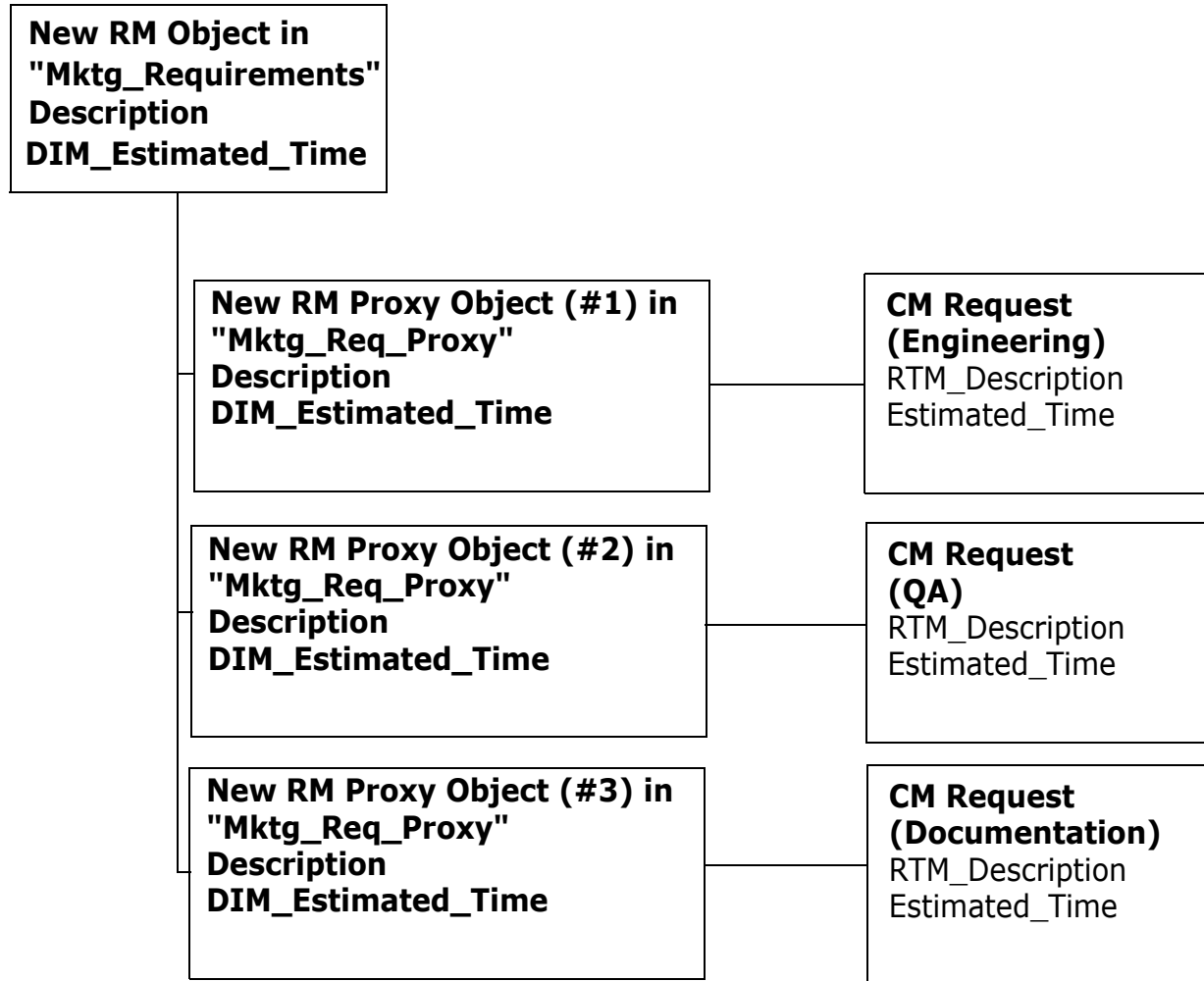
To handle this situation, you define a proxy class, and a link to the original class, in the Dimensions RM schema editor (also known as the Class Definition tool).



Creating this structure—the two classes, and relationship between the two—enables the RM-CM integration to handle the one-to-many situation. When a Dimensions RM object is created (#1 below), the integration creates new Dimensions CM requests (#2), and also creates a new Dimensions RM proxy object for each request (#3):



Finally, the RM-CM integration links the original Dimensions RM object to the newly-created proxy objects, and links the newly-created Dimensions CM requests to the proxy objects:



Because each proxy object is linked to the original Dimensions RM object, a user of Dimensions RM can view the original Dimensions RM object and the linked proxy objects.

Using a proxy gives you more capabilities than not using a proxy. When you use a proxy, you have more control over what happens when the information is transferred. You can, for example, set up the proxy so that additional fields are copied from the source application to the destination application.

Behavior When Dimensions Reads Info from RM

When Dimensions CM needs to read information from Dimensions RM, the integration loads attribute information from the original Dimensions RM object first. Only if the attributes are not found in the original object will the integration load data from the proxy object.



NOTE You should treat the proxy objects as read-only and modify only the original Dimensions RM object.

Where Proxy Classes and Proxy Instances Are Created

You create proxy classes in the Class Definition tool, which can be launched from the RM Manage application of Dimensions RM.

The RM-CM integration creates proxy instances based on the proxy class you created (and on the relationship that you create between the original class and the proxy class).

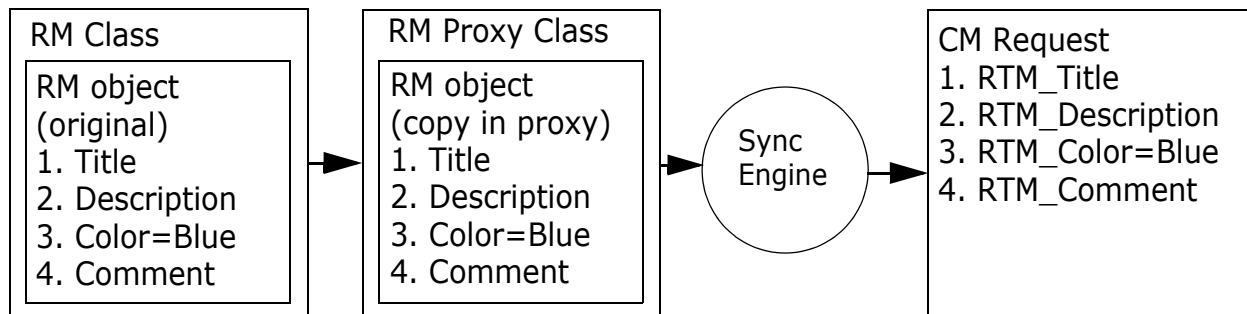
XML configuration file examples for proxy usage and proxyless usage are shown in [Chapter 4, "Configuring the Integration"](#).

Diagrams Illustrating Behavior of the Advanced Configuration

The following text and diagrams illustrate the behavior of the advanced RM-CM configuration—that is, using proxy classes.

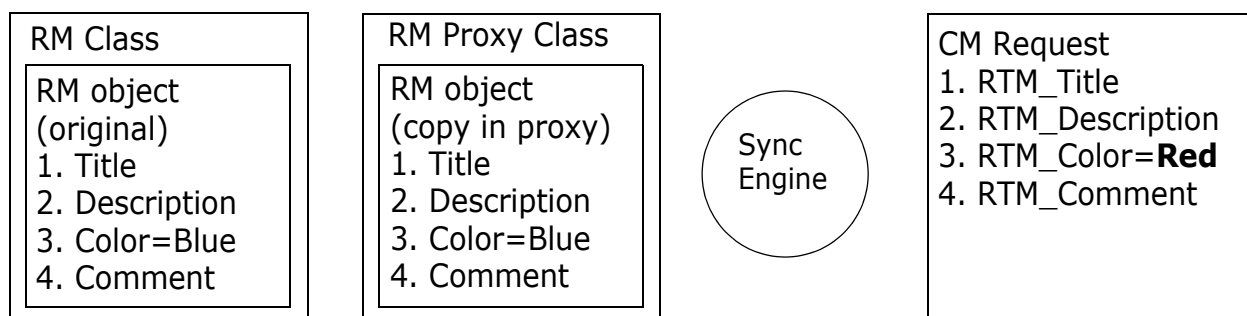
Assume that a Dimensions RM proxy class has been created for the synchronization. This proxy class is linked to the original class in Dimensions RM. Both classes contain an object representing a list of requirements.

The RM-CM integration, through the Sync Engine, causes the creation of a Dimensions CM request:

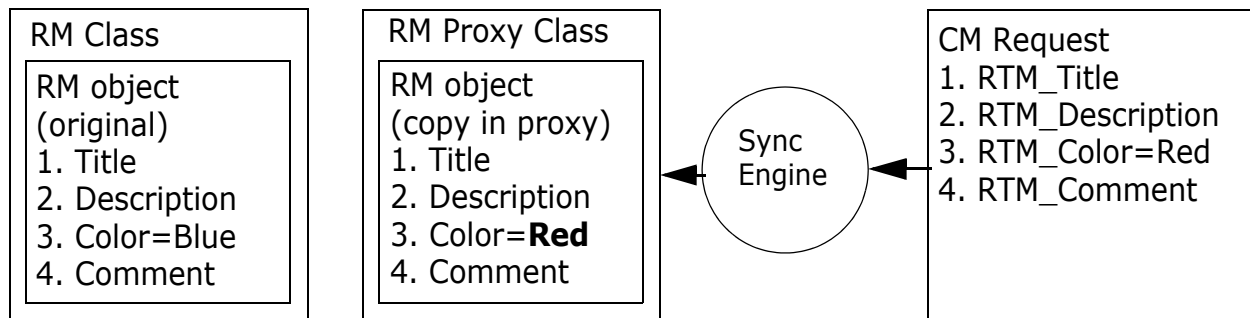


This Dimensions CM request is linked to the Dimensions RM proxy object.

In the next step, a Dimensions CM user modifies one of the fields in the request so that Color=Red instead of Blue. Because the Sync Engine is not running at the moment, there are no changes to the object in the Dimensions RM proxy class or to the original object.

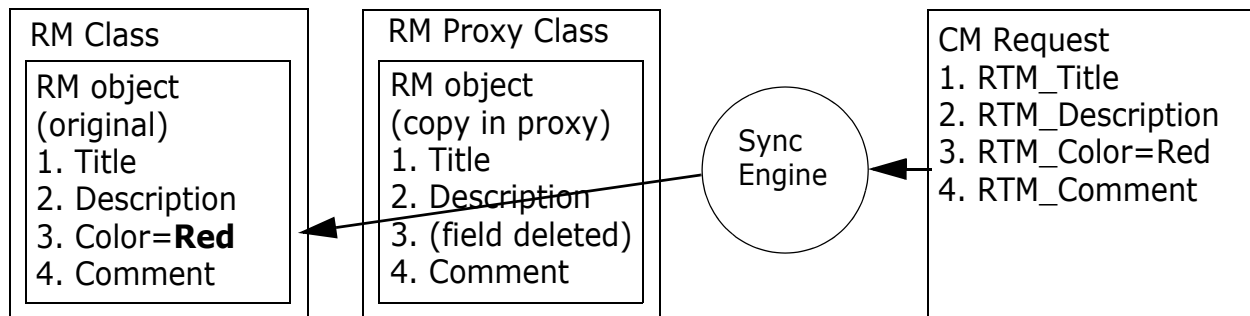


Now the Sync Engine runs again, and detects that the Dimensions CM request has been modified. Because this request was created by the Sync Engine from a Dimensions RM object, it is still linked to the Dimensions RM object in the proxy. The result is shown in the next diagram:



Notice that the original Dimensions RM object is NOT updated. The Sync Engine updates fields in the proxy object. If all fields are found in the proxy object, the update ends there. The original object is unchanged.

In the next diagram, note that one field in the original class has been omitted in the proxy class. In a proxy class, you can decide which fields are included in the integration. However, if the field to be updated is not found in the proxy object, then the Sync Engine modifies the field in the original object:



Comparison of the Basic and Advanced Configurations

The following scenario and diagrams show how data is transferred from Dimensions RM to Dimensions CM if you use a proxy class, and how that behavior is changed if you do not use a proxy class.

In the marketing department of a cell phone company, the marketing representative creates a list of requirements. One of the requirements is "Cell phone must have removable faceplates in different colors, including color of Customer X's logo".

There are several ways to capture the progress from this one marketing requirement to one or more Dimensions CM requests:

Proxy Usage	Fields Captured in Proxy	What Happens
Proxy is used	Title, Description, Color	Changes to Color recorded in the Color field of the proxy object; if Color field were present in original, but not in proxy, changes would be recorded in original (root) requirement object.
Proxy is used	Title, Description	Changes to Color recorded in the Color field of the original requirement object; however, if more than one object maps back to the original requirement object (for example, if the original requirement object causes two requests to be created), the most-recent change is recorded. That means that if Engineering Group 1 says the Color should be Red, but Engineering Group 2 updates five minutes later saying the Color should be Blue, the Color is set to Blue.
Proxy is used, but not for additional "Overall Status" field	Title, Description, Color	Changes to Color recorded in the proxy objects. Overall Status field can only be changed by a human; Marketing representative must examine fields in proxy objects and use that information to determine what value can be set for Overall Status.
No proxy used	(No proxy)	All changes to fields are recorded 1:1 in the original requirement object.

If method #1 is used, any changes to the Color field are recorded in the proxy object and NOT in the original requirement. The marketing representative might look at the original requirement in Dimensions RM, see that the Color field = Blue, and think everything is fine. Meanwhile, Engineering might report that the only plastic that is light enough and strong enough to meet the other requirements for removable faceplates is Red.

Because the proxy object is storing the changes to the Color field, the original requirement object is never changed.

A better way to set up the project is to use an additional status field called Overall Status, that the Marketing Person changes in Dimensions RM after examining the individual status of all other Dimensions RM proxy objects.

Other Considerations

The RM-CM integration is not designed to synchronize all Dimensions RM data with all Dimensions CM data. Only Dimensions RM data in a specific collection is synchronized with Dimensions CM. You need to decide which projects should be synchronized.

Which Projects Should Be Synchronized?

You should consult with the project managers and determine which projects have a need for the RM-CM integration. These projects typically require close cooperation between the people creating the first draft of the requirements (such as Marketing) and the people who implement the requirements (such as Engineering). This is especially true if these departments are in different locations or time zones.

Projects that are small in scope or whose requirements are likely to generate little discussion (a minor update to an existing product) might not need to be synchronized.

Chapter 3

Setting Up Dimensions CM and Dimensions RM

Introduction	30
Setting Up Dimensions CM	30
Advanced Configuration: Setting Up a Proxy Class in Dimensions RM	31
Next Steps	35

Introduction

In the [Overview](#) chapter you learned about the basic idea of the RM-CM integration. This chapter covers setting up the RM-CM integration, up to the point of editing the configuration files. Editing the configuration files is covered in the next chapter.

Setting Up Dimensions CM

The machine running the RM-CM integration must have both Dimensions RM and Dimensions CM installed. After you have installed Dimensions CM, there are two administration tasks that you must complete.

Create the Sync User

In the Dimensions CM Administration Console, you must create a special user in Dimensions CM with the role of CHANGE-MANAGER. This user, which shall be referred to in this document as the Sync User, is used by the integration to make changes to Dimensions CM requests and attributes.



IMPORTANT! It is essential that this user not be an existing Dimensions CM user ID. The Sync Engine is designed to ignore changes created by the Sync User, so that it is not caught in an endless loop of changes created by the Sync User.

Add Report Views

You must also add some report views to Dimensions CM, using the following steps:

- 1 On the Dimensions CM server machine, open a command prompt.
- 2 Issue the following command:

```
Dmdba system/password@databaseInstance
```

where *password* is your system password, and *databaseInstance* is the name of your database instance. The DMDBA system prompt appears:

```
C:\Documents and Settings\Administrator>dmdba system/manager@rtm1
Dimensions DMDBA 9.1.0 (Production) at 15:25:07 Tuesday 21 June 2005
Copyright (c) 1998-2004 Merant International Limited. All rights
reserved.
```

```
Connected to system@rtm1 (oracle)
```

```
SYSTEM>
```

- 3 Enter `INSV databaseName`, where *databaseName* is the name of your Dimensions CM database (for example, "intermediate"). The system displays a message that the views are being installed:

```
SYSTEM> INSV intermediate
Installing report views in intermediate@rtm1, please wait....
```

- 4 When the report views have finished installing, type Exit to leave DMDBA.

Create C:\temp Directory

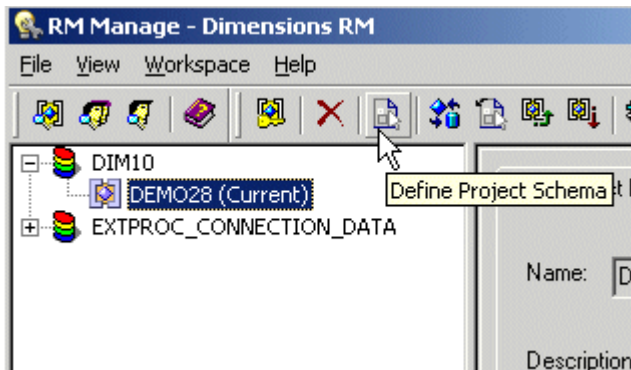
When you add a requirement to a collection in Dimensions RM, a new request is created in Dimensions CM. Information about the requirement is written to a Description file in Dimensions CM. This file is placed in the C:\temp directory. If this directory does not exist, you must create it. Otherwise, the information from Dimensions RM cannot be transferred to Dimensions CM.

Advanced Configuration: Setting Up a Proxy Class in Dimensions RM



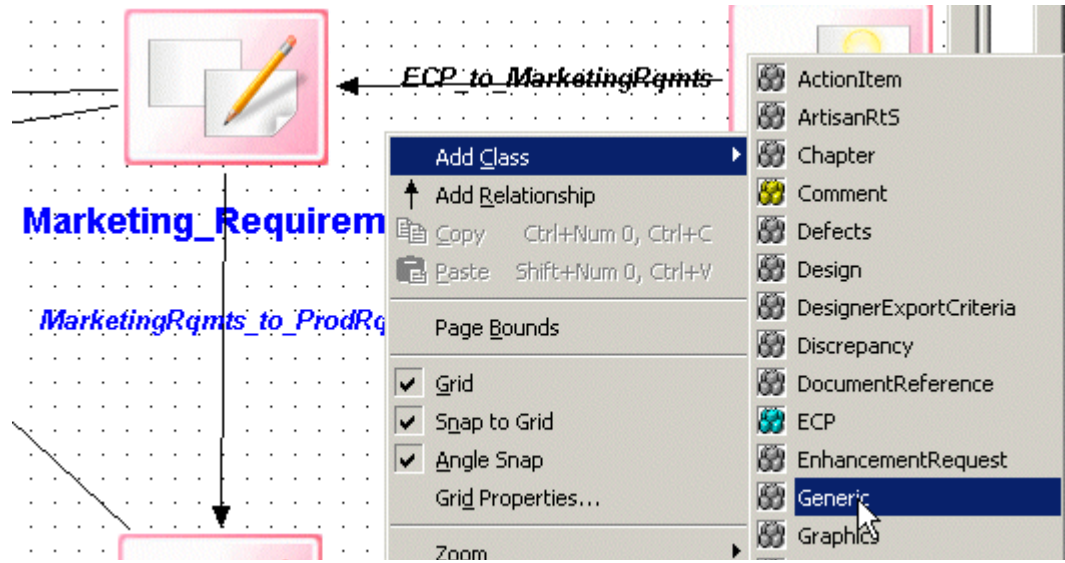
NOTE If you choose not to use proxy classes, you can skip this section. See ["Comparison of the Basic and Advanced Configurations" on page 26](#) for a discussion of proxy configurations as compared to proxyless configurations.

- 1 Determine what values the proxy class should contain, and how many proxy classes you require.
- 2 Invoke RM Manage and select the project you wish to work on. In this example, the project is called DEMO28.
- 3 Click the Define Project Schema icon to open the project schema editor.



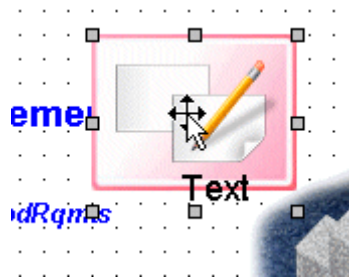
- 4 In the project schema editor, locate the object for which you wish to have a proxy class. (In this example, the Marketing_Requirements object.)

- 5 Right-click and select Add Class > Generic.



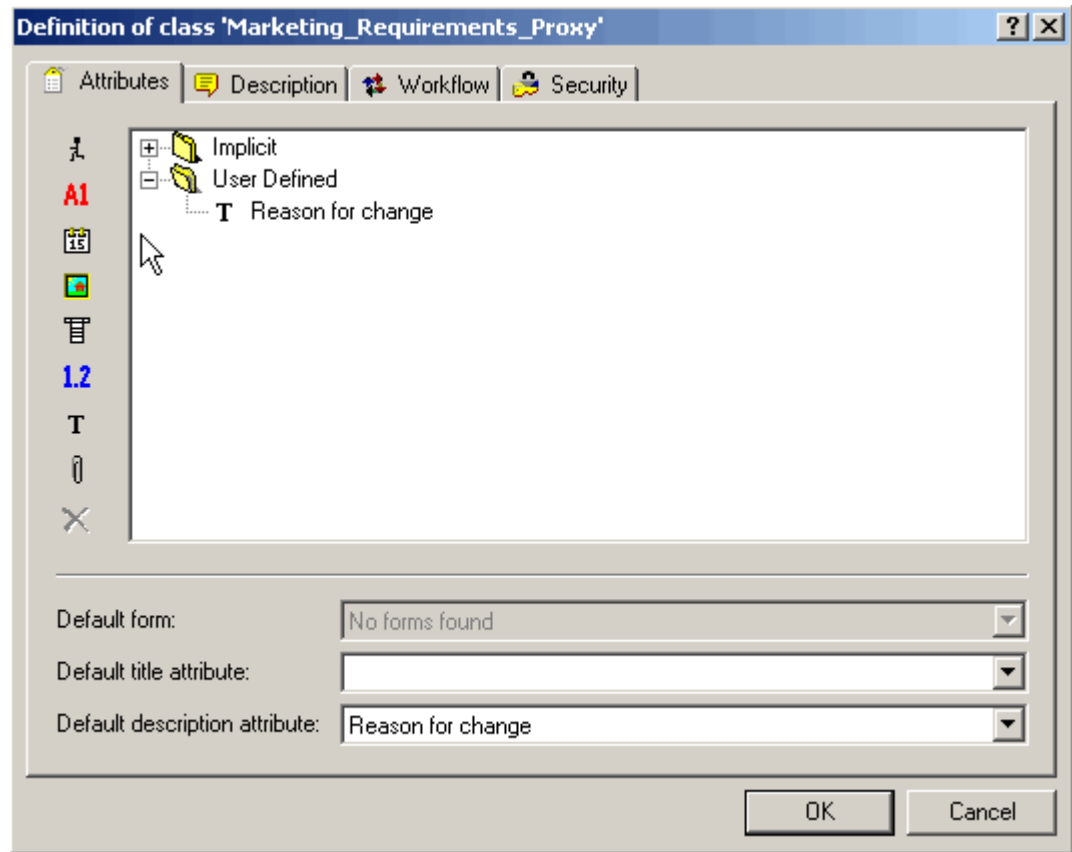
The cursor changes to indicate that you can create a class object.

- 6 Click to place the new class object.

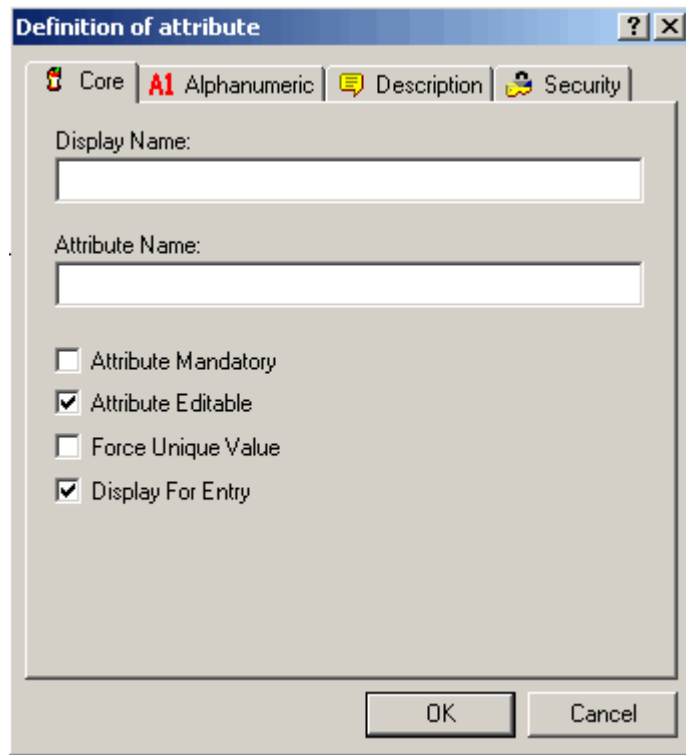


- 7 Name the class. This name is important because you will add an entry to the XML configuration file later for the proxy class, and it must match the name you give the class here. In this example, the proxy class is named Marketing_Requirements_Proxy.

- 8 Now you must define the proxy class. Double-click the proxy class object, or select Define from the context menu. The class definition dialog box appears.



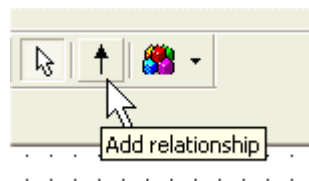
- 9 Click on one of the attribute icons at the left of the dialog box to add a new attribute. The Definition of attribute dialog box appears.



The 'Definition of attribute' dialog box has a title bar with a question mark and a close button. It features four tabs: 'Core' (selected), 'A1 Alphanumeric', 'Description', and 'Security'. Below the tabs are two text input fields: 'Display Name:' and 'Attribute Name:'. Under these fields are four checkboxes: 'Attribute Mandatory' (unchecked), 'Attribute Editable' (checked), 'Force Unique Value' (unchecked), and 'Display For Entry' (checked). At the bottom right are 'OK' and 'Cancel' buttons.

You should add an attribute for each value in the original class that you wish to track. For most values, you can use an alphanumeric attribute.

- 10 When defining the attributes, you must define a Display Name and an Attribute Name. Be aware that the RM-CM integration tries to match the Attribute Name, not the Display Name.
- 11 Uncheck the Attribute Editable checkbox, so that end-users are not tempted to modify the proxy class in Dimensions RM.
- 12 Click **OK** to accept the attribute definition. If you get an error indicating that the name for the attribute must be unique, check the attribute names of the Implicit attributes. There may be a name collision with an implicit attribute. For example, the implicit attribute whose display name is Current Status has an attribute name of STATUS.
- 13 Now click the **Add relationship** button from the toolbar.



- 14 Select the primary class and define a relationship from that class to the proxy class. An arrow appears with an editable text label.
- 15 Enter a name for the relationship. As with the class name, you will have to add this name to the configuration file later, so make a note of it.

- 16** Save the entire schema. If you see an error indicating that other Dimensions RM tools must be closed before you can save the schema, close down other tools such as RM Concept and try the save again.
- 17** Repeat steps 3 - 16 for each proxy class you require.



NOTE The RM-CM integration does not support the use of proxies when mapping Dimensions CM data to Dimensions RM. Dimensions CM does not have the concept of a proxy class, so use a proxyless XML configuration file in that instance.

Next Steps

The steps described in this chapter are essential to getting the RM-CM integration to work, but you also need to edit two configuration files:

- The XML configuration file controls almost all behavior of the integration. This file controls how often the integration runs, what events it looks for, and what it does when those events are detected.
- The logging configuration file controls the level of detail recorded in the log, and where the log is located.

You must edit each of these files whether you use the basic or the advanced configuration. You will learn how to configure these files in the next chapter.

Chapter 4

Configuring the Integration

Introduction	38
Editing the XML Configuration File: Getting Started	38
Editing the XML File for the Basic Configuration	39
Error Handling for Create, Update, and Delete Actions	50
Editing the XML File for the Advanced Configuration	51
Validating the XML Configuration File	52
The Logging Configuration File	54
Next Steps	56

Introduction

In previous chapters you learned about the RM-CM integration, and about how you need to decide how your data is going to be transferred from Dimensions RM to Dimensions CM and from Dimensions CM to Dimensions RM. In this chapter you learn how to change the behavior of the integration.

You have three ways of changing the behavior of the RM-CM integration:

- Changing the command-line parameters used to invoke the Sync Engine affects which configuration file the integration uses. This is covered in [Chapter 5, "Running the Integration" on page 57](#).
- Editing the contents of the XML configuration file (`config.xml`) determines which events the integration watches for, what the integration does when it detects those events, and how often the integration runs. This is the primary method of changing the way the RM-CM integration behaves.
 - You can set up the integration for basic functionality; that is, not using a proxy.
 - You can set up the integration for advanced functionality, using proxy classes.
- Editing the contents of the logging configuration file (`log4cpp.conf`) affects where the main log file is stored, and how much detail the error log contains.

Editing the XML Configuration File: Getting Started

Even if you are unfamiliar with XML, you should be able to modify the sample XML configuration file. The rest of this chapter gives advice on how to do this.

Default Location of the XML Configuration File

By default, the XML configuration file is located at:

`RM_install_directory\conf\config.xml`



NOTE Make a backup before editing the file.

If you need the XML configuration file to be located elsewhere, remember to configure the Sync Engine service to use the new location:

```
syncengine -k config -f file_location
```

If you cannot get the Sync Engine to invoke properly, try moving to the `RM\bin` directory, or add that directory to your `PATH` environment variable.

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. Microsoft® Notepad can save with UTF-8 encoding, but does not do so by default. 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.



IMPORTANT! The XML configuration file is case-sensitive—for example, "provider" is different from "Provider". All terms and names used in the file must take case sensitivity into account.

Suggested Work Environment

As you begin working with the XML configuration file, you will likely be in an edit-test-revise cycle initially. For that reason, you should work in an environment that provides the following:

- Access to the Services dialog box, so that you can verify that the Sync Engine service is running, and stop and start the service if necessary
- Access to the Dimension RM logs directory; this is the location where the log files from the Sync Engine startup and operations can be viewed
- An editing window for your copy of the XML configuration file
- Backup copies of past versions of the XML configuration file (for example, you might wish to save one version that uses proxies and another version that does not)

Structure of the XML Configuration File

The XML Configuration File has the following structure:

```
<XML Header w/version info and encoding>
<Descriptive comment>
<IntegrationConfiguration...>
  <General Options>
  <Provider info>
  <DataSource info>
  <ValueMaps>
  <ProxyDefs (optional; used with proxies)>
  <Events>
</IntegrationConfiguration>
```

The remainder of this chapter describes what you need to change in this file.

Editing the XML File for the Basic Configuration

This section describes how to edit the XML configuration file for the basic RM-CM integration functionality.

Essentially you need to edit all of the sections described in ["Structure of the XML Configuration File,"](#) except the ProxyDefs section.

XML Header

This section consists of the following lines:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- RTM / Dimensions integration configuration file -->
```

You do not need to change anything in this section.

<IntegrationConfiguration>

This tag marks the start of the definition of the integration configuration:

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

Everything else in the file, aside from the XML header, must be between this tag and its closing tag, </IntegrationConfiguration>.

You do not need to modify this tag.

General Options

This section defines the times for how long the integration waits between synchronization runs, and how long it waits between events.

```
<!-- General options
      interval    minutes between the cycle triggering.
      sleep      number of seconds between events -->
<Sync interval="2" sleep="15" />
```

The value for interval is the amount of time, in minutes, that the Sync Engine waits after completing all of the events in the config file before starting over at the first event and repeating the cycle. This value affects how frequently data is exchanged between Dimensions RM and Dimensions CM.

The value for sleep is the amount of time, in seconds, that the Sync Engine waits between each event in the XML configuration file. This value can affect system performance.

You can alter these values as needed for testing or for final execution. As noted earlier, you must not set a interval value longer than one day. A value of one or more minutes is typical.

Provider Info

This section defines the name of the provider DLLs used by the integration.

```
<!-- Provider information
      name          provides the name of the provider DLL used by the
DataSource tag
      location      location of the provider DLL relative to the
SyncEngine
      description   used within the log file -->
<Provider name="RTM" location="syncRTM.dll" description="Serena RTM
Provider DLL" />
<Provider name="DIM" location="SyncDIM.dll" description="Serena
Dimensions Provider DLL" />
```

You do not need to modify anything in this section.

DataSource Info

This section defines the data source information for the integration.



IMPORTANT! There must be exactly two data source entries—one for Dimensions RM (previously known as RTM), and one for Dimensions CM.

```
<!-- DataSource information
      name          name of the data source. Used as a reference by the
SyncEngine.
      provider      name of the provider to use for the DataSource -->

<DataSource name="RTM" provider="RTM">
  <Param name="user" value="Your_RTM_User_ID" />
  <Param name="password" value="Your_RTM_Password" />
  <Param name="host" value="OracleDB_in_tnsnames.ora_file" />
  <Param name="project" value="Your_RTM_Project" />
  <Param name="max_connections" value="4" />
</DataSource>

<DataSource name="DIM" provider="DIM">
  <Param name="user" value="UserID_of_DimensionsSynchUser" />
  <Param name="password" value="Password_of_DimensionsSynchUser" />
  <Param name="dbuser" value="Intermediate_or_other_Base_DB_Name" />
  <Param name="dbpassword" value="DB_Password_Is_Often_Blank" />
  <Param name="server" value="Your_Dimensions_Server" />
  <Param name="database" value="DB_Connection" />
</DataSource>
```

You must replace parameter values (Param values) such as "Your_RTM_Project" with the information for your specific site. See the next section for details.

Values for Parameters

- For the RTM data source:
 - The parameter for user should be your Dimensions RM user ID.
 - The parameter for password should be your Dimensions RM password.
 - The parameter for host should be the entry name in the *tnsnames.ora* file.

For example, the *tnsnames.ora* file begins with a section like this:

```
# TNSNAMES.ORA Network Configuration File:
D:\Oracle\network\admin\tnsnames.ora
# Generated by Oracle configuration tools.
```

The file contains entries such as the following:

```
RTM.SERENA.COM =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = PC1001)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = RTM)
    )
  )
```

The name to use for the host parameter is the entry name, which in this example is "RTM.SERENA.COM". You do not want to use the value labeled HOST.



NOTE If you are unfamiliar with or do not have access to the *tnsnames.ora* file, ask your Oracle administrator for the information detailed here.

- The parameter for project should be the name of your Dimensions RM project. This value is the default project. It is possible to specify projects at the Condition level or at the Action level, but the integration uses the project specified in the DataSource area by default.
- The parameter for *max_connections* indicates the maximum number of Dimensions RM projects that can be kept open while the Sync Engine executes. This value is used when multiple Dimensions RM projects are referred to in the same configuration file.

If the number of projects in a configuration file exceeds the *max_connections* value, the integration closes the least recently used project in favor of the next project needed.

This does not affect the number of Dimensions RM projects that can be referred to in a configuration file, only the number that may be kept open at one time.

If *max_connections* is not specified, the default value is 5. A value of -1 indicates that there is no limit.

- For the Dimensions CM data source:
 - The parameter for user is the user ID of the Dimensions CM Sync User.
 - The parameter for password is the Dimensions CM Sync User's password.

- The parameter for dbuser should be set to the base database name; for example, "intermediate".
- The parameter for dbpassword is the base database password, typically an empty string ("").
- The parameter for server is the name of your Dimensions CM server.
- The parameter for database is what Dimensions CM calls the DB connection.

Value Maps

This optional section allows you to define which values in Dimensions RM are equivalent to specific Dimensions CM values, and vice versa.

```
<!-- ValueMaps
      name          name of the value map.  Used as a reference by the
      SyncEngine. -->
<ValueMap name="SEVERITY" leftsource="RTM">
  <Map left="urgent" right="1_critical" />
  <Map left="high" right="2_severe" />
  <Map left="medium" right="3_moderate" />
  <Map left="low" right="4_minor" />
</ValueMap>

<ValueMap name="Users" leftsource="RTM">
  <Map left="ICDEMOAdmin" right="dmsynch" />
</ValueMap>
```

In this example, the values for the SEVERITY field in the leftsource (RTM) are being mapped to specific values in Dimensions CM. For example, a Dimensions RM value of "urgent" would be mapped to a value of "1_critical" in Dimensions CM.

You must specify the leftsource. Since there are only two data sources, the integration assumes that the remaining data source should be used for the rightsource. You can optionally define the rightsource, or omit it, as you prefer.

Field Mapping Rules

If you map a date field, the formats in Dimensions RM and Dimensions CM must match.

If you map Dimensions CM selection fields to Dimensions RM list attributes, the values must match exactly or be mapped in a ValueMap.

Mapping n Values to m Values, Where $n < m$

If you need to map a set of values to another set of values that is larger, you will face a problem. The problem is that a value from the small set can be 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="RTM">
  <Map left="einsteina" right="alberteinstein" primary="true"/>
  <Map left="einsteina" right="albert_e"/>
  <Map left="einsteina" right="alberte"/>
</ValueMap>
```



NOTE For attributes that use Boolean values (true or false), use lower-case only.

ProxyDefs

This section is only required if you are using the advanced features of the RM-CM integration. See ["Editing the XML File for the Advanced Configuration" on page 51](#).

Events

This section defines the behavior of the integration when events occur in either Dimensions RM or Dimensions CM. An event can be thought of as one or more actions plus a trigger that defines when the actions will be executed. You must modify this section to get the integration to do what you want.

Following is a pseudo-code example of an event. This example is for the purpose of illustrating the structure of an event. An example showing actual code occurs later in this section.

```
<Event name="Name_of_Event" datasource="Name_of_Datasource"
      description="Description_of_Event">
<!--The first line of the Event contains the name of the event, the name
      of the data source, and a description of the event. -->
<!--Within the Event tag, there must be at least one Trigger tag and at
      least one action. An action is a Create tag, an Update tag, or a
      Delete tag (it is not named Action). -->

  <Trigger>
    <!--A Trigger tag contains one or more Condition tags. -->

    <Condition>
      <!--A Condition tag describes the conditions that trigger the
            action associated with the event. -->
    </Condition> <!--End Condition tag-->

  </Trigger> <!--End Trigger tag-->

  <Create>
    <!--This is the action to be performed when the Condition or
            conditions described in the Trigger tag are met. The action in
            this pseudo-code example is Create, but also could be Update or
            Delete. -->

  </Create> <!--End of action -->
```

```
</Event> <!--End of event-->
```

In an actual Event, the Trigger, Condition, and action would contain more detail. Each of these tags is discussed subsequently in its own section. A complete Event appears in ["Example of a Complete Event" on page 49](#).



IMPORTANT! Event names must be unique within each XML configuration file.



IMPORTANT! Once an XML configuration file has been parsed by the Sync Engine, the names of the events are fixed and must not be changed or deleted. The event names are written to the registry and any changes to the event names interfere with proper execution of the integration.

Triggers

This section contains the Condition tags. A trigger can be thought of as one or more conditions.

No other tags may be placed in the Trigger section.

Conditions

This section defines the conditions that trigger the actions named in the event. When the conditions described in the Condition tag are met, the integration begins executing the actions in the Event.

Here is an example of a Condition:

```
<Condition>
  <Param event="included" />
  <Param name="collection" value="Scoping" />
  <Param name="class" value="Marketing_Requirements" />
  <Param name="project" value="ICDEMO3" />
  <!--If there are multiple Condition tags in a Trigger section, and
    if the Condition tags specify a project, each Condition tag
    must specify the same project. If no project is specified, the
    integration uses the project named in the DataSource section.
  -->

</Condition>
```

This says that when a Dimensions RM object is included in the collection named "Scoping", in the class named "Marketing_Requirements", in the Dimensions RM project "ICDEMO3", the actions named in the Event should be performed.

The example Condition specifies a project. This is useful if you are working with multiple Dimensions RM projects.

By default, the integration uses the project named in the DataSource section. If you use the project parameter to specify a different project, all Condition tags in that Trigger section must use that project. Only one Condition tag in the Trigger section needs to specify the project.

The event Parameter

The event parameter (not to be confused with the Event tag) defines the types of events you can watch for in a Condition tag. Here is a table showing the possible values for the event parameter, and their meanings:

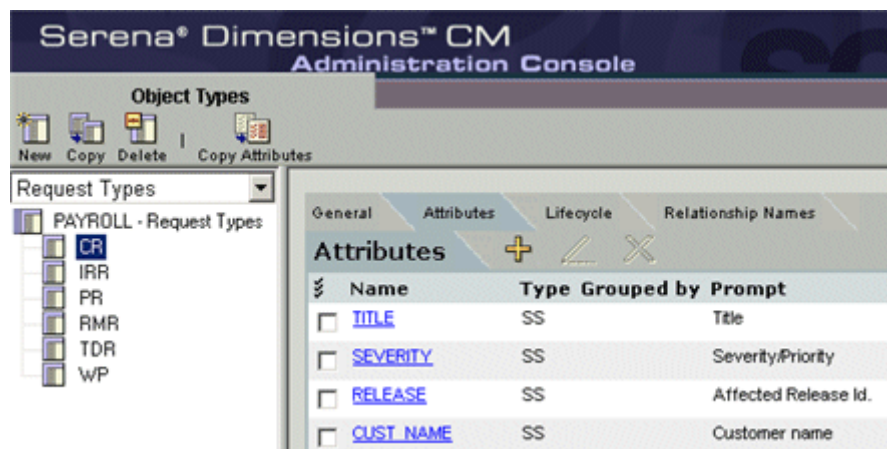
Name of event	Meaning in Dimensions RM	Meaning in Dimensions CM
included	Dimensions RM object is included in the named collection	Dimensions CM request matches the parameters (for example, named design part and request type); must also have been created since last synchronization run
excluded	Dimensions RM object is not included in the named collection	Dimensions CM request matches the named parameters AND was previously in the named state; for example, if state=RAISED, the request must have recently left the RAISED state
modified	Dimensions RM object has been modified	Dimensions CM request has been modified
deleted	Dimensions RM object has been deleted	Dimensions CM request was actioned to the defined end state



IMPORTANT! For events occurring in Dimensions CM, you can have only a single Condition within the Trigger.

What Dimensions CM Values Can Be Used for Params?

For Dimensions CM, you can use any of the attributes that appear in the **Attributes** tab for a request type in the Dimensions CM Administration Console:



IMPORTANT! Use the Name of the attribute, not the Prompt; that is, use "CUST_NAME" instead of "Customer name".

You can also use any of the System Attributes. These attributes do not appear in the Administration Console, but are listed here:

- PRODUCT – the product that we are referring to
- CHANGEDOCTYPE – the request type
- STATUS – the status value of the request
- DESIGNPART – design part that contains the request
- DESCRIPTION – the detailed description of the request (note that the description can be set when a request is created, but cannot be modified after that)
- TITLE – the title of the request
- ID – the ID value of the request (note this can only be read, not written)



IMPORTANT! You must use UPPERCASE when using these attributes in the XML configuration file.

How Do I Specify Multiple Values for a Parameter?

Normally you specify a parameter like this:

```
<Param name = "DESIGNPART" value ="PAYROLL:PAYROLL.A;1">
```

When you have to specify multiple values for a single parameter, use this construction:

```
<Param name = "DESIGNPART">
  <ParamValue value ="PAYROLL:APPLICATIONS.A;1"/>
  <ParamValue value ="PAYROLL:REPORTS.A;1"/>
</Param>
```

Actions

There is no <Action> tag. Most other sections of the XML configuration file have tags that match their name—for example, the Events section begins with a tag containing the word "Event". The tags in the Actions section, however, begin with one of the possible actions:

- Create
- Update
- Delete

Here is a table with summaries of what actions can be performed:

Tag	In Dimensions RM	In Dimensions CM
<Create>	Create a new Dimensions RM object.	Create a new request.

Tag	In Dimensions RM	In Dimensions CM
<Update>	Update an existing Dimensions RM object.	Update an existing request.
<Delete>	Delete a Dimensions RM object (remains a member of the named collection).	No other operations can be performed during the Delete action. To change the status of a request (for example, to REJECTED), use an Update action just prior to the Delete action. Be sure to include both actions in the same Event tag.

Here is an example of a Create action:

```
<Create name="create" datasource="RTM" description="Replace RTM object
  with Dimensions data">
  <Param project="ICDEMO3"/>
  <Field name="DIM_SEVERITY" value="SEVERITY"/>
  <Field name="DIM_DESCRIPTION" source="DESCRIPTION"/>
  <Field name="DIM_TITLE" source="TITLE"/>
</Create>
```



NOTE The project parameter in this example indicates that the ICDEMO3 project should be used. By default, the integration uses the project specified in the DataSource section. If you wish to refer to multiple Dimensions RM projects, you must specify the projects using a project parameter inside each Create, Delete, or Update action tag.

Here is an example of an Update action:

```
<Update name = "UpdateCD" datasource="DIM">
  <Param name="CHANGEDOCTYPE" value="CR"/>
  <Param name="PRODUCT" value="PAYROLL"/>
  <Param name="DESIGNPART">
    <ParamValue value="PAYROLL:PAYROLL.A;1"/>
  </Param>
  <Param name="SEVERITY" value="1_critical"/>
  <Field name="DESCRIPTION" source="Text"/>
  <Field name="TITLE" source="Text"/>
</Update>
```

Here is an example of a Delete action:

```
<Delete name="delete" datasource="DIM" description="Mark the Dim CD as
  deleted.">
  <Param name="PRODUCT" value="PAYROLL"/>
</Delete>
```



IMPORTANT! The Delete action is sufficient to mark the request so that the Sync Engine no longer considers the request when doing updates. However, you may wish to do something to change the appearance of the request in Dimensions CM—for example, changing the status to REJECTED, or changing the title of the request. If you wish to do this, you must do it in an Update action immediately preceding the Delete action. Be sure to include both actions in the same Event tag.

Fields in Actions

Within action elements, <Field ...> elements have a special behavior. They 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 is interpreted as a literal value for the field.

A field may also contain data from the event object (the Dimensions RM object or Dimensions request in which the event occurs). This will be indicated with the source attribute. 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="DIM_ID" />
```

The use of the {0} token in the text attribute value creates a composite value (for example, "RTM Issue 0009"). The {0} token is replaced with the current value of the DIM_ID field.



NOTE This release supports only one source value, so only the token {0} will be replaced. The format leaves open the possibility of tokens such as {1}, {2}, and so forth in the future, allowing for multiple source attributes.

If you use the {0} token in a text attribute and do not specify a source attribute, the token is replaced with nothing.

Example of a Complete Event

Here is an example showing an Event, Trigger, Condition, and action:

```
<Event name="RTMincluded" datasource="RTM" description="Start
Requirement Scoping">
  <Trigger>
    <Condition>
      <Param event="included" />
      <Param name="collection" value="Scoping" />
      <Param name="class" value="Product_Requirements" />
      <Param name="class" value="Marketing_Requirements" />
    </Condition>
  </Trigger>
  <Create name="create" datasource="DIM" description="create a
corresponding Dim item">
    <Param name="CHANGEDOCTYPE" value="CR"/>
    <Param name="PRODUCT" value="PAYROLL"/>
    <Param name="DESIGNPART">
      <ParamValue value="PAYROLL:PAYROLL.A;1"/>
    </Param>
    <Param name="SEVERITY" value="1_critical"/>
    <Field name="DESCRIPTION" source="Text"/>
    <Field name="TITLE" source="Text"/>
  </Create>
</Event>
```

```
</Create>  
</Event>
```

</IntegrationConfiguration>

This tag closes the <IntegrationConfiguration> tag that occurs at the start of the file. Be sure that the XML configuration file ends with this tag:

```
</IntegrationConfiguration>
```



NOTE You can find example XML configuration files in the *RM_Install_Dir\conf* directory.

Error Handling for Create, Update, and Delete Actions

If the Dimensions RM integration is unable to complete a Create, Update, or Delete action, it attempts to re-execute the entire action and all subsequent actions the next time the integration processes the surrounding Event.

Error Handling for Create Actions

A Create action seeks to create a new data record in one application in order to match new data created or added to the other application.

If the Create action fails, the integration reports an error. In addition, the integration attempts to delete new items and links created in the steps outlined previously. This is done as follows:

- In Dimensions RM, the integration marks orphaned items or links for deletion.
- In Dimensions CM, if the integration is unable to complete the Create action when it is creating a new proxy object, the request that was created remains. If the request appears as a duplicate after the next Event processing run, you should delete the duplicate.

Error Handling for Update Actions

If the Update action fails, there should be no need to roll back or delete anything. The integration simply reports the failure to Update as an error in the Sync Engine log.

Error Handling for Delete Actions

A Delete action seeks to delete an item in one application in order to match data deleted in the other application. The action also deletes links to that item in both applications.

If the Delete action fails, the integration reports an error but will not attempt to roll back any delete operations. If an item or link existed and could not be deleted, the integration reports that as an error.

The next time the integration executes, it retries the Delete action. If an item was deleted during a previous Delete action, no additional error is reported if the Delete action had to be retried.

Editing the XML File for the Advanced Configuration

If you use the advanced configuration, you must set up everything in the basic configuration—events, actions, value maps, and so on. In addition, you must set up proxies, and modify the events to use the proxies.

Modifying Events to Use Proxies

If you decide to use proxy classes, you need to make the following changes to the XML configuration file.

Add a ProxyDefs Section to Define the Proxy

After the ValueMaps section, add a section called ProxyDefs, as shown here:

```
<!-- ProxyDefs
  name - name of the proxy definition.
        Used as a reference by the SyncEngine.
  description - description of the proxy. -->
<ProxyDef name="DIM_Item_Proxy" datasource="RTM" description="Test
  ProxyDef1">
  <Param class="Marketing_Requirements_Proxy" />
  <Param relationship="Marketing_Requirements_To_Proxy" />
  <Field name="STATUS" text="Current" />
  <Field name="Dim ID" source="ID" />
  <Field name="Text" source="TITLE" />
  <Field name="Current Status" source="STATUS" />
</ProxyDef>
```

You must create a ProxyDef for each of the proxies you created in the Class Definition tool. Each ProxyDef needs:

- a Param for the name of the proxy class
That is, if you started with a class named Marketing_Requirements, then created a proxy class called Marketing_Requirements_Proxy, the Param here must match "Marketing_Requirements_Proxy".
- a Param for the name of the relationship from the proxy class to the original class
- a Field for each of the attributes defined in the proxy class



NOTE In the line `ProxyDef name="DIM_Item_Proxy"`, the name does not need to match the name of the proxy class created in the Class Definition tool.

See [Chapter 3, "Setting Up Dimensions CM and Dimensions RM" on page 29](#) for a discussion of how to define a proxy class for the integration.

Add the Proxy Name to the Create Events

In the Create events, add the following line, between the name of the event and the name of the data source:

```
proxy="Name_of_Proxy"
```

where *Name_of_Proxy* must be replaced by the name of the proxy instance.

Example of the first line of the Create event NOT using proxies:

```
<Create name="create" datasource="DIM" description="Create a DIM item">
```

Example of the first line of the Create event using proxies:

```
<Create name="create" proxy="DIM_Item_Proxy" datasource="DIM"
  description="Create a DIM item">
```

It is not necessary to add the name of the proxy to the other events.



NOTE A complete example of an XML configuration file that uses proxies appears in the *RM_Install_Dir\conf* directory.

Differences Between Events

When you create a new request based on a Dimensions RM object, you must provide the name of the proxy.

When you update an existing request based on a change to a Dimensions RM object, you don't have to provide the name of the proxy.

Validating the XML Configuration File

When you attempt to run the Sync Engine, the Sync Engine validates 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-CM integration attempts 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 section 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 data source 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 gives the line number and this should be a good indication of where the error is.

ERROR SyncEngine.RTM - RTM Database Error (942): 942, ORA-00942: table or view does not exist

This error message generally indicates that the name of the class in the condition is incorrect. This message may be followed by a message indicating that the Sync Engine failed to retrieve any objects from the class in question:

DEBUG SyncEngine.RTM - Failed to retrieve <Name_of_Class> objects.

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 errors by using one of the following:

- Altova XMLSpy
- Pixware Xsdvalid

These tools typically have more descriptive error messages and can help you identify the problem.

The Logging Configuration File

The logging configuration file determines how the RM-CM integration logs information to a file.

This file (`RM_Install_Dir\conf\log4cpp.conf` by default) is not as important to you as the XML configuration file, but nevertheless controls the following:

- The level of error detail
- 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 four categories:

- fileBrowser
- fileSyncEngine
- fileWebService
- fileDbDoctor

What You Can Change

You can change the level of error detail logged (DEBUG, INFO, WARN, ERROR)—different values provide a greater or lesser amount of detail in the log.

You can change the location of the logging file by updating the `log4cpp.conf` file. This file is located in the `<RM Install Dir>\conf` directory.

You can set limits for the size of the log file and the maximum number of log files maintained (see [Controlling Log File Size](#) below).

You can change the layout of the output of the log file by editing either or both of two lines that include `log4j.appender.file<category>.layout`. The layout uses standard log4j formatting; do not edit these lines if you are unfamiliar with this format.



NOTE You must not change anything else in the `log4cpp.conf` file.

Controlling Log File Size

You can limit the maximum size of the log file so that it doesn't consume excessive system resources. You accomplish this by specifying a "rolling file appender", which starts a new file (such as `syncEngine.log.2`) after the first file exceeds the specified size limit. In the following example, the maximum file size is 5,000,000 bytes, and only the three most recent log files are retained at one time:

```
log4j.appender.file<category>=org.apache.log4j.RollingFileAppender
log4j.appender.file<category>.maxFileSize=5000000
log4j.appender.file<category>.maxBackupIndex=3
```

Note that if you set `log4j.appender.file<category>.append` to `false` and you are using the rolling file appender, the Sync Engine starts a new log file each time that you start the service, regardless of whether the maximum file size has been reached.

If you are not using the rolling file appender and you set `log4j.appender.file<category>.append` to `false`, the Sync Engine overwrites the log file each time that the service starts.

Example Logging Configuration File

```
# RTM log4cpp config

# no output generated for a given category if an appender isn't specified
log4j.rootCategory=DEBUG
log4j.category.icBrowser=WARN, fileBrowser
log4j.category.SyncEngine=WARN, fileSyncEngine
log4j.category.rtmService=WARN, fileWebService
log4j.category.dbDoctor=INFO, fileDbDoctor

log4j.appender.fileBrowser=org.apache.log4j.RollingFileAppender
log4j.appender.fileBrowser.append=true
log4j.appender.fileBrowser.maxFileSize=5000000
# Keep two backup files
log4j.appender.fileBrowser.maxBackupIndex=2
log4j.appender.fileBrowser.fileName=D:\Program Files\Serena\Dimensions 10.1\RM\logs\rtmBrowser.log
log4j.appender.fileBrowser.layout=org.apache.log4j.PatternLayout
log4j.appender.fileBrowser.layout.ConversionPattern=%d{%m-%d-%y %H:%M:%S.%1} %-6p %-20c - %m%n

log4j.appender.fileSyncEngine=org.apache.log4j.RollingFileAppender
log4j.appender.fileSyncEngine.append=true
log4j.appender.fileSyncEngine.maxFileSize=5000000
#Keep two backup files
log4j.appender.fileSyncEngine.maxBackupIndex=2
log4j.appender.fileSyncEngine.fileName=D:\Program Files\Serena\Dimensions 10.1\RM\logs\syncEngine.log
log4j.appender.fileSyncEngine.layout=org.zpache.log4j.PatternLayout
log4j.appender.fileSyncEngine.layout.ConversionPattern=%d{%m-%d-%y %H:%M:%S.%1} %-6p %-20c - %m%n

# for webservice.
# NOTE: The filepath for the webservice log needs to be in a directory where the IIS user that is set to run the
# service has write privileges
log4j.appender.fileWebService=org.apache.log4j.RollingFileAppender
log4j.appender.fileWebService.append=true
log4j.appender.fileWebService.maxFileSize=5000000
#Keep two backup files
log4j.appender.fileWebService.maxBackupIndex=2
log4j.appender.fileWebService.fileName=D:\Program Files\Serena\Dimensions 10.1\RM\logs\webService.log
log4j.appender.fileWebService.layout=org.apache.log4j.PatternLayout
```

```
log4j.appender.fileWebService.layout.ConversionPattern=%d{%m-%d-%y %H:%M:%S.%1} %-6p %-20c - %m%n

log4j.appender.fileDbDoctor=org.apache.log4j.RollingFileAppender
log4j.appender.fileDbDoctor.append=true
log4j.appender.fileDbDoctor.maxFileSize=5000000
#Keep two backup files
log4j.appender.fileDbDoctor.maxBackupIndex=2
log4j.appender.fileDbDoctor.fileName=D:\Program Files\Serena\Dimensions 10.1\RM\logs\dbDoctor.log
log4j.appender.fileDbDoctor.layout=org.apache.log4j.PatternLayout
log4j.appender.fileDbDoctor.layout.ConversionPattern=%d{%m-%d-%y %H:%M:%S.%1} %-6p %-20c - %m%n

log4j.appender.debugger=org.apache.log4j.Win32DebugAppender
log4j.appender.debugger.layout=org.apache.log4j.PatternLayout
log4j.appender.debugger.layout.ConversionPattern=%d{%H:%M:%S.%1} %-6p %-20c - %m%n
```

Next Steps

After you have edited the XML configuration file and the logging configuration file, you are ready to try running the integration. The next chapter shows you how to do that.

Chapter 5

Running the Integration

Introduction	58
Things You Should Do Before Running the Integration	58
Creating a Dimensions CM Request from a Dimensions RM Object	58
Next Steps	62

Introduction

When you have edited the XML configuration file to add password information and other information specific to your site, you are ready to try running the RM-CM integration.

As mentioned earlier, the Sync Engine is a Windows service that is installed as an option when you install Dimensions RM.

You should review ["Things You Should Do Before Running the Integration,"](#) and then begin with ["Creating a Dimensions CM Request from a Dimensions RM Object" on page 58.](#)

Things You Should Do Before Running the Integration

- Make a backup copy of your XML configuration file.
- Name the working XML configuration file `config.xml` and place it in `RM_install_directory\conf`. If you need to name the XML configuration file something else, then follow the procedure in ["Configuring the Sync Engine Service to Use a Specific XML Configuration File" on page 67.](#)
- Be sure you have created the Sync User and added the report views, as described in ["Setting Up Dimensions CM" on page 30.](#)

Creating a Dimensions CM Request from a Dimensions RM Object

This section describes how to use the RM-CM integration to create a new Dimensions CM request from a Dimensions RM object.

Here is a suggested set of steps for you to get started:

- 1 Create some test data to be transferred to the other application.
- 2 Start the Sync Engine service.
- 3 Wait for the integration to complete a cycle.
- 4 Check the destination application to see if the test data was transferred successfully.
- 5 If the data was not transferred, check the log files and see if the integration is behaving as you desire.

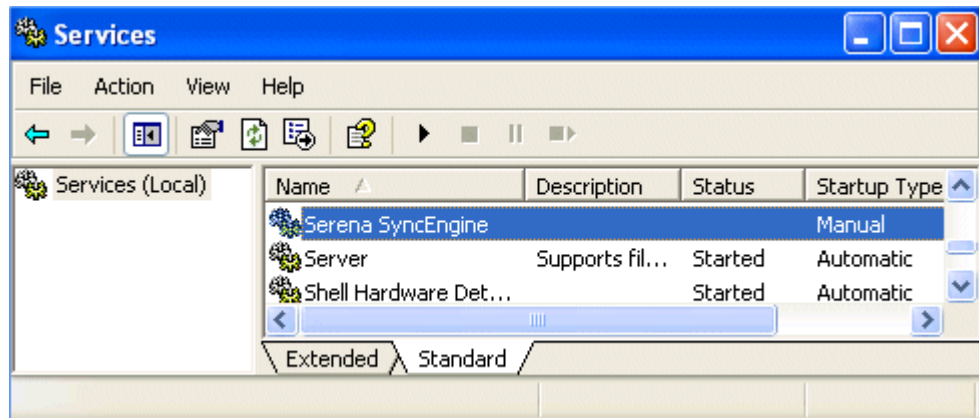
For the examples in this chapter, the data is being sent from Dimensions RM to Dimensions CM. Transferring data from Dimensions CM to Dimensions RM is similar, and it is best to start with data going in only one direction at first.

Creating Test Data

Since the XML configuration file is set up to watch a particular collection, you should create a new Dimensions RM object in that collection. If you use ICDEMO, for example, you might have set up the XML configuration file to watch the Marketing_Requirements or Product_Requirements collections.

Starting the Sync Engine Service

To start the Sync Engine service, use the Windows Service Manager:



As it is likely that you will be stopping and restarting the service several times as you test the XML configuration file, move the Services dialog to a convenient location and use the Start Service and Stop Service controls:



TIP The green Refresh icon in the Services dialog refreshes the display. This can be useful when you want to verify that the service has been restarted.

You can also use `syncengine -k start` from a command prompt. For a list of the possible command prompt options, see ["Controlling the Sync Engine from a Command Prompt" on page 66](#).

Waiting for a Cycle to Complete

After you have started the Sync Engine service, let it run for a little while so that the integration has a chance to complete a data exchange cycle.

How long should you wait before checking? Use the values specified for `interval` and `sleep` in the XML configuration file as a guide. If the value for `interval` is two minutes, wait at least that long.

Checking the Destination Application

After you have waited for the integration to complete a data exchange cycle, check the destination application to see if the data was transferred successfully. For example, check

the list of change documents in the appropriate Dimensions CM design part to see if the data from the Dimensions RM object was successfully transferred.

Checking the Log Files

If the data was not transferred successfully, check the log files to see what happened. The Sync Engine creates two log files when the integration starts:

- `synceng.log`
This file captures information from the startup activities of the Sync Engine.
- `syncEngine.log`
This file captures information from the post-startup activities of the Sync Engine, such as synchronization activity.

You can find the log files in `RM_Install_Dir\logs`.

Following is a list of questions and answers that can help you understand the information in the log files.

Was any information was added to the log files at all?

If the log files have a file size of zero, or if the files are not even created, then something is not set up correctly. Search for the `syncEngine.log` and `synceng.log` files in case they were created somewhere else.

Was the Sync Engine service started successfully?

If the Sync Engine starts successfully, you will see a message similar to the following in `syncEngine.log`:

```
07-14-05 10:40:53.484 ALERT SyncEngine - - The Sync Engine  
(1.0.0.6) is starting...
```

If the Sync Engine can't find the XML configuration file, you will see a Windows error dialog. A message similar to the following also appears in the `synceng.log`:

```
ERROR -  
Parser Fatal Error at file , line 0, char 0  
Message: An exception occurred! Type:RuntimeException, Message:The  
primary document entity could not be opened. Id=E:\data\config.xml
```

If the Sync Engine can't find the `SyncEngine.xsd` file, you will see a Windows error dialog indicating that the Sync Engine service could not be started. A message similar to the following also appears in the `synceng.log`:

Parser Warning at file , line 0, char 0

Message: An exception occurred! Type:RuntimeException,
Message:Warning: The primary document entity could not be opened.
Id=e:\data\SyncEngine.xsd
FATAL - Exception during DOM Initialization:

Is there enough detail in the log files?

If the log files have information, but do not contain enough detail, you can set the level of logging detail to a more verbose level. See ["What You Can Change" on page 54](#).

Was the connection to Dimensions RM opened successfully?

If the connection to Dimensions RM was not opened successfully, you will see a message similar to the following in syncEngine.log:

```
07-14-05 11:14:42.828 ALERT   SyncEngine           - - The Sync Engine
      (1.0.0.6) is starting...
07-14-05 11:14:44.875 ERROR   SyncEngine.RTM       - RTM Error (75):
      Logon failure
07-14-05 11:14:44.875 ERROR   SyncEngine           - !! Failed to open
      connection to DataSource 'RTM' (Serena RTM Sync Proxy). RTM Error
      (75): Logon failure
07-14-05 11:14:44.875 WARN    SyncEngine           - OpenConnection failed
      for datasource 'RTM'
```

This likely means that the login information for RTM is not correct.

Was the connection to Dimensions CM opened successfully?

If the connection to Dimensions CM was not opened successfully, you will see a message similar to the following in syncEngine.log:

```
07-14-05 11:28:38.984 ERROR   SyncEngine.DIM       -
Error 001: Login to Dimensions failed.
Error: User authentication failed

Failed to start Dimensions server process

07-14-05 11:28:38.984 ERROR   SyncEngine.DIM       -
07-14-05 11:28:38.984 ERROR   SyncEngine           - !! Failed to open
      connection to DataSource 'DIM' (Serena Dimensions Sync Proxy).
07-14-05 11:28:38.984 WARN    SyncEngine           - OpenConnection failed
      for datasource 'DIM'
```

This message appears if the user ID or password for Dimensions CM is not correct.

If there is an error in the name of the database, you will see this message:

```
Error 001: Login to Dimensions failed.
1019-DBIO: Database I/O error occurred
ORA-01017: invalid username/password; logon denied
Error: Unable to connect to database "Intermediate_wrong"
```

If there is an error in the name of the database connection, you will see this message:

```
Error 001: Login to Dimensions failed.  
1019-DBIO: Database I/O error occurred  
ORA-12154: TNS:could not resolve service name
```

Was the correct Dimensions RM collection recognized?

If the integration appears to be connecting to Dimensions RM and to Dimensions CM successfully, but no information is being transferred, check the log for a message similar to the following:

```
Successfully retrieved 0 Name_of_Class objects.
```

If zero objects are retrieved, the integration is having trouble finding a match. This could be caused by any of the following:

- Name of the project is wrong
- Name of the class is wrong
- Name of the collection is wrong
- The new objects were created since the last time the Sync Engine ran but before a successful connection was made

In the ProxyDef, are you calling the proxy class and not the original class?

If you are calling the original Dimensions RM class instead of the proxy class, the Dimensions RM objects will be recognized, but you may not be able to exchange any information with Dimensions CM. Check the ProxyDef section of the XML configuration file.

Is the name of the product correct?

If you have mistyped the name of the product, you will see an error message like this:

```
Error 090: Product PAYROLL_BAD does not exist in this base database.
```

Is the name of the design part correct?

If you have mistyped the name of the design part, you will see an error message like this:

```
Error 001: Creating Change Document failed. (CreateCD)  
1111-Design Part PAYROLL:BADPART.A;1 does not exist
```

Next Steps

Successfully creating a new Dimensions CM request from a Dimensions RM object is a good start in mastering the RM-CM integration. Depending on your business needs, you may also be interested in going on to the following tasks:

- Update an Dimensions RM object and use an Update event to have the change appear in the linked Dimensions CM request.

- Remove a Dimensions RM object from a collection and use a Delete event to have the linked Dimensions CM request change status to REJECTED (or another status of your choice).
- Do all of the above using proxies.
- Create a new Dimensions CM request and have a new Dimensions RM object created.
- Update a Dimensions CM request and have the updated information appear in the linked Dimensions RM object.
- Change the status of a Dimensions CM request to REJECTED and have the linked Dimensions RM object deleted.



NOTE The use of proxies is not supported when Dimensions CM data is being sent to Dimensions RM.

Chapter 6

Sync Engine Command Prompt Usage and Special Cases

Introduction	66
Controlling the Sync Engine from a Command Prompt	66
Configuring the Sync Engine Service to Use a Specific XML Configuration File	67
Reinstalling the Sync Engine Service	67
Running with Multiple Sync Engines	68

Introduction

This chapter presents information on how to use a command prompt to control the Sync Engine component of the RM-CM integration, and how to configure the Sync Engine for special situations.

Controlling the Sync Engine from a Command Prompt

Certain Sync Engine operations are more convenient from a command prompt. For example, if you want to use an XML configuration file that is not in the default location, it is easy to use the command prompt to accomplish this (see ["Configuring the Sync Engine Service to Use a Specific XML Configuration File" on page 67](#)).

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

```
Usage  syncengine [-f file] [-k {install|config|uninstall|start|stop}]
        [-n serviceName] [-L serviceName] [-v] [-h] [-p password]
        [-P eventName]
```

Options

Option	Description
-k install	Installs a Sync Engine service.
-k config	Changes the startup options for a Sync Engine service; for example: syncengine -k config -f <i>new config file</i> .
-k uninstall	Uninstalls a Sync Engine service.
-k start	Starts the Sync Engine service.
-k stop	Stops the Sync Engine service.
-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.
-f <i>file</i>	Specifies an alternative configuration file.
-h	Lists the available command-line options.
-n <i>serviceName</i>	Sets the service name and specifies that the corresponding configuration file will be used.
-L <i>servicePriority</i>	Sets the Windows system priority for the Sync Engine service. <i>servicePriority</i> 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.

Option	Description
-p <i>password</i>	Encrypts the password text for use in the configuration file.
-P <i>eventName</i>	Purges the queue of data to be processed of all objects associated with the named event. This is used when orphaned objects (for example, from a disconnection) are beginning to affect performance. You can remove objects from multiple events by using the -P switch more than once: "-P RTMIncluded -P RTMExcluded" If the event named is not in the default config.xml file, use the -f option to specify the configuration file where the event is defined.
-v	Displays the version number.

Configuring the Sync Engine Service to Use a Specific XML Configuration File

The default XML configuration file is named *config.xml* and is expected to be in the conf subdirectory of the Dimensions RM installation directory.

If you need to use an XML configuration file that is not in the default location, or is named something other than *config.xml*, use the -k config option to make these changes to the service:

```
syncengine -k config -f Location_and_Name_of_Your_Config_File
```

This avoids the need to uninstall and reinstall the service.

You should make this configuration change before attempting to start the service for the first time.

Reinstalling the Sync Engine Service

The Sync Engine service is installed as an option with Dimensions RM, though the service is not automatically started.

If you need to reinstall the Sync Engine service, use the -k install option. This creates an instance of the service with all the other parameters that you specify.



IMPORTANT! Do not change the name of the Sync Engine service once you have begun running it. The name of the service is used as a key into the registry.

If you need to install more than one instance of the Sync Engine service, see ["Running with Multiple Sync Engines,"](#) next.

Running with Multiple Sync Engines

To install a second Sync Engine service, use the `-k install` option with the `-n serviceName` option. Otherwise, you will be unable to distinguish the two services. (Possible reason for installing a second Sync Engine service: to synchronize with both Dimensions CM and TeamTrack.)

To uninstall the Sync Engine service, use the `-k uninstall` option. You must specify the same service name (`-n serviceName`) that you used when you installed the service; otherwise, the service will not be correctly uninstalled.

If Dimensions RM itself is uninstalled, the default Sync Engine service will be uninstalled at the same time.

If you have installed a second Sync Engine service, that service will not be uninstalled when Dimensions RM is uninstalled. You must uninstall it yourself using the following command:

```
syncengine -k uninstall -n NameOfSecondSyncEngineService
```

Chapter 7

Use Cases

Introduction	70
Scoping a Requirement	70
Applying a Dimensions Lifecycle to Requirement Approval	71
Implementing Requirements	71
Defect and Enhancement Management	71

Introduction

This chapter contains a few narrative descriptions of using the integration between Serena® Dimensions® RM and Serena® Dimensions® CM. These use cases demonstrate how the integration can be used for application lifecycle management.

Scoping a Requirement

Requirements can participate in several different processes or workflows as they move from inception to completion. For example, after a user enters a requirement into Dimensions RM, the requirement may require input from multiple team members using Dimensions CM. This might be an important step in the process of assessing the cost of the requirement.

Responses from the various Dimensions CM users can be linked to a single Dimensions RM object. Each team member can be assigned a Dimensions CM request requesting his or her evaluation of the requirement. As the team members enter their responses into Dimensions CM, the RM-CM integration can watch for the addition of new information to the Dimensions CM requests—already linked to the Dimensions RM object—and return that information to Dimensions RM. The requirement creator can then use Dimensions RM to review the returned information.

Example Details

Let's say that the Dimensions RM requirement, once created, needs to be reviewed by Development, Quality Assurance, and Documentation for initial scoping.

Knowing in advance that the Dimensions RM requirement requires comments from multiple departments, the requirement creator sets up a proxy class and defines a relationship between the proxy class and the original class. The RM-CM integration uses this relationship to create proxy objects that can receive the information entered by the Dimensions CM users.

Once the proxy class and relationship have been set up, the Dimensions RM user creates a requirement (an object of a particular class) and adds it to a collection called "Need Scoping".

The integration triggers a Create event to create a new Dimensions CM request to represent the scoping task. The Dimensions CM request has read-only fields set up to receive the appropriate requirement information from the Dimensions RM requirement.

The Dimensions CM users examine the requests in their Pending lists, and begin to add their comments and evaluations. After an interval, the integration runs again, and the new information is sent back to Dimensions RM. (The time interval for updates is based on settings in the integration's XML configuration file.) Specifically, the Dimensions CM information is stored in read-only fields in instances of the proxy class created earlier by the Dimensions RM user.

By examining all of the objects linked to the original requirement, the Dimensions RM user can view the requirement comments returned by the different departments.

Applying a Dimensions Lifecycle to Requirement Approval

As requirements move through the planning lifecycle, you may have approval paths and other workflow needs that cannot be addressed by Dimensions RM as well as through the lifecycle in Dimensions CM. A Dimensions RM requirement's lifecycle can be executed by Dimensions CM to provide the extended workflow support.

Example Details

You can add events in the integration's XML configuration file to create Dimensions CM requests and return the changes in status to the requirement in Dimensions RM.

If a formal approval process is needed for certain requirements, those requirements can be added to a special collection. Based on the XML configuration file, the integration can create a Dimensions CM request. While the Dimensions CM request goes through the changes in status defined in its workflow (for example, from RAISED to ANALYSED), the Dimensions RM user can see the changes in status in Dimensions RM by running reports. When the Dimensions CM request is approved, the Dimensions RM user can change the requirement "approval status" attribute accordingly.

Implementing Requirements

In an implementation similar to ["Scoping a Requirement,"](#) a Development Manager could move the requirements from the "Need Scoping" collection to the "End of Year Release" collection and use them to distribute tasks to the developers.

Again using the RM-CM integration to create change documents based on Dimensions RM objects, the Development Manager could assign the tasks required by the "End of Year Release" requirements, and observe the changes in status using Dimensions RM reports.

Defect and Enhancement Management

As defects and enhancements are entered into Dimensions CM, it may be necessary to seamlessly move these requests from Dimensions CM into Dimensions RM requirements. The defects and enhancements can then be treated as would any other Dimensions RM requirement in your planning process (for example, during the planning for the next release).

Example Details

After a new defect or enhancement is captured in Dimensions CM, a new requirement can be created in Dimensions RM. For example, an enhancement request can be added to a collection named "Future Enhancements". Defects that fell outside the release cycle can be added to a collection named "Future Bug Fixes".

Once in the "Future Enhancements" or "Future Bug Fixes" collections, the Dimensions RM objects can be evaluated for further attention. Some enhancements or bug fixes may be rejected immediately as impossible; others are retained for further evaluation. These items can be moved to a "Need Scoping" collection and handled as in ["Scoping a Requirement" on page 70](#).

Index

A

- actions
 - Create 47
 - Delete 48
 - limitations of 48
 - table of possible 47
 - Update 48
- adding report views to Dimensions CM 30
- advanced configuration
 - behavior 21
 - compared with basic 26
 - setting up proxy class 31
- Altova XMLSpy 54
- attributes
 - System Attributes 47
 - use Name, not Prompt 46
 - which Dimensions CM values can be used as parameters 46
- audience 7

B

- basic configuration
 - behavior 16
 - compared with advanced 26
- behavior when data goes from Dimensions CM to Dimensions RM 19
- Boolean values
 - must be lower-case 44

C

- CHANGE-MANAGER 30
- CHANGE-MANAGER user 12
- changing the status of a Dimensions CM request 48
- command-line usage 66
- conditions
 - example 45
 - in Dimensions CM, only single condition allowed in a Trigger 46
- config.xml 10
- configuring the integration 37
- creating a Dimensions CM request from a Dimensions RM object 58

D

- data sources
 - exactly two required 41
 - sample 41
- date fields
 - must match in Dimensions RM and Dimensions CM 43
- DEBUG 54
- deciding which projects to synchronize 28
- defect management (use case) 71
- deleted (event parameter) 46
- Dimensions CM
 - adding report views 30
 - need for desktop client 11
 - requirement for CHANGE-MANAGER user 12
 - selection fields 43
 - setting up 30
 - supported version 11
- Dimensions RM
 - list attributes 43
 - manuals 7
 - multiple projects specified in actions 48
 - multiple projects specified in conditions 45
 - Synchronization Engine 7
 - using multiple projects with integration 13

E

- enhancement management (use case) 71
- ERROR 54
- error
 - data source (in Event) not valid 53
 - data source (in ProxyDef or action) not valid 53
 - design part does not exist 62
 - Dimensions CM connection not opened 61
 - Dimensions RM connection not opened 61
 - duplicate key value declared 53
 - key for identity constraint not found 53
 - leftsource/rightsource not valid 53
 - level of detail logged 54
 - log file size zero 60
 - product does not exist 62
 - provider not valid 53
 - Sync Engine not started 60
 - table or view does not exist 53
 - zero objects retrieved 62

- error handling
 - create actions 50
 - create, delete, and update actions 50
 - delete actions 50
 - update actions 50
- event parameter (not same as Event tag)
 - table of possible values 46
- events
 - modifying to use proxies 51
 - names must be unique within XML configuration file 45
 - names must not be changed once parsed 45
 - pseudo-code example 44
- example use cases 70
- example XML configuration file 50
- excluded (event parameter) 46

F

- fields
 - in actions 49
 - variable value 49

H

- hints 54

I

- included (event parameter) 46
- INFO 54
- integration concepts 15
- interval 40
- interval for exchanging data 12

L

- licensing 12
- log file
 - controlling size of 55
 - where to find 60
- log4cpp.conf
 - should not be renamed 10
- log4j.appender.file 55
- log4j.appender.file.layout 55
- logging configuration file
 - default location 54
 - example 55
 - log4j formatting of 55
 - things controlled by 54

M

- manuals
 - Dimensions CM 7
 - Dimensions RM 7
- mapping n to m values 43
- max_connections parameter
 - defined 42
 - example 41
- Microsoft Notepad 38
- modified (event parameter) 46
- multiple Dimensions RM projects
 - and actions 48
 - and conditions 45
 - overview explanation 13

O

- one-to-many 21, 23
- one-to-one 21
- original object modified if field not in proxy object 26
- overview 9

P

- parameters
 - specifying multiple values 47
- password encryption 67
- PATH environment variable 38
- Pixware Xsdvalid 54
- possible errors 53
- provider DLLs 41
- proxy
 - adding name to Create event 52
 - not supported when mapping Dimensions CM data to Dimensions RM 35
- proxy class
 - diagram 23
 - setting up 31
- proxy classes
 - where created 25
- proxy instances
 - where created 25
- ProxyDef
 - original class versus proxy class 62
 - required elements 51

R

- read-only fields
 - in application receiving data 18
 - to avoid data loss 19

Refresh icon in Services dialog 59
 related documentation 7
 RM-CM integration
 advanced configuration 21
 basic configuration 16
 before running 58
 components 10
 described briefly 10
 diagram of Dimensions RM and Dimensions
 CM clients 11
 explanation of how it works 12
 if using another integration 13
 supported Dimensions CM version 11
 who should set up 12
 who should use 10
 RM-TeamTrack integration 13
 RollingFileAppender 55
 running the integration 57

S

scoping a requirement (use case) 70
 setting up a proxy class in Dimensions RM 31
 setting up Dimensions CM 30
 Setting Up Dimensions CM and Dimensions RM
 29
 sleep 40
 suggested work environment 39
 Sync Engine
 command prompt options 66
 command prompt usage 65
 command-line usage 66
 do not change name of service 67
 reinstalling 67
 running multiple 68
 starting the service 59
 uninstalling 68
 Sync User
 cannot be existing Dimensions CM user 30
 creating 30
 synceng.log 60
 syncEngine.log 60
 SyncEngine.xsd 52, 60

T

triggers 45

U

uninstalling
 need to uninstall manually if using multiple
 Sync Engines 13

uninstalling the Sync Engine 68
 use cases
 defect and enhancement management 71
 scoping a requirement 70
 UTF-8 encoding 38

V

validating the XML configuration file 52
 value maps
 leftsource required, rightsource optional 43
 sample 43
 version number switch 67

W

WARN 54
 Welcome 7

X

XML configuration file
 </IntegrationConfiguration> 50
 <IntegrationConfiguration> 40
 case-sensitive 39
 checking against schema 66
 default location 38
 default name 10
 editing 38
 editing advanced configuration 51
 editing basic configuration 39
 examples 50
 general options 40
 header 40
 how to use alternative location 67
 structure of 39
 validating 52

