TIBCO BusinessEvents™ Rule Template Design Specification

T

Document Revisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 1.0.0 | March 15th 2011 | Suresh Subramani | Initial Document |
|  |  |  |  |
|  |  |  |  |

Disclaimer : This Document is subject to change without notice

Copyright Notice

COPYRIGHT© 2011 TIBCO Software Inc. This document is unpublished and the foregoing notice is affixed to protect TIBCO Software Inc. in the event of inadvertent publication. All rights reserved. No part of this document may be reproduced in any form, including photocopying or transmission electronically to any computer, without prior written consent of TIBCO Software Inc. The information contained in this document is confidential and proprietary to TIBCO Software Inc. and may not be used or disclosed except as expressly authorized in writing by TIBCO Software Inc. Copyright protection includes material generated from our software programs displayed on the screen, such as icons, screen displays, and the like.

Trademarks

Technologies described herein are either covered by existing patents or patent applications are in progress. All brand and product names are trademarks or registered trademarks of their respective holders and are hereby acknowledged.

Confidentiality

The information in this document is subject to change without notice. This document contains information that is confidential and proprietary to TIBCO Software Inc. and may not be copied, published, or disclosed to others, or used for any purposes other than review, without written authorization of an officer of TIBCO Software Inc. Submission of this document does not represent a commitment to implement any portion of this specification in the products of the submitters.

Content Warranty

The information in this document is subject to change without notice. THIS DOCUMENT IS PROVIDED "AS IS" AND TIBCO MAKES NO WARRANTY, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO ALL WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TIBCO Software Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

For more information, please contact:

TIBCO Software Inc.  
3303 Hillview Avenue  
Palo Alto, CA 94304  
USA

**Table of Contents**

[TIBCO BusinessEvents™ Rule Template Design Specification 1](file:///Q:\simulator\1.0\docs\TIBCO%20BusinessEvents%20RuleTemplates.docx#_Toc287954451)

[1 Introduction 4](#_Toc287954452)

[1.1 Purpose 4](#_Toc287954453)

[1.2 Scope 4](#_Toc287954454)

[1.3 Audiences 4](#_Toc287954455)

[2 Concepts 5](#_Toc287954456)

[2.1 Design Elements 5](#_Toc287954457)

[2.2 Roles and Responsibility 5](#_Toc287954458)

[2.3 Governance 5](#_Toc287954459)

[3 Rule Templates 5](#_Toc287954460)

[3.1 Model 5](#_Toc287954461)

[3.2 Editor 5](#_Toc287954462)

[3.3 Packaging 5](#_Toc287954463)

[4 Rule Template Instances 6](#_Toc287954464)

[4.1 WebStudio 6](#_Toc287954465)

[4.2 Constraints 6](#_Toc287954466)

[4.3 Testing 6](#_Toc287954467)

[4.4 Code Management and Persistence 6](#_Toc287954468)

[5 Deployment 7](#_Toc287954469)

[5.1 Code Generation 7](#_Toc287954470)

[5.2 Packaging 7](#_Toc287954471)

[5.3 Policy and Approvals 7](#_Toc287954472)

[6 Runtime 7](#_Toc287954473)

[6.1 Execution Semantics 7](#_Toc287954474)

[6.2 Notifications 7](#_Toc287954475)

[6.3 Exceptions 7](#_Toc287954476)

# Introduction

## Purpose

This document describes the design principles of BusinessEvents RuleTemplates.

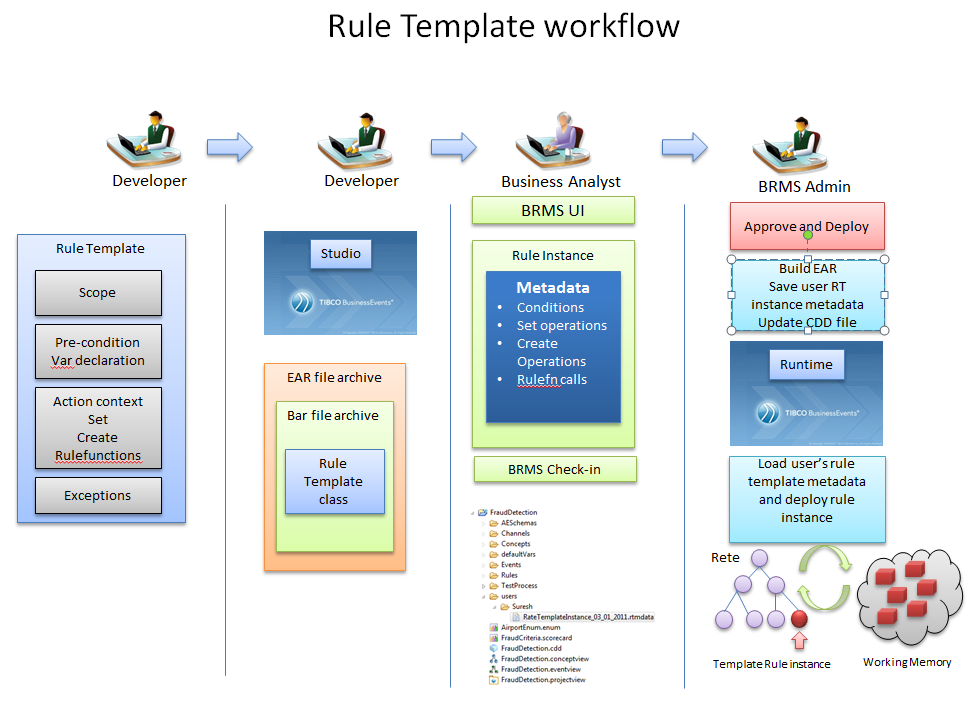
## Scope

The scope of the document is to provide a complete spec of Rule Templates in terms of UI, Model, Packaging, Runtime and Deployment aspects

## Audiences

Developers, Architects, QA, Support, and Product Managers

# Concepts



## Design Elements

A rule template is very similar to a Virtual Rule Function. In that it has the structure of the Rule but no execution logic – both in condition and action.

The design elements of the Rule templates revolves around the following concept

* Rule Template Definition in BE Studio
* Rule Template Instance in Web Studio
* Deployment Methodology
* Execution Semantics

## Roles and Responsibility

We assume the following roles and responsibility for the entire design to execution phase

### Developer Role :

* Has the responsibility of defining Rules in BE Studio
* Has the responsibility of creating the Rule Templates in BE Studio
* Has the responsibility of providing the right functions and computed variables in Template for later use

### Business User/Analyst Role

* Creates a Rule Instance from the Rule Template
* Saves it, Test it, Provides policies around its execution
* Submit for Approval/Deployment

### Administrator Role

* Verifies consistency in project
* Build Ears, and hot deploys to engine

## Assumptions

* Template rules are implemented and instantiated by Business users.
* Templates provide a highly constrained scope of implementation.
* Template rules have no complex statements.
* Template rules have no conditional branching.
* Template rules have simple assignments.
* Template rules have no cache statements.

## Governance

<<TODO List from RMS>

# Rule Templates

* Rule Templates are first class citizens in BE Studio.
* The rule template rule is refactor able itself and its dependencies.
* The template rule will use a subset of the catalog functions allowed categorized separately.
* The template rule cannot be invoked using Catalog invokeXXXXX() functions.
* They are stored in disk as a file (.ruletemplate) and visible in any folder in the studio project as shown below

<RMS BE Project>

<Rules>

<RuleTemplateName.template>

<Rulefunctions>

<Concepts>

<Users>

TemplateRule< RuleTemplateName >\_<UserName>.rtmetadata

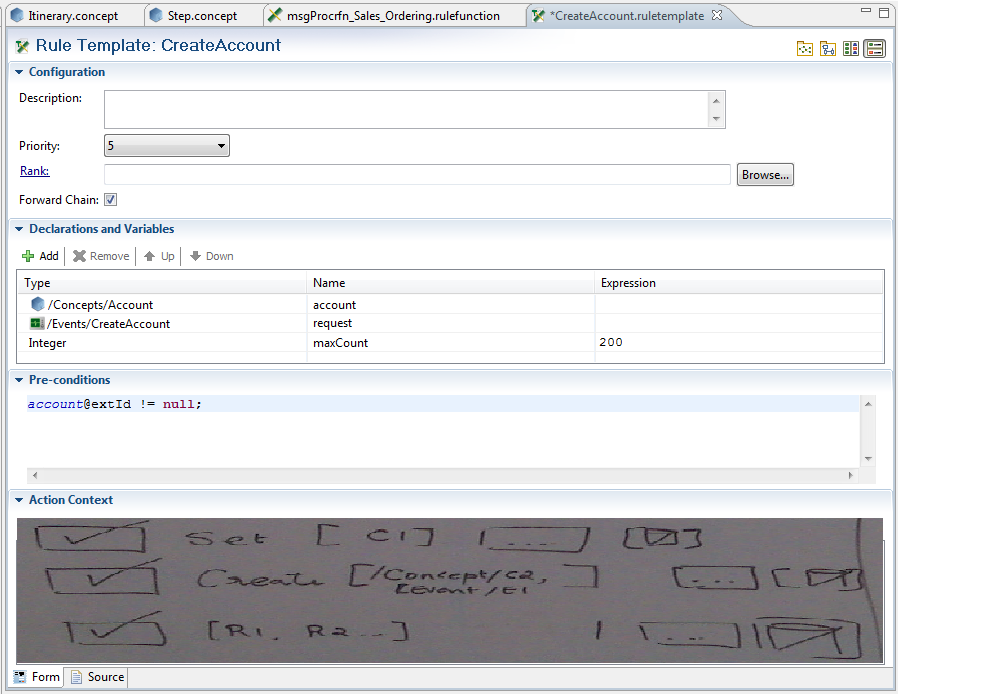
TemplateRule< RuleTemplateName >\_<UserName>.rtmetadata

## Model

* The template rule metadata will be captured into the Studio ECore model.
* The template rule is available via ontology for code generation.
* The template rule will be edited by a dedicated Rule template editor(see next section)

## Editor

A sample mock up UI



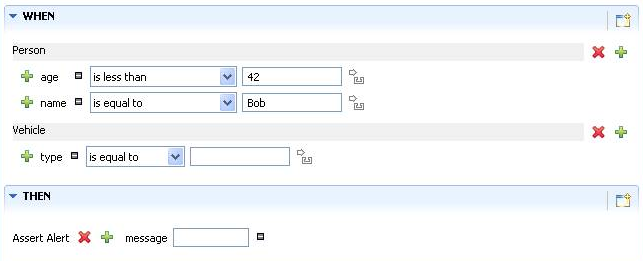
* The editor context is determined by the scope artifacts and the variable declarations.
* The editor context is visible to the pre-conditions and the action-context.
* The pre-condition section can be used to define variables i.e primitive types and BE ontology types.
* Each line in the pre-condition is considered an expression whether it is a declaration/assignment or Boolean expression.
* The precondition expression allows the template designer and end user to drill down to ontology type properties and attributes with code assist. The template designer can provide base conditions and leave the fine tuning conditions to be done by the end user.
* The action context sections have 3 categories
  + Set – The set section will list all the ontology types to be selected by the template designer and made available to the end user for selection. The end user can drill down into the ontology object properties and attributes to set values.
  + Create – This create section will list all the ontology types to be selected by the template designer and made available to the end user for ontology object creation. The end user has the responsibility to provide construction parameter values. Q: Is mapping allowed here ?
  + Rulefunctions – The rulefunctions section will list all the available rule functions to be selected by the template designer and made available to the end user for the implemented call. The end user has the responsibility to provide parameter values. Q: Is mapping allowed here ?

## Packaging

* The Rule template will generate java class which Implements the Rule interface. The class will be packaged inside the be.jar file inside the .bar file.

# Rule Template Instances

## WebStudio



## Constraints

The constraints are driven by the template.

The template provides

## Testing

## Code Management and Persistence

# Deployment

## Code Generation

The rule template is used to generate a template rule class which does not have conditions eval() function implementation and Action exec() function implementation. The implementation is provided from the business user’s metadata which is saved in the user’s folder inside the Studio project folder structure. The rule template generated class has a different constructor and the standard runtime needs to differentiate a template rule class from a regular rule class.

## Packaging

The deployed rule metadata for every rule instance is kept in the project folder named after the user who created the instance from the BUI interface. The rule path information is added to the deployment CDD file by the BUI/Rule administrator. The rule metadata also available from the Shared Archive folder where the whole project tree is archived during EAR build process and the SAR functions can be used to retrieve them to create instances after hot deploy. The administrator needs to build the EAR and update the CDD when a rule template instance is hot deployed. During hot deployment it needs to be ensured that all processing units in the cluster deploy the template and the generated class in the EAR file and create instances from the metadata which are available in the SAR and indicated in the CDD for deployment.

## Policy and Approvals

# Runtime

## Execution Semantics

## Notifications

## Exceptions