SERENA®

DIMENSIONS® RM 2010 R1

Integration Guide for SBM

Serena Proprietary and Confidential Information

Copyright © 2001–2010 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., 1900 Seaport Blvd., Redwood City, CA 94063.

Publication date: September 2010

Table of Contents

Welcome **7**

Chapter 1 **Overview** **9**

What Is the RM-SBM Integration? 10
Who Should Use the RM-SBM Integration? 10
What Do I Have to Install to Use the RM-SBM Integration? 10
Who Should Configure the RM-SBM Integration? 11
How Often Does the RM-SBM Integration Synchronize the Data? 11
How Does the RM-SBM Integration Work?. 11
What If I Want to Use Multiple Dimensions RM Projects? 12
What If I Am Also Using Another Dimensions RM Integration?. 12

Chapter 2 **Setting Up Dimensions RM to Integrate with SBM** **13**

Dimensions RM Proxy Classes for SBM Items 14
To Proxy or Not to Proxy?. 14
Setting Up a Proxy Class in Dimensions RM 14
Behavior When SBM Reads Data from Dimensions RM 15

Chapter 3 **Setting Up SBM to Integrate with Dimensions RM** **17**

SBM Workflow Guidelines 18

Chapter 4 **Sample Integration** **19**

Overview 20
Supporting Files 20
SBM Setup 20
Create "SyncAdmin" User. 21
Create "Sync_Update" Transition 21
Create Fields 23
Edit Transition 23
Clear Host IP Address 24
Dimensions RM Setup. 24
Create User 24
Create Collection. 24
Create Attributes. 24
Start Oracle 25
Sync Engine 25
Edit the Configuration File 25
Start the Sync Engine Service. 25
Use Cases. 25
Scope a Requirement. 25
Update a Requirement 26
Remove Requirement from Collection 26

Delete a Requirement 26
Promote Enhancement Request to Requirement 27
Update the Enhancement Request 27
Transition the Enhancement Request 27
Cancel the Enhancement Request 28
Delete an Enhancement Request 28

Chapter 5 **Configuring the Sync Engine 29**
Introduction 30
The XML Configuration File 30
Character Encoding and Text Editor Considerations 30
Specifying General Options 30
Specifying Data Source Providers 31
Specifying Data Sources 31
Specifying Value Mappings 34
Specifying Proxy Definitions 35
Specifying Events and Actions 35
Error Handling for Create, Update, and Delete Actions 39
Error Handling for Create Actions 39
Error Handling for Update Actions 40
Error Handling for Delete Actions 40
Validating the XML Configuration File 40
Possible Errors 41
Hints 42

Chapter 6 **Running the Sync Engine 43**
Controlling the Sync Engine from the Command Line 44
Installing the Sync Engine Service 45
Logging 45
The Logging Configuration File 46

Chapter 7 **Use Cases 49**
Scoping a Requirement 50
Example Details 50
Propagating Changes Made to an Already-Approved Requirement . . . 50
Implementing a Requirement 51
Applying a SBM Workflow to Requirement Approval 51
Example Details 51
Defect and Enhancement Management 51
Example Details 52

Appendix **Event Triggers and Actions 53**
Event Triggers 54
Param Element 54
SQL Element 55
Actions 55
Create Action 56
Update Action 57

Delete Action 59
Action Fields Sent to the Other Provider. 60

Welcome

The *Serena® Dimensions® RM Integration Guide for Serena Business Manager* describes the process of using the RM-SBM integration engine to synchronize data between Dimensions RM and SBM as well as to automatically generate new SBM items in response to Dimensions RM events and new Dimensions RM objects in response to SBM 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 administrators who have extensive domain knowledge about the Dimensions RM and SBM data sources to be synchronized.

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."

Chapter 1

Overview

What Is the RM-SBM Integration?	10
Who Should Use the RM-SBM Integration?	10
What Do I Have to Install to Use the RM-SBM Integration?	10
Who Should Configure the RM-SBM Integration?	11
How Often Does the RM-SBM Integration Synchronize the Data?	11
How Does the RM-SBM Integration Work?	11
What If I Want to Use Multiple Dimensions RM Projects?	12
What If I Am Also Using Another Dimensions RM Integration?	12

What Is the RM-SBM Integration?

The RM-SBM integration is a feature of Dimensions RM that enables information to be synchronized between Dimensions RM and SBM. Specifically, you can do any of the following:

- When new objects are added to a designated collection in Dimensions RM, trigger the creation of new items in SBM.
- When existing objects in Dimensions RM are updated, trigger updates in the corresponding items in SBM.
- When existing objects in Dimensions RM are marked as deleted, transition the corresponding SBM items to an inactive state and dissolve the link between the Dimensions RM objects and the SBM items.
- Trigger the creation of new objects in Dimensions RM when new items are created in SBM.
- Trigger updates in existing objects in Dimensions RM when the corresponding items in SBM are updated.
- Mark objects in Dimensions RM as deleted when the corresponding items in SBM become inactive or are deleted.
- Specify a start date retroactively so that events in the past that have not already been processed will act as triggers.
- Images added to HTML-enabled attributes in RM are synchronized as attachments to SBM

For more information and examples of what the integration can accomplish, see "Use Cases" on page 49.

Who Should Use the RM-SBM Integration?

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

What Do I Have to Install to Use the RM-SBM Integration?

On the machine that runs Dimensions RM, a Windows service (the *Sync Engine*) for the RM-SBM integration is installed as an option (but not started) when Dimensions RM is installed. This machine must have access to the Dimensions RM database and the SBM database.

You must set up the integration so that it can access the Dimensions RM and SBM data. (See "Specifying Data Sources" on page 31.)

Who Should Configure the RM-SBM Integration?

The RM-SBM integration is installed as an option with Dimensions RM, but the integration must be configured for each company's needs.

The RM-SBM integration should be set up and administered by a user with administrative privileges.

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

The RM-SBM integration can be set to synchronize data in intervals as short as a minute, or in much longer intervals of several hours or days. For best results, do not use an interval longer than one day.

Shorter intervals keep data more up-to-date but create additional stress on system resources.

The synchronization interval is one of the settings that you adjust specifically for your site. You specify the interval in the XML configuration file (see "Specifying General Options" on page 30).

How Does the RM-SBM Integration Work?

The heart of the integration is the Dimensions RM Synchronization Engine (Sync Engine), which runs as a Windows service. Synchronization occurs at a configurable regular interval.

The integration performs actions according to a configuration file in XML format that describes how to map data between the two products. This file defines events, triggers, and actions. The integration watches for certain events, and triggers specific actions accordingly. For example, a named Dimensions RM collection can be watched for the addition of a new object to the collection. When that happens, the trigger for that event can be set to create a SBM item that is linked to the Dimensions RM object.

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

The configuration file controls the following settings:

- The accounts to use when the integration connects to Dimensions RM or SBM
- The mapping of fields from one application to the other

- The execution of actions in one application based on "trigger" events in the other (see "Specifying Events and Actions" on page 35)

What If I Want to Use Multiple Dimensions RM Projects?

If you want to exchange data between SBM and multiple Dimensions RM projects, you must tell the integration which projects you want to use and the maximum number of projects that can be open at a time. This is specified in the configuration file. See Chapter 5, "Configuring the Sync Engine" on page 29.

What If I Am Also Using Another Dimensions RM Integration?

If you are using more than one RM-SBM/RM-Dimensions CM integration (for example, one RM-SBM integration and one RM-Dimensions CM integration), you must take these extra steps:

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

Chapter 2

Setting Up Dimensions RM to Integrate with SBM

Dimensions RM Proxy Classes for SBM Items 14

Dimensions RM Proxy Classes for SBM Items

Because one Dimensions RM object (requirement) can be related to more than one SBM item, the integration includes support for a "*TT_Item*_Proxy" class in Dimensions RM. In this case, you create one proxy class for each Dimensions RM class used in the integration. An example might be a "Scope_Proxy" class to coordinate with an existing Scope class. The Sync Engine will then create one proxy object (for example, one Scope_Proxy instance) for each SBM item linked to the requirement object in Dimensions RM. The proxy object attribute value is populated based on the proxy definition (`<ProxyDef...>`) in the configuration file.

Users can then run reports on product requirements that have active SBM tasks.

In Dimensions RM, you must manually create the proxy class and the relationship between the root Dimensions RM class and the proxy class.

Although the proxy class is required only if you have a one-to-many relationship between Dimensions RM objects and SBM items, it can be helpful to use a proxy class even in a one-to-one scenario, because you can include custom fields that may not be in the root Dimensions RM class. You can then map SBM fields to these new fields.

IMPORTANT! If there is a proxy class, the Sync Engine looks there first when updating Dimensions RM with data from SBM. If the field does not exist in the proxy class, the Sync Engine looks for it in the root Dimensions RM class related to the proxy class.

To Proxy or Not to Proxy?

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.

However, using a proxy class is more complicated, and for simple one-to-one (one Dimensions RM object to one SBM item) relationships, it is not necessary.

NOTE Use of a proxy class is not supported for a scenario in which a new defect or enhancement captured in SBM triggers the creation of a new requirement in Dimensions RM.

Use of a proxy class is supported when the trigger is a Dimensions RM event and the action is to create one or more new SBM items.

Setting Up a Proxy Class in Dimensions RM

To set up a proxy class and relate it to an existing Dimensions RM class, perform the following steps:

1. Determine what values the proxy class should contain.
2. Invoke RM Manage and open the project schema editor (Class Definition tool).
3. In the project schema editor, locate the object (for example, Marketing_Requirements) for which you wish to have a proxy class.

4. Create a new class based on the Generic class type.
5. Name the class. This name is important because you will add an entry to the configuration file later for the new proxy class, and the names must match.
6. Define the proxy class by double-clicking the class object, or by selecting **Define** from the context menu.
7. Add new attributes for the values that you want to track in the proxy class. In most cases, you can use an alphanumeric value.

NOTE The proxy class should not contain any mandatory attributes other than those specified in the `ProxyDef` element in the configuration file.

8. When adding a new attribute, you provide a **Display Name** and an **Attribute Name**. (You must use the **Attribute Name** when providing a **name** value for the **Field** tags in the configuration XML file when referring to Dimensions RM attributes.)
9. Clear the **Attribute Editable** check box so that end users will not be able to modify the proxy class in Dimensions RM, and then click **OK** to accept the attribute definition.
10. If you get an error indicating that the name for the attribute must be unique, check for duplicate names in both the attribute names and display names of the other attributes.
11. Define a relationship between the proxy class and the existing requirements class. It does not matter in which direction the relationship arrow goes.
12. Name 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.
13. Save the schema. If you get an error indicating that other Dimensions RM tools must be closed before you can save the schema, close any open tools such as RM Concept and try to save the schema again.

Behavior When SBM Reads Data from Dimensions RM

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

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

Chapter 3

Setting Up SBM to Integrate with Dimensions RM

SBM Workflow Guidelines 18

SBM Workflow Guidelines

The following guidelines for setting up SBM workflows ensure that the integration performs smoothly:

1. Define or identify the fields that you want to receive values from linked objects in Dimensions RM. Consider the field types as you map the fields. SBM accepts values from the foreign data source as text and converts the value if necessary. If an incoming text string is too long, SBM truncates it. If you map a date field, the date formats must match. If you map SBM selection fields to Dimensions RM list attributes, the values must match exactly or be mapped using a `<ValueMap …>` element in the XML configuration file. (See "Specifying Value Mappings" on page 34.)

   For best results, make the fields receiving Dimensions RM data in the SBM database read-only so that browser users cannot modify those fields. They represent Dimensions RM data that normally should be changed only within Dimensions RM.

2. For each SBM project that you plan to integrate with Dimensions RM, determine the common workflow or workflows. For each, add an update transition with the name "Sync_Update". Select the **No Button On Form** check box in the **Edit Transition** dialog box so that this transition is not visible or available to browser users. For the Sync_Update transition, edit each field mapped to receive integration values that you have made read-only, and on the **Attributes** tab, select the **Allow Override** check box, and then clear the **Read Only** check box. Save changes to this transition, and repeat as necessary for additional workflows.

3. Create a user named (for example) SyncAdmin with API/script access. You can specify this user in the XML configuration file so that changes made in SBM by the Sync Engine will be associated with this user:

```
<DataSource name="TT" provider="TT">
    <Param name="user" value="SyncAdmin" />
    ...
</DataSource>
```

Chapter 4

Sample Integration

Overview	20
SBM Setup	20
Dimensions RM Setup	24
Sync Engine	25
Use Cases	25

Overview

This chapter describes an example integration using a sample SBM database, the Dimensions RM RMDEMO project, and an XML configuration file (`config_demoTT.xml`) that demonstrates the integration between the two products.

What follows are simple use cases that describe a business process that could be extrapolated into a real-world production use for the RM-SBM integration.

This sample assumes familiarity with Dimensions RM and SBM. The following setup instructions and procedures are described at a high level and do not go into painstaking (click-by-click) detail.

Supporting Files

The files listed below are relative to the Dimensions RM installation directory (for example, `C:\Program Files\Serena\Dimensions 10.1\RM`):

File	Description
`tutorial\icDemo\integrations\RTMTTSample.mdb`	SBM database
`conf\config_demoTT.xml`	XML configuration file

SBM Setup

If you are using the `RTMTTSample.mdb` database, these setup procedures have already been performed. They are detailed here so that you know exactly what was done to the original sample database. Alternatively, you could start with a copy of a production database and perform the following or similar steps as a means of setting up a real integration.

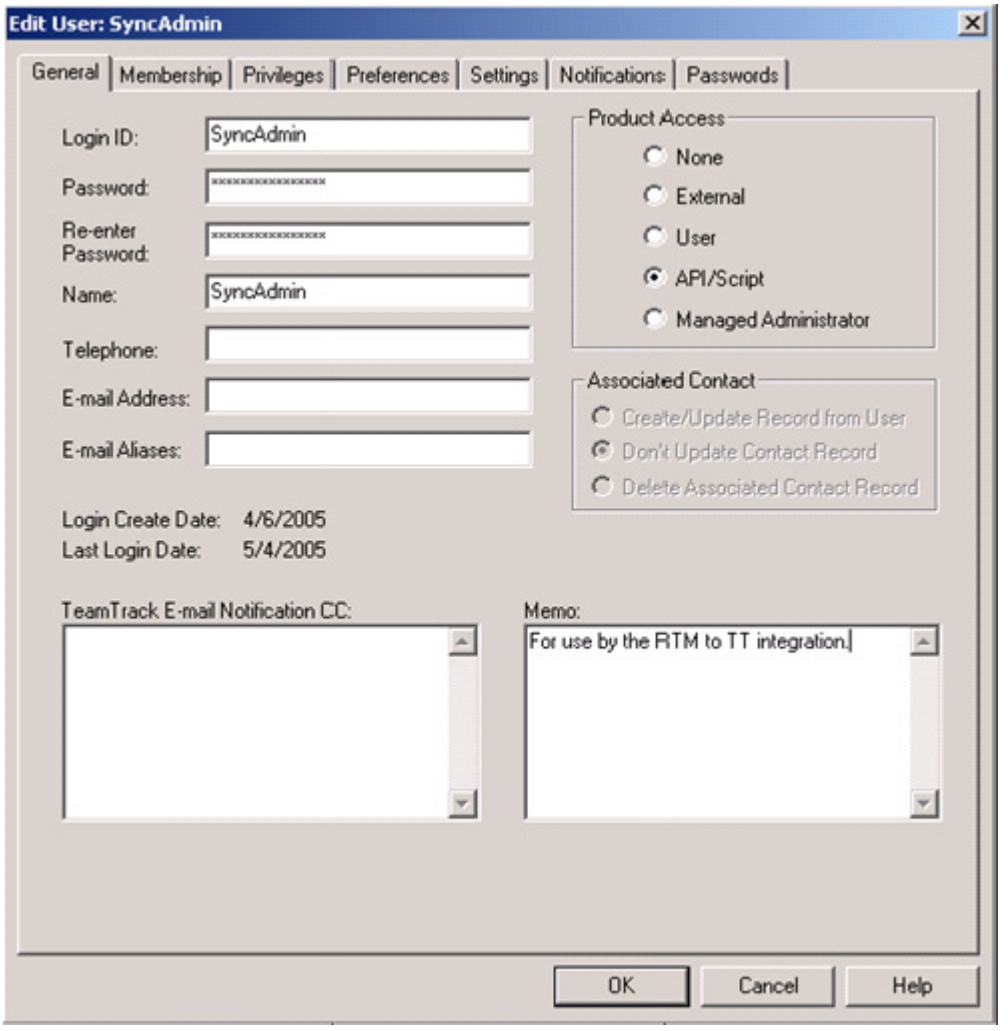
Regardless of which database you are using, configure it as an ODBC System DSN in the ODBC Data Source Administrator, which you can access via the **Connect** dialog box in the SBM Administrator client.

Connect to the data source in the Administrator client, upgrading the database if necessary.

NOTE If you want to load the sample database that has been modified for this integration and you are using IIS as your SBM Web server, you must give the IUSR_Machine user the necessary permission to access the integrations directory. For details, see the *Dimensions RM Installation Guide*.

Create "SyncAdmin" User

Create a user account with API/Script access for use by the Sync Engine:



Grant this user the system privilege to "Connect Using the API" and all Item and Field privileges for every project with which you intend to synchronize. In this example, that will be the Change Management project. Also, grant this user all privileges to the Contacts and Companies system auxiliary tables. The easiest way to do this is to make this user a member of the Administrator group.

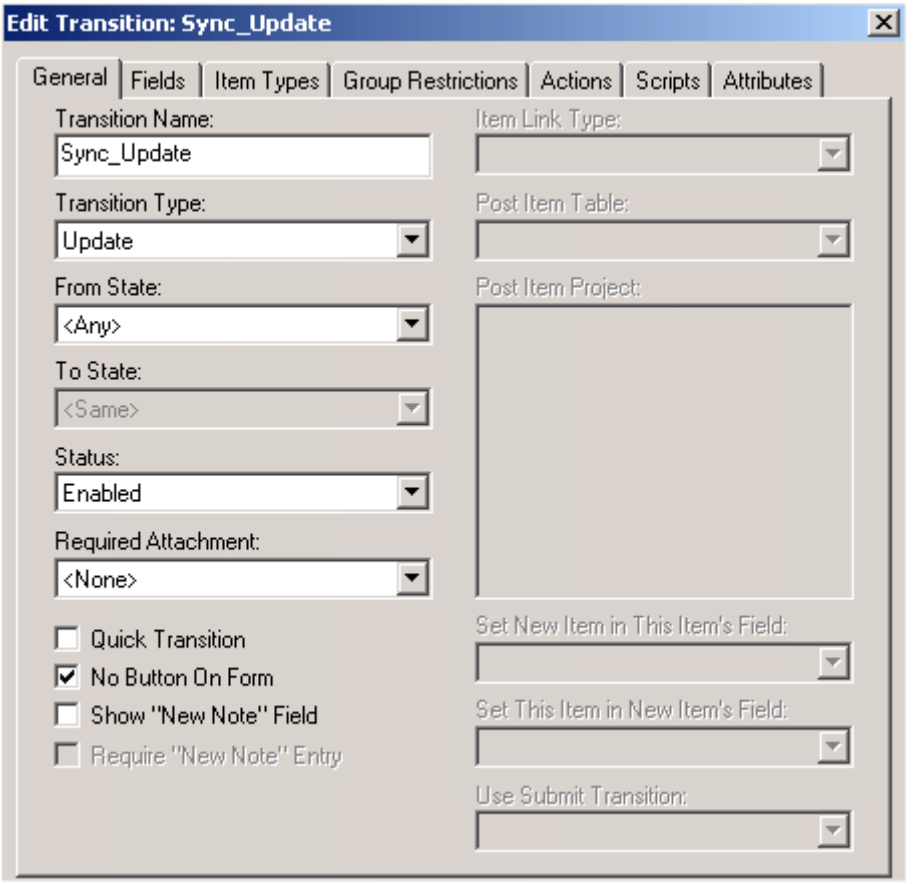
Create "Sync_Update" Transition

NOTE You must create the "Sync_Update" transition if you have any actions that do not specify transitions or any read-only fields that you want to update.

For the purpose of this example, you need to do the following only in the Change Requests workflow.

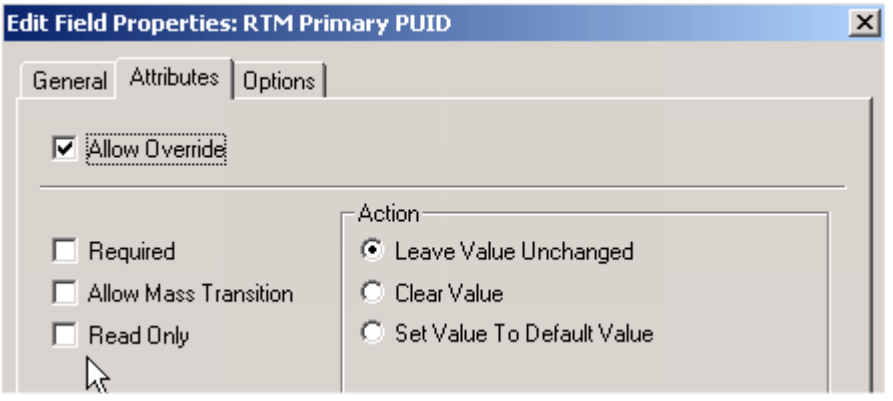
For each project with which you intend to integrate, determine the most common workflow or workflows for these projects.

For each of these workflows, add an update transition with the exact name of "Sync_Update":



Select the **No Button On Form** check box so that this transition will not be visible or available to browser users.

Next, edit the transition fields, and edit each field mapped to receive integration values that has also been made read-only. Go to the field's **Attributes** tab, select the **Allow Override** check box, and then clear the **Read Only** check box. It is important to clear the read-only check box, because if you do not clear it, the sync engine cannot write to the read-only field. This must be done for each attribute that you want to update.



Save the changes to this transition, and repeat as necessary within other workflows.

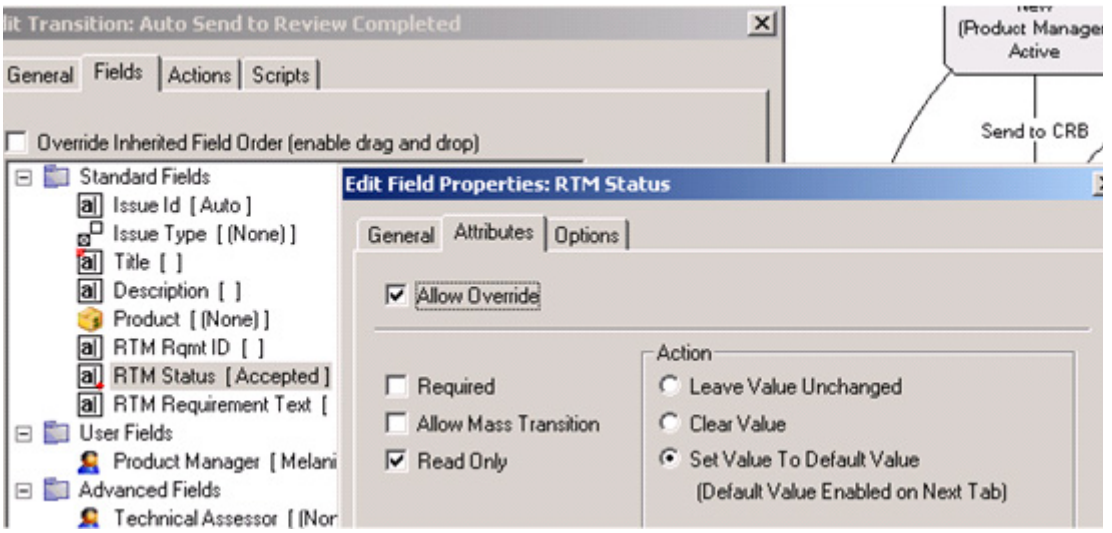
Create Fields

Create the fields referenced in the configuration file:

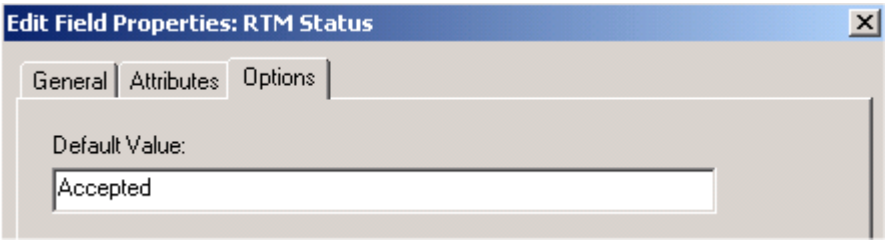
Type	Display Name	Database Name	Attributes	Options
Text	RTM Rqmt ID	RTM_RQMT_ID	Read Only	Fixed, 12 bytes
Text	RTM Status	RTM_STATUS	Read Only	Fixed, 20 bytes
Text	RTM Requirement Text	RTM_REQUIREMENT_TEXT	Read Only	Memo

Edit Transition

Edit the **Change Requests** project, edit the **Auto Send to Review Completed** transition, and then edit the **RTM Status** field. Select the **Attributes** tab, and then select the **Allow Override** check box. Select the **Set Value to Default Value** radio button.



Select the **Options** tab and enter a value of "Accepted" in the **Default Value** box:



Save all changes to the project by clicking **OK** in each nested dialog box.

Clear Host IP Address

If after starting the SBM Server, you notice that it is not listening on a valid IP address (restore the application and observe the title bar), stop the server and edit the following registry key:

`My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\TeamShare\TeamTrackDebug\WebServer`

Delete the "Host" value and restart the server. It should now generate a new, usable IP address.

Dimensions RM Setup

If you create a Dimensions RM project using the RMDEMO sample project, the following setup has already been performed. If you are using an existing project, you can perform the following steps enable this example integration within that project.

Create User

Create a Dimensions RM admin user. You can use any admin user you choose, as long as that user is specified in the `DataSource` section of the configuration file. Then, either create a SBM user with the same login name, or use the Users ValueMap to map the Dimensions RM user to a valid SBM user who has sufficient privileges to perform the desired actions. For example:

```
<ValueMap name="Users" leftsource="RTM" rightsource="TT">
    <Map left="RTMAdmin" right="bill" />
    ...
</ValueMap>
```

The RMDEMO project and the sample configuration file (`config_demoTT.xml`) shipped with Dimensions RM specify a user named "joe" with a password of "joe". In the sample SBM database, integration actions are performed as "Joe Manager".

Create Collection

Create a new collection called "Scoping".

Create Attributes

Create the attributes referenced in the configuration file in the Marketing_Requirements class, and clear the **Attribute Editable** check box.

Type	Display Name	Attribute Name	Options
A1	TT Item ID	TT_ITEM_ID	10, 10
A1	TT Item State	TT_ITEM_STATE	64, 32
A1	TT Title	TT_TITLE	80, 80

Start Oracle

Start the Oracle TNS listener service if it is not already running.

Sync Engine

Edit the Configuration File

Find the `config_demoTT.xml` file:

<RM_Install_Dir>`\conf\config_demoTT.xml`

Copy `config_demoTT.xml` to `config.xml`.

Modify the DataSource sections as needed for connectivity in your environment.

Start the Sync Engine Service

Edit the *<RM_Install_Dir>*`\conf\log4cpp.conf` file. Change the output level from WARN to INFO on line 5.

Start (or restart) the Serena SyncEngine service.

Verify that the configuration file was parsed correctly by viewing the *<RM_Install_Dir>*`\logs\synceng.log` file.

Between events triggered by the use cases listed below, you can view the progress in the log file at *<RM_Install_Dir>*`\logs\syncEngine.log`.

Use Cases

In each of the follow scenarios, after you perform the initiating actions in the user interface of one product or the other, you must wait for the Sync Engine's "sync interval" cycle so that the resulting actions have time to complete.

Scope a Requirement

1. Log in to RM Concept as "joe".
2. Open the Marketing_Requirements generated form.
3. Create a new RM object (select all, scroll to last object, click **New**). Make sure that the Delivery Phase attribute is set to TBD, and enter some text into the Text attribute.
4. Click **Update**, and then add the object to the Scoping collection.
5. Wait for the sync interval for event **(B1) RTMincluded** to fire.
6. Log in to SBM as "joe". You should see a new item with the title: "Submitted by RTM MRKT_000000 object" (000000 will reflect the true sequential requirement ID).

7 Notice that a SBM item is created and reflects the Dimensions RM attribute values.

Update a Requirement

1 In RM Concept, click **Find** to refresh, click **Select All**, and scroll to the last object to view the object that you just created, which is now populated with the values from the new SBM item.

2 Change the value of the Text attribute and click **Update**.

3 Wait for the sync interval for event **(B3) RTMchanged** to fire.

4 In SBM, refresh the report (click the Home button on the SBM toolbar), and select the linked item.

5 Notice that the changed Text attribute value is reflected in the linked SBM item's **RTM Requirement Text** field.

Remove Requirement from Collection

1 In RM Concept, make sure that **Scoping** is selected in the collection drop-down list, and then click **Remove from Collection**.

2 Wait for the sync interval for event **(B5a) RTMremovedA** to fire.

3 In SBM, refresh the report (click the Home button on the SBM toolbar), and select the linked item.

4 Notice that the SBM item is transitioned to the Closed (Inactive) state. Additionally, the cross-reference links have been removed from both the SBM and Dimensions RM databases, and further updates to either the Dimensions RM object or the SBM item will not be synchronized.

Delete a Requirement

1 In RM Concept, create another object and add it to the Scoping collection as described under "Scope a Requirement" on page 25, waiting for the sync interval for the SBM item to be created.

2 Still in RM Concept, click **Find** to refresh, click **Select All**, and scroll to the last object to view the object that you just created, which is now populated with the values from the new SBM item.

3 From the **Object** menu, select **Delete**.

4 Wait for the sync interval for event **(B7) RTMdeleted** to fire.

5 In SBM, refresh the report (click the **Home** button on the SBM toolbar), and select the linked item.

6 Notice that the SBM item is transitioned to the Closed (Inactive) state. Additionally, the cross-reference links have been removed from both the SBM and Dimensions RM databases, and further updates to either the Dimensions RM object or the SBM item will not be synchronized.

Promote Enhancement Request to Requirement

1. In SBM, submit into the Change Requests project.
2. Select an Issue Type of Enhancement Request, fill in the Title field, and then click **OK** to complete the submit.
3. Wait for the sync interval for event **(C1) TTadded** to fire.
4. Go back to RM Concept, click **Find** to refresh, click **Select All**, and scroll to the last object.
5. Notice that this new object reflects the correct values in the TT attributes.

Update the Enhancement Request

1. In SBM, update the Change Request item submitted under "Promote Enhancement Request to Requirement" on page 27.
2. Modify the **Title** field, and then click **OK** to complete the update.
3. Wait for the sync interval for event **(B2) TTchanged** to fire.
4. Go back to RM Concept, click **Find** to refresh, click **Select All**, and scroll to the last object.
5. Notice that this object reflects the updated value in the TT Title attribute.

Transition the Enhancement Request

1. In SBM, execute the **Tech Assessment** transition. In the **Technical Assessor** field, select Joe Manager, and then click **OK**.
2. Optionally, wait for the sync interval for event **(B2) TTchanged** to fire, go back to RM Concept, refresh the view of the linked object, and verify that the TT Item State attribute has changed to Technical Assessment.
3. In SBM, execute the **Assessment Complete** transition. (It is not necessary to change any field values, but you can modify the Title field if you like.) Click **OK**.
4. Optionally, wait for the sync interval for event **(B2) TTchanged** to fire again, go back to RM Concept, refresh the view of the linked object, and verify that the TT Item State attribute has changed to Assessment Complete.
5. In SBM, execute the **Send to CRB** transition. In the **Reviewers** field, choose Joe Manager, and then click **OK**.
6. Optionally, wait for the sync interval for event **(B2) TTchanged** to fire, go back to RM Concept, refresh the view of the linked object, and verify that the TT Item State attribute has changed to Change Review Board.
7. In SBM, execute the **Approve** transition. (It is not necessary to change any field values, but you can modify the **Title** field if you like.) Click **OK**.
8. Wait for the sync interval for event **(B4a) TTreviewed** to fire, go back to RM Concept, refresh the view of the linked object, and verify that the TT Item State

attribute has changed to Review Completed and the Delivery Phase attribute now has a value of Build1.

Cancel the Enhancement Request

1. In SBM, execute the **Close** transition. It is not necessary to change any field values; however, modify the Title field, and then click **OK**.
2. Wait for the sync interval for event **(B4a) TTclosed** to fire.
3. Go back to RM Concept, and from the **View** menu, clear the **Current Object Only** option. Click **Find** to refresh, click **Select All**, and scroll to the last object.
4. Notice that this object reflects the updated value in the TT Title attribute, and its **Current Status** is Deleted. Additionally, the cross-reference links have been removed from both the SBM and Dimensions RM databases, and further updates to either the Dimensions RM object or the SBM item will not be synchronized.

Delete an Enhancement Request

1. Repeat "Promote Enhancement Request to Requirement" on page 27, submitting another SBM item, which in turn creates another Dimensions RM object and links the two together.
2. Wait for the sync interval and confirm.
3. In SBM, delete the SBM item by clicking on the **Delete** transition button.
4. Wait for the sync interval for event **(B6) TTdeleted** to fire.
5. Go back to RM Concept, click **Find** to refresh, click **Select All**, and scroll to the last object.
6. Notice that the **Current Status** is Deleted, the TT Title attribute has a value of "TT Item was deleted" and the TT Item State attribute has a value of "(deleted)". Additionally, the cross-reference links have been removed from both the SBM and Dimensions RM databases, and further updates to the Dimensions RM object will not be synchronized. You cannot make any further changes to the SBM item because it has been completely deleted from the SBM database.
7. Reset RM Concept to normal behavior (from the **View** menu, select **Current Object Only**).

Chapter 5

Configuring the Sync Engine

Introduction	30
The XML Configuration File	30
Error Handling for Create, Update, and Delete Actions	39
Validating the XML Configuration File	40

Introduction

The Sync Engine runs as a Windows service, is controlled through a command-line interface (see Chapter 6, "Running the Sync Engine" on page 43), and performs its work according to the data in an XML configuration file (see below).

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

The XML Configuration File

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

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

Character Encoding and Text Editor Considerations

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

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

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

Specifying General Options

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

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

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

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

Specifying Data Source Providers

```
<Provider name="RTM" location="syncrtm.dll" description="Serena RTM
    Sync Proxy" />
<Provider name="TT" location="synctt.dll" description="Serena
    Business Mashup Sync Proxy" />
```

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

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

Specifying Data Sources

```
<DataSource name="RTM" provider="RTM">
    <Param name="user" value="RTMDEMOAdmin" />
    <Param name="password" value="password" />
    <Param name="host" value="RTM56" />
    <Param name="project" value="RTMDEMO" />
    <Param name="max_connections" value="4" />
</DataSource>

<DataSource name="TT" provider="TT">
    <Param name="user" value="SyncAdmin" />
    <Param name="password" value="password" />
    <Param name="host" value="pc2596" />
    <Param name="progid" value="TTProvider" />
    <Param name="prefix" value="RTT" />
    <Param name="start_date" value="05/26/2005" />
    <Param name="max_records" value="5001" />
    <Param name="act_as_user" value="true" />
    <Param name="admin_retry" value="true" />
    <Param name="break_lock" value="true" />
    <Param name="break_lock" value="true" />
    <Param name="proxy" value="pc2999" />
</DataSource>
```

The configuration file must specify exactly two data sources. The Sync Engine synchronizes data between only two data sources, so if there are more than two, it will report an error and fail to start up.

The `<DataSource …>` element contains connection information for a specific data source. This information is sent to the data source when the Sync Engine tries to connect.

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

NOTES:

- You can encrypt any parameter using the Sync Engine. Use the command-line option `-p` to retrieve a sample parameter with the value encrypted, and then paste that encrypted value into the configuration file. (See "Controlling the Sync Engine from the Command Line" on page 44.)
- For best results, create a special SBM user account with API/script access for use by the integration and specify that user in the SBM `<DataSource …>` element (`<Param name="user" value="SyncAdmin" />`).

Notes on Specific Attributes

- For the RTM data source, the host value is the entry name in the `tnsnames.ora` file.

  The `tnsnames.ora` file begins with a section like this:

  ```
  # NSNAMES.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_DTA =
        (SERVER = DEDICATED)
        (SERVICE_NAME = RTM)
      )
    )
  ```

  The name to use for the host attribute is the entry name, which in this example is "RTM.SERENA.com". Do not use the value labeled HOST.

  NOTE If you are unfamiliar with or do not have access to the `tnsnames.ora` file, see your Oracle administrator.

- The `max_connections` attribute for the RTM data source 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 Sync Engine uses the default value of 5. A value of -1 indicates that there is no limit.

- For the SBM data source, the `host` attribute is the IP address or DNS name of the SBM server.
- The required `progid` attribute for the SBM data source is an arbitrary name to uniquely identify this instance of the SBM integration. It can be up to 255 characters in length, but by convention is a short mixed-case alphabetic name.
- The required `prefix` attribute is an arbitrary string that in combination with `progid`, conforms to the existing SBM mechanism for supporting multiple integrations and more than one instance of a particular integration. It can be up to 16 characters in length, but by convention is a three-letter uppercase string. Both `progid` and `prefix` must be unique within the TS_TOOLS table of the SBM database.
- The optional `start_date` attribute can be used to specify a retroactive start date (see "Retroactive Triggering" below).
- The max_records attribute is optional. The default is 1000. This is the maximum number of records from the TS_CHANGES table that will be processed during a single processing of each event where SBM is the database where the event occurs.
- The `act_as_user` attribute is optional. If specified with a value of "true", actions targeting the SBM server will be performed under a named user (as opposed to the "SyncAdmin" user specified in the `user` attribute). This is recommended so that state changes and change history reflect a real user in the system for better auditing ability.
- The `admin_retry` attribute is optional. If specified with a value of "true" along with the `act_as_user` attribute, and the specified action fails to occur under the named user (typically due to insufficient privileges), the action is retried under the "SyncAdmin" user. Assuming that you have granted broad privileges to the "SyncAdmin" user, the action will likely succeed even if the named-user attempt failed. This provides a fallback mechanism under the assumption that synchronization and other integration actions are more important than user accountability.
- The `break_lock` attribute is optional. If specified with a value of "true", actions targeting the SBM server will be performed even if another user (for example, currently editing an item in the browser interface) has a lock on that item. This is the traditional and recommended approach for SBM integrations, because the browser user is informed that the lock was broken so that the user can simply resubmit any changes.
- The `proxy` attribute is optional. It allows the integration to work through firewall settings. The value must be an entity that can be resolved by a domain name server (DNS). Such entities include IP addresses, machine names, and URLs.

Retroactive Triggering

The `<DataSource …>` element is where you can specify a retroactive start date so that events in the past that have not already been processed will act as triggers:

```
<Param name="start_date" value="05/26/2005" />
```

This allows you to add to your process in stages and have newly defined triggers operate retroactively.

NOTE This works only for the SBM data source, not the RTM data source, so only SBM events can act as retroactive triggers. However, once you start using RTM 5.6 or later, events will be queued. For example, if you install RTM 5.6 into production on September 1, but do not run the Sync Engine until October 1, objects that were added to the Scoping collection during the month of September will still trigger a properly configured `included` event.

Specifying Value Mappings

```
<ValueMap name="Users" leftsource="RTM" rightsource="TT">
    <Map left="RTMDEMOAdmin" right="bill" />
    <Map left="joseph" right="joe" />
    <Map left="steve" right="stephen" />
</ValueMap>

ValueMap name="Priority" leftsource="RTM" rightsource="TT">
    <Map left="urgent" right="must" />
    <Map left="high" right="must" primary="true" />
    <Map left="medium" right="hope" />
    <Map left="low" right="wish" />
</ValueMap>
```

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

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

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

Mapping a Set of Values to a Larger Set

If you need to map a set of values to another set of values that is larger, you need to handle the situation in which a value from the smaller set is mapped to multiple values in

the larger set. To handle this problem, use the **primary** keyword to indicate which of the multiple possible values is the preferred choice:

```
<ValueMap name="UserIDs" leftSource="Test1">
    <Map left="einsteina" right="alberteinstein" primary="true"/>
    <Map left="einsteina" right="albert_e"/>
    <Map left="einsteina" right="alberte"/>
</ValueMap>
```

Specifying Proxy Definitions

If you implement your RM-SBM integration using proxy classes in Dimensions RM (see "Dimensions RM Proxy Classes for SBM Items" on page 14), you specify those proxy classes within the configuration file using `<ProxyDef …>` elements.

A `<ProxyDef …>` element contains `<Param …>` and `<Field …>` child elements. Here is an example in which the Scope_Requirement proxy class is linked via the MarketingRqmts_to_Scope relationship to the Marketing Requirements root Dimensions RM class and several field mappings are specified:

```
<ProxyDef name="Scope_Proxy" datasource="RTM" description="Test
    ProxyDef1">
        <Param class="Scope_Requirement" />
        <Param relationship="MarketingRqmts_to_Scope" />
        <Field name="TT_ITEM_ID" source="itemid" />
        <Field name="Description" source="DESCRIPTION" />
        <Field name="TIME_ESTIMATE" source="TIME_ESTIMATE" />
</ProxyDef>
```

Specifying Events and Actions

```
<Event name="RTMincluded" datasource="RTM" description="Unlinked
    object is added to a collection">
  <Trigger>
    <Condition>
      <Param event="included" />
        <Param name="collection" value="Scoping" />
        <Param name="class" value="Marketing_Requirements" />
        <Param name="project" value="RMDEMO3" />
      </Condition>
    </Trigger>

  <Create name="createTT" datasource="TT" description="Create a
    corresponding TT item">
    <Param project="Your Company Name||Issues||Change Requests" />
    <Field name="ISSUETYPE" text="Enhancement Request" />
    <Field name="TITLE" text="Submitted by RTM {0} object"
    source="PUID" />
    <Field name="RTM_RQMT_ID" source="PUID" />
    <Field name="RTM_STATUS" source="Status" />
    <Field name="RTM_REQUIREMENT_TEXT" source="Text" />
  </Create>
</Event>
```

The `<Event …>` elements contain both triggers (events) and actions to be performed when the Sync Engine detects the trigger.

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

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

Event names must be unique within the configuration file.

Events

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

NOTE In the example under "Specifying Events and Actions" on page 35, the <Condition …> element specifies a project. This is useful if you are working with multiple Dimensions RM projects.

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

The Sync Engine processes events sequentially within the configuration file.

Notes on the `ParamValue` Parameter

```
<Param name="ISSUETYPE">
    <ParamValue value="Enhancement Request"/>
    <ParamValue value="Task"/>
</Param>
```

This construct enables you to specify multiple possible values for a single parameter. The above example means that a value of either Enhancement Request or Task for the ISSUETYPE field would cause this parameter to be true.

Notes on the `excluded` Event

```
<Trigger>
    <Condition>
        <Param event="excluded" />
        <Param project="Company Name||Issues||Change Requests" />
        <Param state="New" />
    </Condition>
</Trigger>
```

Dimensions RM has the concept of a collection, from which objects can be included or excluded. In SBM terms, an exclusion event pertains to the workflow state of a SBM item. To say that a SBM item has been excluded from a state means that the item was in that state during the last synchronization cycle and has been transitioned to another state as of the current cycle.

Note that in SBM, a state change is also an update, so if you define both an `excluded` event and a `modified` event, both events act as triggers; the Sync Engine evaluates them independently.

TIP To specify multiple collections in `included` and `excluded` events, use multiple condition statements (one for each collection).

The event type is mandatory and should be specified in the first condition block.

SBM recognizes the following event types:

Event	Description
`included`	Defines an event where you want to find newly submitted or changed SBM items that could be candidates for linking to Dimensions RM objects. For example, new SBM items, or those that have moved to a particular state and meet additional criteria, will cause new Dimensions RM objects to be created with attribute values set to those of specified SBM item fields.
`excluded`	Defines an event where you want to find existing SBM items that have moved out of a particular state. If you specify a `<Param state="`*`statename`*`" />` parameter, the item must already be linked to a Dimensions RM object but must *not* be in the specified state. All other trigger conditions filter the object exactly as in the `modified` event, in that the items must match the specified criteria. For this event to work properly, the state specified in the `included` event should be the same as the state specified here. Typically, a Dimensions RM object (and a link) is created when the items move *into* state X, and the link is broken when the item move *out of* state X. **NOTE:** If you do not specify a *`statename`* parameter, the Sync Engine treats the `excluded` event the same way it treats a `modified` event.
`modified`	Defines an event where you want to find existing SBM items that have changed since the last time that this event was processed. These items, if they also meet the additional criteria, are processed, and typically the result is that their changed field values are updated in the linked Dimensions RM object.
`deleted`	Defines an event where you want to find SBM items that have been deleted since the last time this event was processed. **NOTE:** In SBM, when an item is deleted, its record is truly deleted from the primary table where it used to reside. The table ID and record ID of the deleted item is retained in the TS_CHANGES table.

Dimensions RM recognizes the following event types:

Event	Description
`included`	Defines an event where you want to find newly created requirements in Dimensions RM that are ready for scoping and can be candidates for linking to SBM items. For example, new requirements in Dimensions RM are put into a collection called *Scoping*. The collection that should be used for scoping Dimensions RM requirements should be specified in the configuration file. Putting the requirements in the collection causes new SBM items to be created with field values set to those of specified Dimensions RM attributes.
`excluded`	Defines an event where you want to find existing Dimensions RM requirements that were moved out of a particular collection. This event occurs if the requirement that was moved out of the collection has a link with a SBM item. For this event to work properly, the collection specified in the `included` event should be the same as the collection specified for the `excluded` event. This ensures that the Dimensions RM requirement is linked to a SBM item. When a Dimensions RM requirement that has a link to a SBM item, the transition specified in the configuration file is executed and the link between the Dimensions RM requirement and the SBM item is broken.
`modified`	Defines an event where you want to find existing Dimensions RM requirements that are linked to SBM items and have changed since the last time that this event was processed. If these requirements meet additional criteria, they are processed. Typically the result is that their changed attribute values are updated in the linked SBM item.
`deleted`	Defines an event where you want to find Dimensions RM requirements that were deleted since the last time this event was processed. Typically, this event causes the links between Dimensions RM requirements and SBM items to break.

Actions

There are exactly three types of action: `<Create …>`, `<Update …>`, and `<Delete …>`. The actions specify the data source that is to accept the change triggered by an event in the other data source as well as the fields and parameters necessary to perform the action. The action block also specifies a name and description, which are both optional and used for logging purposes.

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

If an action element specifies a project (`<Param project="RMDEMO3" />`),it `overrides` 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 element.

An action element (`<Create …>`, `<Update …>`, or `<Delete …>`) contains both `<Param …>` and `<Field …>` child elements.

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 will be interpreted as a literal value for the field.

A field may also contain data from the event object (the Dimensions RM object or SBM item 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="PUID" />
```

The use of the `{0}` token in the text attribute value creates a composite value (for example, "`RTM Issue MRKT_0009`"). The `{0}` token is replaced with the current value of the PUID field.

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.

Error Handling for Create, Update, and Delete Actions

If the RM-SBM 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 that 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 in the other application.

If the Create action fails, the Sync Engine reports an error and attempts to delete new items and links created in the steps outlined previously. It does this as follows:

- In Dimensions RM, the integration marks orphaned items or links for deletion.
- In SBM, if the Sync Engine cannot complete the Create action when creating a new proxy object, the created item remains. If the item appears as a duplicate after the next Event processing run, you must manually delete it.

NOTE If you add a SBM system field with "auto" behavior to the configuration file, SBM replaces data from Dimensions RM with system data. For example, the SBM **Owner** field is populated by SBM based on the value specified for each state. If you map that field to Dimensions RM, SBM will replace the value from Dimensions RM with the correct owner in SBM to prevent the integration from ignoring workflow behavior set up in SBM.

Examples of "auto" system fields are **Owner**, **Submitter**, **Secondary Owner**, **Project**, and **Submit Date**.

Error Handling for Update Actions

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

NOTE If you update a SBM system field with "auto" behavior in the configuration file, SBM replaces data from Dimensions RM with system data. For example, the SBM **Owner** field is populated by SBM based on the value specified for each state. If you mapped that field to Dimensions RM, SBM will replace the value from Dimensions RM with the correct owner in SBM to prevent the integration from ignoring workflow behavior set up in SBM.

Examples of "auto" system fields are **Owner**, **Submitter**, **Secondary Owner**, **Project**, and **Submit Date**.

Error Handling for Delete Actions

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

If the Delete action fails, the integration reports an error but does 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 in the interim, no additional error is reported when the Delete action is retried.

Validating the XML Configuration File

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

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

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

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

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

The file named `SyncEngine.xsd` is the XML schema file used for the integration. By default, this file is located in the *<RM_Install_Dir>*`\conf` directory.

The XML parser in the RM-SBM integration will attempt to validate the XML configuration file against the XML schema.

Possible Errors

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

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

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

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

- The data source referenced in a ProxyDef or action (for example, Create/Update/Delete) is not valid

  This refers to the DataSource section at the beginning of the XML configuration file. Check this section to be sure that you are using the correct name for the data source.

- The leftsource/rightsource in a ValueMap is not valid

  Check to make sure that the leftsource matches one of the two data sources. If you specified the rightsource, it must match as well.

- The provider referenced in a DataSource is not valid

  Check the Provider section to be sure that you are using the correct name for the provider and for the DLL.

- The data source referenced in an Event is not valid

  Check the DataSource section at the beginning of the XML configuration file to be sure that you are using the correct name for the data source.

Inspect the configuration file if this error message appears.

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

This error generally indicates one of the following:

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

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

```
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
Marketing_Requirements_Bad_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.

Chapter 6

Running the Sync Engine

Controlling the Sync Engine from the Command Line 44

Logging 45

Controlling the Sync Engine from the Command Line

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

Usage

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

Options

Option	Description
`-c`	Checks (validates) the XML configuration file against the associated schema document. The schema document is specified in the second line of the XML configuration file. After validating the file and logging any errors, the Sync Engine quits without any further processing.
`-e` *`level`*	Shows startup errors of the following levels: debug, info, warn, error.
`-E` *`file`*	Logs startup errors to a file.
`-f` *`file`*	Specifies an alternative configuration file.
`-h`	Lists the available command-line options.
`-k install`	Installs a Sync Engine service.
`-k config`	Changes the location in which the Sync Engine looks for the XML configuration file. Use this option in conjunction with the `-f` option (for example, `syncengine -k config -f` *`filePath`*).
`-k uninstall`	Uninstalls the Sync Engine service.
`-k start`	Starts the Sync Engine service.
`-k stop`	Stops the Sync Engine service.
`-L` *`priority`*	Sets the Windows system priority for the Sync Engine service. *`servicePriority`* must be `LOW`, `BELOWNORMAL`, `NORMAL`, `ABOVENORMAL`, `HIGH`, or `REALTIME`. If you do not specify this option, the Sync Engine service runs under the default priority of `BELOWNORMAL`.
`-n` *`serviceName`*	Sets the service name and specifies that the corresponding configuration file will be used.
`-p` *`password`*	Encrypts the password text for use in the configuration file. **NOTE:** You must paste the output from this command into the XML configuration file within the appropriate `<DataSource>` element.

Option	Description
`-P` *`eventName`*	Purges the queue containing data to be processed of all objects associated with the named event. Use this when orphaned objects (for example, from a disconnection) are beginning to affect performance. You can remove objects associated with multiple events by using the `-P` option more than once: `"-P RTMIncluded -P RTMExcluded"`. If the event named 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.

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

Installing the Sync Engine Service

To install the Sync Engine service, use the `-k install` option. This creates an instance of the service with all the other parameters that you specify.

NOTE For a multiple-integration environment (for example, if you are integrating Dimensions RM with both SBM and Dimensions CM), you must have multiple instances of the Sync Engine service installed, with a separate configuration file for each.

You can rename the services and the configuration files for clarity. Install each Sync Engine service using the `-f` option (`-k install -f` *`configFilePath`*) to specify the appropriate configuration file.

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.

Logging

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

NOTE The file named `synceng.log` contains startup information. This is what happens during the parsing of the XML configuration file and other activities by the Sync Engine prior to starting an event cycle.

This file contains information about the event cycles themselves. This is the file that you will check the most often.

The Logging Configuration File

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

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

The `log4cpp.conf` file contains five categories:

- fileBrowser
- fileSyncEngine
- fileWebService
- fileDbDoctor
- fileAlfEventEmitter

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

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

You can alter the name of the log file.

To control the size of the log file, see "Controlling Log File Size" below.

The layout of the output of the log file is controlled by the `log4j.appender.file<`*category*`>.layout` settings, which use standard log4j formatting.

You should not change anything else in this file.

Controlling Log File Size

There are two possible settings for `log4jappender.file<`*category*`>`, which controls how information gets appended to the log. The simplest configuration allows the log file to grow indefinitely:

```
log4j.appender.file<category>=org.apache.log4j.FileAppender
```

However, it is more likely that you want to limit the maximum size of the log file so that it does not consume excessive system resources. You accomplish this by specifying a "rolling file appender", which starts a new file (such as `syncEngine.log2`) after the first file exceeds the specified size limit. In the following example, the maximum file size is

5,000,000 bytes, and no more than three (the most recent three) log files will be retained at any 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.

Chapter 7

Use Cases

This chapter contains a few example use cases of the integration between Serena® Dimensions® RM and Serena® Business Manager®. These use cases demonstrate how the integration extends application life-cycle management.

NOTE To implement these use cases, you need to make the appropriate entries and adjustments in your XML configuration file.

For the location of a sample configuration file, and for a complete sample integration, see Chapter 4, "Sample Integration" on page 19.

Scoping a Requirement	50
Implementing a Requirement	51
Applying a SBM Workflow to Requirement Approval	51
Defect and Enhancement Management	51

Scoping a Requirement

Requirements can participate in several different processes or workflows as they move from inception to completion. For example, after a requirement is entered into Dimensions RM, the requirement may require input from multiple team members using SBM. This might be an important step in the process of scoping and weighing the requirement.

In this scenario, each team member would receive a SBM task requesting additional information. After this information is entered into SBM, it would return to Dimensions RM for further processing. An "approval" phase may require feedback from a group of users all responding to a single SBM task. These separate workflow states would be visible to the Dimensions RM user.

Example Details

The Dimensions RM requirement needs to be reviewed by development for initial scoping. Within Dimensions RM, it is assigned to a special "scoping" collection. This triggers the creation of a task in SBM. The SBM task has appropriate requirement information (read only). The Dimensions RM user can see the SBM information in Dimensions RM. This information is periodically synchronized by the Sync Engine.

When the SBM task transitions, SBM sends an e-mail message to the designated users, and the Dimensions RM requirement attribute "Scoped" is changed to "Complete".

The Dimensions RM user adds a requirement (an object of a particular class) to a collection called "Need Scoping". Based on the content of the integration configuration file, a new scoping task is created in the appropriate SBM project. A new object of a special administrator-configured SBM proxy class is created in Dimensions RM to store the SBM item information. The SBM item information in the Dimensions RM proxy object is read-only. The content of this item information and the time interval at which it is updated are based on settings in the integration's XML configuration file.

The item in the SBM proxy class can then be part of a SBM workflow process for necessary approvals and additional workflow transitions. The Dimensions RM user can run a query (script) to check on the state of the scoping tasks.

In Dimensions RM, the SBM proxy class is related to the original Dimensions RM requirement class. A particular SBM item is linked to the original requirement enabling all types of queries to be run in Dimensions RM.

NOTE Based on `<ProxyDef ...>` definitions (and references to them) in the XML configuration file, a Dimensions RM object is linked to one or more SBM items, provided that the proxy definition is referenced in a SBM `Create` action block.

If only one SBM item is mapped to the Dimensions RM object, a proxy class is not strictly required.

Propagating Changes Made to an Already-Approved Requirement

When changes are made to an "approved" Dimensions RM requirement, the related SBM tasks may need to be automatically updated. The integration will notify the user using the

SBM e-mail notification feature and modify the status of the created tasks to reflect the change. Your business process should define what behavior is best in this situation.

Implementing a Requirement

This use case is similar to "Scoping a Requirement" on page 50, but here, instead of soliciting information for scoping a requirement, you generate SBM tasks in order to implement a requirement that has already been scoped and agreed upon.

In this scenario, each team member (from various disciplines such as Development, Documentation, and QA) would receive a SBM task of the appropriate item type. After this information is entered into SBM, it would return to Dimensions RM for further processing. The technical aspects of the example details would be similar to those in the earlier section on scoping a requirement.

Applying a SBM Workflow 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 rich workflow support in SBM. A Dimensions RM requirement's lifecycle can be executed by SBM to provide the extended workflow support.

Example Details

You can add synchronization rules in the integration's XML configuration file to create SBM approval tickets and return the workflow results to the Dimensions RM requirement's objects (mapped in the XML configuration file).

If a formal approval process is needed for certain requirements, they are added to a special collection. Based on the integration configuration file, the SBM task with the appropriate workflow is created. While the SBM task goes through its process, the Dimensions RM user can see the states in Dimensions RM by running reports. When the SBM task is approved, the Dimensions RM requirement "approval status" attribute is changed to "approved" based on configuration file mapping. The notification system within SBM can be used to notify users that the requirement has been approved.

Defect and Enhancement Management

As defects and enhancements are entered into SBM, it may be necessary to automatically move these items from SBM 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 a Change Control Review Board meeting).

Example Details

After a new defect or enhancement is captured in SBM, a new requirement will be created in Dimensions RM. This means that in Dimensions RM, a new object of a class specified in the integration configuration will be created.

IMPORTANT! Use of a proxy class is not supported in this scenario.

The new requirement will become a regular object of the Dimensions RM class, mapped in the configuration file from SBM to Dimensions RM. The information in both the Dimensions RM object and the corresponding SBM item will be periodically synchronized, based upon the settings in the integration's XML configuration file.

When a requirement is approved in Dimensions RM, the corresponding SBM defect or enhancement is transitioned to a "Get to Work" state. The approval of the Dimensions RM requirement is indicated by the attribute change or by being assigned to a special collection as specified in the integration configuration file.

Note that the Dimensions RM user can find the SBM defect or enhancement information by running a report in Dimensions RM.

Appendix

Event Triggers and Actions

This appendix covers event-related XML elements, attributes, and values not covered in Chapter 5, "Configuring the Sync Engine" on page 29.

Event Triggers	54
Actions	55

Event Triggers

Param Element

table Parameter

```
<Param table="TTT_ISSUES" />
```

Either `table` or `project` must be specified in the first condition block. If `table` is specified again in subsequent condition blocks, its value is ignored. The value is the database name of the primary table (including the prefix) in which the items you want to query reside.

Both `table` and project could be specified, but that would be redundant and error prone. The primary *table* can be derived from the *project*, so the two must not conflict.

project Parameter

```
<Param project="Your Company Name||Issues||Change Management" />
```

Candidate items come from the specified project. If `table` is not specified, then `project` must be. The primary table is derived from the project. Project is more specific and therefore recommended for candidate items and `modified` events. In the case of `deleted` events, specifying the table could be considered better since it encompasses more potential candidates with a single event, and the project is no longer applicable for deleted items anyway.

The value is the full path to the project. Project names must be separated by two vertical bar characters and entered exactly as defined with respect to case and spaces.

state Parameter

```
<Param state="Approved" />
```

Candidate items currently reside in or have just transitioned to the workflow state specified. This parameter is optional.

The value is the state name. Be aware that SBM does not enforce unique state names. If this presents a problem for the integration, request that the SBM administrator edit the workflow state names so that they are unique.

name, value Parameter

```
<Param name="ISSUETYPE" value="Enhancement Request" />
```

This parameter specifies a field in the primary table (the workflow upon which the project is based) and the value of that field required for the item to qualify.

The `name` attribute must be the database name of the field. It is the column name, all upper case characters, without the "TS_" prefix. You can also obtain it by viewing the **Database Field Name** control on the General tab of the **Add/Edit field** dialog box in the SBM Administrator client. This name is used instead of the "Logical Field Name", because it cannot be changed once the field is created.

The value attribute is the display value of the field as it would appear to a browser user. It is also case-sensitive.

If a field has more than one qualifying value, you can specify this as follows:

```
<Param name="ISSUETYPE">
    <ParamValue value="Enhancement Request" />
    <ParamValue value="Task" />
</Param>
```

This indicates that ISSUETYPE can have the value of either "Enhancement Request" or "Task". Note that this alternate syntax applies only to field parameters in Condition blocks. It cannot be used in Actions or anywhere else.

SQL Element

The SQL element allows you to specify an SQL "where" clause that otherwise cannot be generated using the mechanisms described previously. It must be valid SQL for the schema of your SBM database and the managing DBMS. Any field must be referenced by the full column name, and values must be as they are stored in those columns (they are typically internal ID values). The resulting SQL is likely to be rather cryptic, so it is recommended that you employ this mechanism only when absolutely necessary.

The SQL must be enclosed in a CDATA block to reduce the possibility of invalidating the XML:

```
<SQL><![CDATA[TS_PRODUCT in (4,5)]]></SQL>
```

In this example derived from the standard SBM sample database, we are requesting that, in addition to other conditions, candidates have the value of the Product single-relational field. The numbers specified in the "in" clause are the TS_IDs of these records in the TTS_PRODUCTS table.

Actions

There are three action elements recognized by the TT Provider: Create, Update, and Delete. Each action element has two attributes.

The `datasource` attribute value indicates the data source upon which the action is performed.

The `description` attribute is arbitrary text that you can use to describe the action. The value might be written to the log file, but it is not otherwise used.

The parameters required for each action differ slightly, so an example of each follows.

Create Action

```
<Create name="createTT" datasource="TT" description="Submit new TT
    item" proxy="TT_Item_Proxy">
    <Param project="Your Company Name||Issues||Change Management" />
    <Param transition="Special Submit" />
    <Field name="TITLE" text="Submitted by integration" />
    <Field name="ISSUETYPE" text="Change Request" />
    <Field name="RTM_REQUIREMENT_TEXT" source="Text" />
    <Field name="RTM_STATUS" source="Current Status" />
</Create>
```

Create Element

name attribute

The `name` attribute is the name of the `Create` action.

datasource attribute

The `datasource` attribute must be the the SBM datasource as defined in the configuration file. This attribute is used when the event is triggered by SBM. The value of this attribute is `TT` when the event is triggered by SBM and `RTM` when the event is triggered by Dimensions RM.

description attribute

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

proxy attribute

The `proxy` attribute is optional and if specified, must be a proxy definition. The proxy definition is defined using the `<ProxyDef...>` tag in the configuration file. This attribute is normally used when more than one SBM items need to be created for a single Dimensions RM object. If only one SBM item is mapped to the Dimensions RM object, a proxy attribute is not strictly required.

NOTE For an example of how to create more than one SBM items, see the `config_demoTT_withProxy.xml` file. This file is located in the `<`*RM Install Dir*`>\conf` folder.

Param Element

The Param element contains a single attribute and a value that has special meaning and logic within the TT Provider DLL. Typically, these special parameters are used to specify the project to submit into and an optional alternate submit transition.

project parameter

```
<Param project="Your Company Name||Issues||Change Management" />
```

Exactly one `project` parameter is required. This is the project into which the new item will be submitted.

The value is the full path to the project. Project names must be separated by two vertical bar characters and entered exactly as defined with respect to case and spaces.

transition parameter

```
<Param transition="Special Submit" />
```

The `transition` parameter is optional. If it is not supplied, the first non-hidden submit transition is used (the same one invoked for browser users). Otherwise, the first hidden submit transition is used (**No Button On Form** is enabled in its definition). If the `transition` parameter is supplied, this must be the name of a valid submit transition for the project. Alternate submit transitions can be defined in the SBM Administrator client to supply default values for required fields, thus ensuring that the submit succeeds.

Submit transition names are unique within a given project, so normally there is no ambiguity. However, if the SBM administrator changes the transition name, this parameter may result in an error.

Field Element

The Field element contains two required attributes and one optional attribute and their associated values: the name of the field receiving the new value, the literal or indirect value itself, and the name of a destination datasource. Each field element will have a `name` attribute and either a `text` or `source` attribute. Each field element can have an optional `datasource` attribute.

name attribute

The `name` attribute is the database name of the SBM field that is to receive the value. It must be all upper case and exactly match the database name of the field.

text attribute

```
<Field name="ISSUETYPE" text="Enhancement Request" />
```

The `text` attribute indicates the literal value to be set in the field.

If the target field is a selection field type, the value must exactly match a selection valid for the project.

source attribute

```
<Field name="RTM_STATUS" source="Verification Status" />
```

The `source` attribute indicates the name of the data element that supplies the actual value sent to the field. This data is usually obtained from items or objects returned by the event trigger. It might have been mapped by the Sync Engine.

In this example, we are setting the SBM field RTM_STATUS with the value of the Dimensions RM attribute "Verification Status".

If the target field is a selection field type, the value must exactly match a selection valid for the project.

datasource attribute

```
<Field name="TT_Item_ID" source="itemid" datasource="RTM" />
```

The `included` event identifies issues or change documents that were newly added by the Sync Engine as well as by the end user. SBM properly identifies issues that were newly added by the Sync Engine; however, it does not identify the issues that were added by the Sync Engine using the `included` event.

The `datasource` attribute specifies the destination datasource and allows information to be transferred back to the originating object.

Update Action

```
<Update name="updateTT" datasource="TT" description="Synchronize TT
    item">
    <Param project="Your Company Name||Issues||Change Management" />
    <Param itemtype="Enhancement Request" />
    <Param transition="Approve" />
    <Field name="RTM_PUID" source="PUID" />
    <Field name="RTM_REQUIREMENT_TEXT" source="Text" />
    <Field name="RTM_STATUS" source="Current Status" />
</Update>
```

Param Element

`Param` elements limit Update actions to specific groups of SBM items or Dimensions RM requirements. Each `Param` element is optional.

project parameter

`<Param project="`*YourCompany Name*`||Issues||Change Management" />` (SBM)

`<Param project="RMDEMO" />` (Dimensions RM)

The `project` parameter is optional. If it is supplied, the update is performed only if the item is in the specified project.

The `project` parameter applies to "TT" and "RTM" datasources.

itemtype parameter

<Param itemtype="Enhancement Request" /> (SBM)

The `itemtype` parameter is optional. If it is supplied, the update is performed only if the item is of the specified item type. In this example, the ISSUETYPE field must have the value of "Enhancement Request".

The `itemtype` parameter applies to "TT" datasources.

transition parameter

`<Param transition="Approve" />`

The `transition` parameter is optional. If it is not supplied, the first non-hidden (browser-available) update transition is used. Otherwise, the first hidden update transition is used (**No Button On Form** is enabled in its definition). If the `transition` parameter is supplied, this must be the name of a transition valid for the item in its current state.

Transition names are not necessarily unique within a given project. If there is more than one valid transition with the specified name, transitions are searched in order as defined by the state (**Transition Order** tab of the Add/Edit State dialog box in the SBM Administrator workflow editor). However, precedence is given to a transition that has its 'From State' explicitly defined as the state the item is currently in. Otherwise, 'From State:' *<Any>* is accepted.

The `transition` parameter applies to "TT" datasources.

class parameter

`<Param class="Marketing Requirements" />`

The class parameter update is performed only if the requirement is in the specified class. In this example, the Update action only updates requirements that are in the Marketing Requirements class.

The `class` parameter applies to "RTM" datasources.

NOTE You cannot put a requirement into a collection using an Update action. You must use a Create action to do this.

Field Element

The Field element contains two attributes and their associated values: the name of the field receiving the new value, and the literal or indirect value itself. Each field element will have a `name` attribute and either a `text` or `source` attribute.

name attribute

The `name` attribute is the database name of the SBM field that is to receive the value. It must be all upper case and exactly match the database name of the field.

text attribute

```
<Field name="UPDATE_PERFORMED_BY" text="RTM" />
```

The `text` attribute indicates the literal value to be set in the field.

If the target field is a selection field type, the value must exactly match a selection valid for the project.

This is not a common use case but could have value. Presumably, an initial value was supplied when the SBM item was first submitted. Now, when it is updated or transitioned, you might want to update the field with a new literal value. For example, you could explicitly set the value of a SBM field to indicate that the update was performed indirectly by the Dimensions RM integration.

source attribute

```
<Field name="RTM_STATUS" source="Verification Status" />
```

The `source` attribute indicates the name of the data element that supplies the actual value sent to the field. This data is usually obtained from items or objects returned by the event trigger. It might have been mapped by the Sync Engine.

In this example, we are synchronizing the SBM field RTM_STATUS with the current value of the Dimensions RM attribute "Verification Status".

If the target field is a selection field type, the value must exactly match a selection valid for the project.

Delete Action

In the case of the TT Provider, a delete action is nothing more than deleting the cross-reference records associating the SBM item with the linked object in the other (RTM) provider. No parameters or fields need be specified in the configuration file.

This action makes no sense as a result of a `submit` event type, because the cross-reference does not yet exist.

There are two useful ways in which this action can be invoked, as described below.

SBM Delete Event

```
<Event name="deleted" datasource="TT" description="Find Deleted TT
    Items">
    <Trigger>
        <Condition>
            <Param type="delete" />
            <Param table="TTT_ISSUES" />
        </Condition>
    </Trigger>

    <Delete name="delete" datasource="RTM" description="Mark the RTM
    object as deleted">
    </Delete>
</Event>
```

In this case, the event specifies TT as the data source and the `type` parameter is set to "delete". This example queries TT for literally deleted TT items in the Issues table. The result set gives us only the record IDs of items recently deleted form the Issues table.

Action Fields Sent to the Other Provider

When the action specifies the other data source as the target of the action, Field elements with the source attribute are in effect asking SBM to supply the value for the destination attribute. All the following are optional of course, but necessary if you want to synchronize the specified data.

```
Update datasource="RTM" description="Update RTM object with SBM
    data">
    <Param type="RTM_REQS" />
    <Field name="TT Full Project Path" source="project" />
    <Field name="TT Leaf Project Name" source="PROJECT" />
    <Field name="TT_ITEMID" source="itemid" />
    <Field name="TT_TITLE" source="TITLE" />
    <Field name="TT_OWNER" source="OWNER" />
    <Field name="UPDATE_PERFORMED_BY" text="RTM" />
</Update>
```

The `<Param type="RTM_REQS" />` line has meaning only for the RTM data source.

The `text` attribute in the last Field element specifies that the literal string "RTM" be inserted into the UPDATE_PERFORMED_BY attribute in the updated Dimensions RM object. It has no interaction with SBM.

source="<SPECIAL_VALUE>"

If the value of the `source` attribute contains no upper-case characters, it is assumed to refer to a special output value. This mechanism is typically used for system fields and contrived data. There is a fixed set of special output values recognized and returned by the TT Provider. Each of these special fields (reserved words) is described below.

project

There are two ways that you can request the project name from SBM and have the value sent to the other data source provider. If you want the full project path, you must request it using the special project field source:

```
<Field name="TT Full Project Path" source="project" />
```

This example would send the following text string:

Your Company Name||Issues||Change Management

However each primary table contains a system field for the project name. If you simply want the leaf name of the project, you can request it as a database field name as follows:

```
<Field name="RTM_SERVER_NAME" source="PROJECTID" />
```

For the same item, this usage would return the following string:

```
Change Management
```

itemid

The itemid is the display item ID text as shown to the SBM user in the browser. This is the common means of identifying an item in SBM and comprises the Item Type and the Item ID sequence number. Because this comprises two field values, this special Field element is provided:

```
Field name="TT_ITEMID" source="itemid" />
```

This example would return the following text:

```
ENH00123
```

source="<FIELD_NAME>"

If the value of the `source` attribute contains no lower-case characters, it is assumed to refer to a SBM database field name. Once again, SBM field names are the database name as shown in the **Database Field Name** control on the **General** tab of the **Add/Edit field** dialog box in the SBM Administrator client. Here is an example:

```
<Field name="TT_TITLE" source="TITLE" />
```

This returns the Title field of the item. For example:

```
Create marquee/scrolling line around pasted image
```

Here is another example:

```
<Field name="TT_OWNER" source="OWNER" />
```

This returns the Login ID of the current owner of the SBM item (for example, `joe`).