Dimensions® RM 12.6

Integration Guide for Atlassian Jira

Publication date: March 2018

Contents

Preface

This document describes the Dimensions RM integration with Atlassian Jira.

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

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

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

Objective

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

Audience

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

Manual Organization

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

Contacting Serena Technical Support

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

http://supportline.microfocus.com

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

You can use the Serena Support Web page to:

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

License and Copyright Information for Third-Party Software

License and copyright information for third-party software included in this release can be found as part of the software download available at:

http://support.serena.com/Download/Default.aspx

Chapter 1

Introduction

What Is the RM-Jira Integration?

The RM-Jira integration enables information to be synchronized between Dimensions RM and Atlassian Jira. 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 Jira.
- Trigger the creation of new objects in Dimensions RM when new items are created in Jira.
- Trigger updates in existing objects in Dimensions RM when the corresponding items in Jira are updated.

Who Should Use the RM-Jira Integration?

The RM-Jira integration is for Dimensions RM or Jira 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 Jira items. Engineering can then view and use (but not edit) the information in those fields as they work on the project.

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

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

NOTE The system on which Dimensions RM is installed must have access to Jira. See "Specifying the Jira Data Source" on page 24.

Who Should Configure the RM-Jira Integration?

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

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

The RM-Jira 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 23.

How Does the RM-Jira Integration Work?

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

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

API Usage and Limitations

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

Configuration File Usage

The configuration file controls the following conditions:

- The execution of actions in Jira based on "trigger" events in RM, or the execution of actions in Dimensions RM based on "trigger" events in Jira. See "Specifying Events and Actions" on page 28.
- The set of classes in Dimensions RM that are monitored for changes.

What If I Am Also Using Another Dimensions RM Integration?

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

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

Chapter 2

Use Cases

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

Scoping a Requirement

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

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

Requirement Change (RM -> Jira)

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

Defect Detection (Jira -> RM)

1. Running the tests in Jira yields a defect, because of a failed test.
2. A new defect is created in Jira and triggers a configured Sync Engine event.
3. The RM-Jira integration retrieves the data for the defect object.
4. When the configured conditions are met, a corresponding object is created in Dimensions RM.
5. Periodically, the Sync Engine checks for updates and transfers the changes to Dimensions RM.
6. A defect is detected in Jira. The transition rules specified in the Sync Engine configuration define a create event in Dimensions RM. This event creates a new object in the class Defect.

Chapter 3

Setting Up Dimensions RM

Overview

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

RM Database Setup Steps

Before you can run the Sync Engine with Jira, you must add an entry with the URL to the database table PROJECT_OPTIONS. The URL is required for linking objects between Dimensions RM and Jira.

NOTE If you change protocol (http or https), server name or port at a later time, you must update the URL setting in the database. If you do not update the URL setting, links from a Jira item to an RM object will point to the wrong location.

Verifying if the URL is already in table PROJECT_OPTIONS

To verify if the URL is in table PROJECT_OPTIONS, do the following:

1. Open a command prompt.
2. Type `sqlplus` and press **Enter**.
3. Enter the user name and press **Enter**. The user name should be in this format *RM_INSTANCE@DATABASE*, e.g. RMDEMO@ORCL.
4. Enter the password and press **Enter**.
5. Type: `select * from PROJECT_OPTIONS where TOOL='SYNC' AND USERNAME='ANY' AND NAME='RTM_URL';`
6. Press **Enter**.
7. If the URL is not in the table, continue with step 5 in section "Adding the URL to table PROJECT_OPTIONS" on page 14.
   If the URL is already in the table, continue with step 5 in section "Updating the URL in table PROJECT_OPTIONS" on page 15.

Adding the URL to table PROJECT_OPTIONS

To add the URL to the table PROJECT_OPTIONS, follow these steps:

1. Open a command prompt.
2. Type `sqlplus` and press **Enter**.
3. Enter the user name and press **Enter**. The user name should be in this format *RM_INSTANCE@DATABASE*, e.g. RMDEMO@ORCL.

4 Enter the password and press **Enter**.

5 Type: `insert into PROJECT_OPTIONS values ('SYNC', 'ANY', 'RTM_URL', 'http://RM_SERVER:PORT');`

NOTE

- When using SSL, replace **http** with **https**.
- Replace **RM_SERVER** and **PORT** with server name and port matching your environment.

6 Press **Enter**.

Updating the URL in table PROJECT_OPTIONS

To update the URL to the table PROJECT_OPTIONS, follow these steps:

1 Open a command prompt.

2 Type `sqlplus` and press **Enter**.

3 Enter the user name and press **Enter**. The user name should be in this format *RM_INSTANCE@DATABASE*, e.g. RMDEMO@ORCL.

4 Enter the password and press **Enter**.

5 Type: `update PROJECT_OPTIONS set VALUE='http://RM_SERVER:PORT' where TOOL='SYNC' and USERNAME='ANY' and NAME='RTM_URL';`

NOTE

- When using SSL, replace **http** with **https**.
- Replace **RM_SERVER** and **PORT** with server name and port matching your environment.

6 Press **Enter**.

Adding Classes

You must add classes to Dimensions RM, which then hold the data transferred from Jira. The types and number of Dimensions RM classes depend on how you plan to store information in Dimensions RM.

Adding Attributes

For each Dimensions RM class, add the necessary attributes to correlate with the fields in Jira. For example, when mapping to the Jira module, you must add the alphanumeric attribute **Summary** to your Dimensions RM class. This attribute will map to **Summary**, a required field in Jira.

Required Fields

The following fields are required:

- Bug: **Summary** and **Reporter**
- Task: **Summary** and **Reporter**
- Improvement: **Summary** and **Reporter**
- Requirement: **Summary** and **Reporter**

Chapter 4

Setting Up Jira

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

Defining Projects and Fields

NOTE Before mapping attributes to fields, you must create Dimensions RM classes that correspond to Jira issue types. These classes must have the necessary attributes to map to Jira fields.

Define or identify the project, and fields in the project that you want to receive values from linked objects in Dimensions RM. Consider the field types as you map the fields. Jira accepts values from the foreign data source as text and converts the value, if necessary. If an incoming text string is too long, Jira throws an exception. It is possible to map date fields if in Dimensions RM the date format is set to "DD-MON-RRRR@HH24:MI:SS". For all other date formats, you must map the date field to a text field. If you map Jira selection fields to Dimensions RM list attributes, the values must match exactly or be mapped using a <ValueMap …> element in the XML configuration file. For best results, make the fields receiving Dimensions RM data in the Jira database read-only so that browser users cannot modify those fields. They represent Dimensions RM data that normally should be changed only within Dimensions RM.

You do not have to map all fields; however, you must map the fields that are defined as required in Jira or mandatory in Dimensions RM. The Dimensions RM attribute, to which you map mandatory Jira fields, should be set to mandatory. This prevents problems that would occur if you try to transfer an object from Dimensions RM with the attribute empty. You cannot map the internal Jira fields like **id** and **key**. These values are used internally by Jira, and cannot be changed. Depending on your configuration, if an attribute in Jira contains HTML formatting, it will be deleted in Dimensions RM. Certain Jira fields refer to Jira users, such as the **Assignee** field in the default Jira project. The transferred value (from Dimensions RM) must be a valid user in Jira.

IMPORTANT! You must have a text attribute for storing synchronization data in Jira. This attribute must be read-only for users and not be used for any other purpose.

Gathering Information about the Jira Data Model

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

Retrieving the Field ID for Custom Fields

For custom fields, the field name always starts with `customfield_`, followed by the field ID.

To retrieve the field ID, do the following:

1 Log in to Jira as an administrator.

2 From the Administration menu, select **Issues**.

3 In the **Fields** section of the Issues pane, click **Custom Fields**.

4. In the settings menu for the desired field, select **Edit**.
5. From the URL of your browser, extract the **id** parameter value.
   Example:
   http://myserver:8080/secure/admin/EditCustomField!default.jspa?**id**=***10018***
6. Combine the **customfield_** prefix and the id value and enter the result into the name attribute of the Field tag.
   Example:
   <Field name="**customfield_*10018***" source="*COST*"/>

Setting Up Jira Users

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

Chapter 5

Configuring the Sync Engine

Installing the Integration

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

About the XML Configuration File

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

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

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

Character Encoding and Text Editor Considerations

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

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

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

XML Entities - Escaping Characters

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

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

Character	Entity	Plain Text	XML / Comment
<	&lt;	Price < $ 30.00	Price &lt; $ 30.00
>	&gt;	Price > $ 30.00	Price &gt; $ 30.00

Sample XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<sample>

<quotation>This is a &quot;good&quot; example.</quotation>

<ampersand>Smith &amp; Co.</ampersand>

<house value='Jack&apos;s house' />

<house value="Jack's house" />

<house>Jack's house</house>

<lessThan>Price &lt; $ 30.00</lessThan>

<greaterThan>Price &gt; $ 30.00</greaterThan>

</sample>
```

Specifying General Options

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

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

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

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

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

Specifying Data Source Providers

About Data Source Providers

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

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

Data Source Definition Example

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

```
<Provider name="RTM" location="syncrtm.dll" description="Serena RM Sync
    Proxy" />
<Provider name="Jira" location="SyncJava.dll" description="Atlassian
    Jira Sync Proxy" />
```

Continue to the following sections for more details.

Specifying the Jira Data Source

Summary

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

Example Syntax

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

```
<DataSource name="JiraSource" provider="Jira">
  <Param name="user" value="admin" />
  <Param name="password" value="xxxx"/>
  <Param name="project" value="Jira_PROJECT"/>
  <Param name="host" value="http://jira.mydomain.com:8080"/>
  <Param name="jvm" value="C:\\Program Files (x86)\\Micro
  Focus\\Dimensions 12.6\\Common Tools
  1.8.0.0\\jre\\8.0\\bin\\client\\jvm.dll"/>
  <Param name="classpath" value="-Djava.class.path=C:\\Program Files
  (x86)\\Micro Focus\\Dimensions 12.6\\RM\\bin\\SyncJira.jar"/>
  <Param name="proxyAttribute" value="customfield_10014"/>
  <Param name="minuteOffset" value="0"/>
  <Param name="htmlFields" value="description"/>
</DataSource>
```

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

NOTE You can encrypt any parameter using the Sync Engine. Use the command-line option `-p` to retrieve a sample parameter with the value encrypted, and then paste that encrypted value into the configuration file. See "Controlling the Sync Engine from the Command Line" on page 40.

Guidelines

Review the following guidelines when defining these options

- `<DataSource name="JiraSource" provider="Jira">`: The DataSource can be set to any name, however this name needs to be set to the same value throughout the config.xml file.
- `<Param name="user" value="admin" />`: The user here is the Jira user account that the Sync Engine will use to access Jira. it is strongly recommended that you create a special Jira user account with API/Database access for use by the integration.
- `<Param name="password" value="xxxx"/>`: The password for the user account above.
- `<Param name="project" value="Jira_PROJECT"/>`: The name of the Jira project to be synchronized.
- `<Param name="host" value="http://jira.mydomain.com:8080"/>`: The URL under which the desired Jira project is available.
- `<Param name="jvm" value="C:\\Program Files (x86)\\Micro Focus\\Dimensions 12.6\\Common Tools 1.8.0.0\\jre\\8.0\\bin\\client\\jvm.dll"/>`: The path to the jvm.dll file.

NOTE Backslashes must be duplicated.

- `<Param name="classpath" value="-Djava.class.path=C:\\Program Files (x86)\\Micro Focus\\Dimensions 12.6\\RM\\bin\\SyncJira.jar"/>`: Change the path to match your system.
- `<Param name="proxyAttribute" value="customfield_10014"/>`: A Jira custom field only used for synchronization meta data.
- `<Param name="minuteOffset" value="0"/>`: The time difference in minutes between the Dimensions RM server and the Jira server.

NOTE If both servers run in different time zones, you have to modify this value each time one of these time zones changes from summer time to winter time or vice versa. To prevent this, use the same time zone for both servers.

- `<Param name="htmlFields" value="description"/>`: A comma separated list of Jira field names which store their content in HTML.

Specifying the Dimensions RM Data Source

Summary

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

Example Syntax

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

```
<DataSource name="RMSource" provider="RTM">
  <Param name="user" value="sync_userid" />
  <Param name="password" value="xxxx" />
  <Param name="host" value="RM_DB_name" />
</DataSource>
```

Guidelines

Review the following guidelines when defining these options

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

Specifying Value Mappings

Summary

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

Mapping Attribute Types

In some cases, Jira stores attribute information as the same type as Dimensions RM, but in a slightly different format. In this case, you must map the Dimensions RM attribute value to match the Jira field type and vice versa.

Example Syntax

The following is an example of a <ValueMap> element to map data from Dimensions RM to Jira. Verify the full range of item types in Jira in order to determine what types of requirements you can map from.

```
<ValueMap name="priority" leftsource="JiraSource"
rightsource="RMSource">
<Map left="High" right="High" />
<Map left="Highest" right="High" />
<Map left="Medium" right="Medium" />
<Map left="Low" right="Low" />
<Map left="Lowest" right="Low" />
</ValueMap>
```

Mapping a Set of Values to a Larger Set

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

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

Specifying the RM Category

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

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

Specifying the RM Category by Full Path

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

Specifying the RM Category by ID

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

Specifying the RM Category by Attribute Value

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

Specifying the RM Category by Value Map

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

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

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

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

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

</ValueMap>
```

Specifying the RM Category by Category Name

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

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

Specifying Events and Actions

Summary

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

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

This section covers important concepts about event triggers and actions. For additional reference material on supported element names, attributes, and values, see Appendix A, "Events, Triggers, and Actions" on page 45.

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.

Sample Event and Action Syntax

The following example synchronizes epics between Dimensions RM and Jira. The names of the Dimensions RM entities here are specific to this example; they will vary from system to system.

```
<Event name="New_RM_Epic" datasource="RMSource"
description="Synchronize Dimensions RM Epics to Jira.">
  <Trigger>
    <Condition>
      <Param event="included"/>
      <Param project="RM_INSTANCE" />
      <Param name="collection" value="Jira"/>
      <Param name="class" value="Epic"/>
    </Condition>
  </Trigger>

  <Create name="create"  datasource="JiraSource" description="Create
  new Epic in Jira">
    <Param name="project" value="Jira_Project_KEY"/>
    <Param name="issuetype" value="Epic" />
    <Field name="assignee" source="ASSIGNED_TO" map="user" />
    <Field name="summary" source="NAME" />
    <Field name="customfield_10005" source="PUID" />
    <Field name="description" source="DESCRIPTION"
    HTML_formatting="true" />
    <Field name="priority" source="PRIORITY" map="priority" />
    <Field name="status" text="To Do" />
    <Field name="customfield_10012" source="REQUIREMENT_LINK" />
  </Create>
</Event>
```

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

Defining Event Triggers

Summary

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

Trigger Parameters

Triggers contain the following parameters:

- `<Param event="included"/>`: Specifies the type of event trigger. See Event types below for more information.

- `<Param project="RM_INSTANCE"/>`: Specifies the Dimensions RM instance.
- `<Param name="collection" value="Jira"/>`: Specifies the collection in Dimensions RM to check for new requirements.
- `<Param name="class" value="Epic"/>`: Specifies the class of requirement to check in Dimensions RM.
- `<Param name="issuetype" value="Epic"/>`: Specifies the class of requirement to check in Jira.

Event types

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

Event Type	**Jira Description**	**Dimensions RM Description**
included	An object of the specified class has been created since the last synchronization. This may be based on its timestamp.	An object of the specific class has been added to the specified collection since the last synchronization.
modified	An object of the specified class (which has previously been synchronized) has a modification date that is newer than the last synchronization.	An object of the specified class has a modification date that is newer than the last synchronization time.
excluded	Not applicable	Dimensions RM has the concept of a collection, from which objects can be included or excluded. Jira does not support the excluded event.

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

Important Events Guidelines

Keep the following important guidelines in mind when defining events:

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

Defining Actions

Summary

There are two types of actions: `<Create …>`, and `<Update …>`. The actions specify the data source that is to accept the change triggered by an event in the other data source as well as the fields and parameters necessary to perform the action. The action block also specifies a name and description for each action, which do not need to be unique, are both optional, and are used for logging purposes.

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

Action Parameters

Actions contain the following parameters:

- `<Param class="`*`type_name`*`"/>`: The name of the item type or class to be created, or updated. For example, `<Param class="Requirement"/>`
- `<Param name="project" value="`*`project_name`*`"/>`: The name of the Jira project or Dimensions RM instance in which the item will be created. For example, to create, or update an item in a Jira project with key UC:
  `<Param name="project" value="UC"/>`

Defining Field Elements

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

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

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

  ```
  <Field name="rmissue" text="RM Issue {0}" source="PUID" />
  ```

  The use of the `{0}` token in the text attribute value creates a composite value (for example, `RM Issue` *`MRKT_00001`*). The `{0}` token is replaced with the current value of the PUID field. If you use the `{0}` token in a `text` attribute and do not specify a `source` attribute, the token is replaced with nothing.
- In the following example, the **`assignee`** field in Jira is mapped to a user field in Dimensions RM called **`ASSIGNED_TO`**. The **`assignee`** field specifies the user name of a Jira user. The `map` attribute specifies a value map that translates the Dimensions RM user (the value of the **`ASSIGNED_TO`** field) to the Jira user name.
  `<Field name="assignee" source="ASSIGN_TO" map="user"/>`

Synchronizing Categories

As Jira has no equivalent to categories in Dimensions RM, you can do one of the following:

1 **Specify no category:** This means that each requirement created in Dimensions RM will be created in the root category.

2 **Specify a fixed category:** You can specify a category under which the new requirements will be created.

3 **Use a Jira text field:** This text field receives or provides the full category path, e.g. *RMDEMO\Support\Maintenance*.

Synchronizing Links

In order to allow synchronizing links, note the following:

1 In Dimensions RM, a relationship must exist for the class to which you synchronize. For further information on how to create relationships, refer to the chapter *"Adding a New Relationship"* in the *Dimensions RM Administrator's Guide*.

2 You can only synchronize links from Jira to Dimensions RM.

3 Synchronization of multiple links is not supported.

For synchronizing links, the XML configuration file must contain the following definitions:

- `<Param name="rmRelationName" value="relates_to" />`
- `<Field name="RELATES_TO" source="issuelinks" />`

Replace ***rmRelationName*** with the name of your Dimensions RM relation.

The following XML is an extract of a configuration and shows the relevant sections for the Create and Modified events with the link related tags (`Param` and `Field`) made bold.

```
<!-- The Create definition -->

<Event name="Jira_Create_RM_Task" datasource="JiraSource"
description="Synchronize Jira Task to Dimensions RM.">

<Trigger>

<Condition>

<Param event="included" />

<Param project="UC" />

<Param name="issuetype" value="Task" />

<Param name="query" value="labels = RM" />

<Param name="rmProject" value="RMDEMO" />
```

```
<Param name="rmRelationName" value="relates_to" />
<!-- Replace rmRelationName with the name of your Dimensions RM
relationship. -->

</Condition>

</Trigger>

<Create name="Jira Create RM Task" datasource="RMSource"
description="Create new Task in Dimensions RM.">

<Param project="RMDEMO" />

<Param class="Task" />

<Field name="TITLE" source="summary" />

<Field name="DESCRIPTION" source="description" HTML_formatting="false"
/>

<Field name="JIRA_LINK" source="url" />

<Field name="RELATES_TO" source="issuelinks" />

</Create>

</Event>


<!-- The Update Definition -->

<Event name="Jira_Update_RM_Task" datasource="JiraSource"
description="Synchronize Jira Task to Dimensions RM.">

<Trigger>

<Condition>

<Param event="modified" />

<Param project="DEMO" />

<Param name="issuetype" value="Task" />

<Param name="rmProject" value="RMDEMO" />

<Param name="rmRelationName" value="relates_to" />
<!-- Replace rmRelationName with the name of your Dimensions RM
relationship. -->

</Condition>

</Trigger>

<Update name="Jira Update RM Task" datasource="RMSource"
description="Create new Task in Dimensions RM.">

<Param project="RMDEMO" />

<Param class="Task" />

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

```
<Field name="DESCRIPTION" source="description" HTML_formatting="false"
/>

<Field name="RELATES_TO" source="issuelinks" />

</Update>

</Event>
```

Validating the XML Configuration File

Summary

When you attempt to run the Sync Engine, the Sync Engine will validate the XML configuration file. Validation is the process of discovering whether the XML in the document is valid; that is, whether it conforms to the schema named in the header section of the file. The process is similar to compiling a program and finding bugs.

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

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

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

The file named `SyncEngine.xsd` is the XML schema file used for the integration. By default, this file is located in the `<RM_Install_Dir>\conf` directory. The XML parser in the RM-Jira 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) is not valid. This refers to the DataSource section at the beginning of the XML configuration file. Check this section to be sure that you are using the correct name for the data source.

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

Inspect the configuration file if this error message appears.

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

This error generally indicates one of the following:

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

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

Hints

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

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

Chapter 6

Using the Sync Engine

About the Sync Engine Service

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

Testing Connections

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

Testing the Jira Connection

To test the connection to Jira:

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

   For example, if the server is called **jira** and the port number is **8080**, open:
   `http://jira:8080`
2. To further confirm that the connection works correctly, consider logging in to Jira and creating a requirement. This also allows you to verify which fields are mandatory.

Testing the Dimensions RM Connection

To test the connection to Dimensions RM:

1. From the Dimensions RM browser client, verify that you can log in using the account that is specified in the config.xml file. See Chapter 5, "Configuring the Sync Engine" on page 21.

Using Synchronization Logs

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

Setting Logging Options

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

- The level of detail (DEBUG, INFO, WARN, ERROR)

- The location of the log file
- The name of the log file
- The maximum size of the log file
- The maximum number of log files that will be maintained as a result of "rolling" to additional files after a log file reaches the maximum size limit

The `log4cpp.conf` file contains five categories:

- fileBrowser
- fileSyncEngine
- fileWebService
- fileDbDoctor
- fileAlfEventEmitter

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

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

Logging Recommendation for Testing

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

Controlling the Sync Engine from the Command Line

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

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

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

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

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

Verifying Synchronization Results

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

Verifying that Synchronization Completed Successfully

To verify synchronization results:

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

   <installation_location>\logs

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

3. Once `syncEngine.log` and `syncEngine_jira.log` files no longer increase in size, the Sync Engine service has likely completed. Display the Windows services, and verify that the service has stopped.

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

5. Scroll to the bottom.

6 In syncEngine.log, the final entries should look something like the following:
```
08-28-16 06:43:26.787 INFO   SyncEngine           - ! Successfully
    closed connection to DataSource 'RMSource' (Serena Test Sync
    Proxy).
08-28-16 06:43:26.798 INFO   SyncEngine           - ! Successfully
    closed connection to DataSource 'JiraSource' (Test Director Sync
    Proxy).
```

The "Successfully closed connection" message indicates that the processes has completed successfully. If this does not appear, then the synchronization may have failed due invalid XML, a data exception, or other issues. See "Troubleshooting" on page 43.

Validating Synchronization Results

To verify synchronization results:

1 Once synchronization is complete, open the `syncEngine.log` file in a text editor.

2 Scroll to the bottom of the file, then scroll up until you find the start of the date and time entries for the most recent synchronization. For example, if the date is August 28, 2016, search for a line like the following:
```
08-28-16 06:43:17.731 NOTICE SyncEngine           - ! Successfully
    opened connection to DataSource 'RMSource' (Serena Sync Proxy
    for Dimensions RM).
08-28-16 06:43:20.260 NOTICE SyncEngine           - ! Successfully
    opened connection to DataSource 'JiraSource' (Serena Sync Proxy
    for Jira).
```

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

- `Successfully retrieved 1 Product_Requirements objects for project`
- `Begin Create Action`
- `Adding a requirement`
- `Posting the requirement to Jira: success`
- `Checking in the item: Success`

Verifying What Records Have Been Synchronized

A record of which requirements have been synchronized is maintained in a text field for each requirement. For further information see chapter "Defining Projects and Fields" on page 18.

Troubleshooting

General Troubleshooting Checklist

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

- Users are configured correctly. For example, the Jira user is a project administrator, the Dimensions RM user is an administrator, and the names are the same across the two systems.
- The latest updates have been installed to the Dimensions RM server, and you are working with supported versions of Jira and Oracle.
- For best results, Jira is running on a 64-bit Windows server.

Troubleshooting Specific Issues

syncengine Service Fails Immediately with No or Minimal Log

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

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

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

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

Possible cause The requirement in question has already have been synchronized to Jira, but has been removed and then re-added to the collection being synchronized. Re-adding the requirement to the collection triggers the Sync Engine to attempt to create it in Jira. This might happen if, for example, you have moved all of your Dimensions RM requirements into a new collection and are now attempting to synchronize the new collection. The Sync Engine therefore registers these as new requirements and the error above appears in the Sync Engine log. When this occurs, clean the create event out of the system registry to remove the failed synchronizations. The next synchronization will run against the new collection as expected.

Synchronization Does Not Create Jira Requirements

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

Synchronization Creates Jira Requirements but Does Not Update Them

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

Synchronization Runs but Reports Oracle Error: object not found

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

Appendix A

Events, Triggers, and Actions

Events

- Create
- Update

```
<Event name="New_RM_Epic" datasource="RMSource"
description="Synchronize Dimensions RM Epics to Jira.">
```

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

The **`datasource`** value must be an Atlassian Jira datasource as defined in the head of the configuration file, in case the event is triggered by Jira.

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

Triggers

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

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

event Parameter

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

Possible values are:

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

- `Modified`—Defines an event where an object that has already been transitioned to Dimensions RM is changed in Dimensions RM or Jira. The event may be filtered by class and attribute conditions. In Jira, field values are changed in response to changes in corresponding items in Dimensions RM.

name Parameter

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

If the `name` parameter is set to **`class`**, then the `value` attribute must be the name of a Dimensions RM class. If the name parameter is set to **issuetype,** the `value` attribute must be the name of a Jira issue type (for example, Requirement, Bug or Task). See Appendix B, "Jira Data Types" on page 49 for a list of available issue types in Jira. The `value` attribute is case sensitive and should appear as it is shown in Jira.

Example:

Dimensions RM: `<Param name="class" value="ClassName" />`

Jira: `<Param name="issuetype" value="IssueTypeName" />`

Actions

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

- Create
- Update

Create

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

```
<Create name="create" datasource="JiraSource" description="Submit
    new RM object">
      <Param name="issuetype" value="Requirement" />
      <Field name="name" source="TITLE" />
      <Field name="description" text="Object from RM" />
</Create>
```

The `class` parameter defines the target class in Dimensions RM. For Jira use name="issuetype" instead. In case the datasource is set to a Dimensions RM type source, this may be any available class in the specified project for this event. In case of Jira set as the datasource, there is a fixed set of available classes in Jira. See Appendix B, "Jira Data Types" on page 49 for a detailed description.

Example:

Dimensions RM: `<Param class="ClassName" />`.

Jira: `<Param name="issuetype" value="IssueTypeName" />`

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

The `name` attribute specifies the target field of the object in the specified class. This can be one of the fixed system attributes (for a detailed view of system attributes of the different class types in Jira refer to Appendix B, "Jira Data Types" on page 49) or any valid user attribute. A valid user attribute is one that was defined in Quality Center

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

```
<Field name="description" text="Requirement {0} from RM" source="PUID"/>
```

The `text` attribute may contain a token `{0}` that will be replaced by the data from the defined source attribute. For example, for a requirement in Dimensions RM with MRKT_000001 as its PUID, the description field of the object in Jira will be set to "Requirement MRKT_000001 from RM".

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

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

Update

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

```
<Update name="update" datasource="JiraSource" description="Replace
    RM object data with Jira data">
      <Field name="description" text="Object was changed in RM" />
      <Field name="summary" source="Title" />
      <Field name="rmstate" source="State" />
</Update>
```

The `name` attribute specifies the target field of the object in the specified class. This can be one of the fixed system attributes (for a detailed view of system attributes of the different class types in Jira (refer to Appendix B, "Jira Data Types" on page 49) or any valid user attribute. A valid user attribute is one that was defined in Jira.

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

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

```
<Field name="description" text="Requirement {0} from RM" source="PUID"/>
```

The `text` attribute may contain a token `{0}` which will be replaced by the data from the defined source attribute. For example, for a requirement in Dimensions RM with MRKT_000001 as its ID, the description field of the object in Jira will be set to ‘Requirement MRKT_000001 from RM’.

```
   <Field name="TEXT" source="description" />
```

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

Appendix B

Jira Data Types

The following table contains a list of Jira fields. Any field may be addressed using either the field label or the field name. Entries in red (Supported = **No**) cannot be mapped, because they use unsupported data types.

Jira Attribute	Jira Type	Data Type	Supported
Assignee	User	User	Yes
Attachment	Attachment		No
Comment	Comment		No
Component Objects	ProjectComponent List	String: Comma separated values	Yes
Created	Date	Date/String	Import to RM only[1]
Creator	User	User	Yes
Description	Text	Memo	Yes
Due Date	Date	Date/String	Yes[1]
Fix Versions	Version List	String: Comma separated values	Yes
Id	Number	Long	Import to RM only
Issue Type	Text	String	Yes
Key	Text	String	Import to RM only
Labels	Text	String: Comma separated values	Yes
Original Estimate	Number	Long	Yes
Priority	Priority	Lookup List	Yes
Project	Project	String	Yes
Remaining Estimate	Number	Long	Yes
Reporter	User	User	Yes
Resolution Date	Date	Date/String	Yes[1]
Resolution Id	Text	String	Yes
Status	Text	String	Yes
Sub Tasks	Issue List		No
Summary	Text	String	Yes
Time Spent	Number	Long	Import to RM only
Updated	Date	Date/String	Import to RM only[1]

Jira Attribute	Jira Type	Data Type	Supported
User defined attributes	Number String Lookup List User	Number String Lookup List User	Yes
User defined attributes	Date	Date/String	Yes[1]
Votes	Number	Long	Import to RM only
Watches	Number	Long	Import to RM only
Workflow Id	Number	Long	Import to RM only

[1] For date fields: Dimensions RM date format is `DD-MON-RRRR@HH24:MI:SS`. Other date formats must be mapped to text.

NOTE Dimensions RM’s group attributes are not supported by Sync Engine.