

SAP NetWeaver Business Rules Management: Best Practices



Applies to:

This article assumes that the reader has enough knowledge of the SAP NW Business Rules Management tools available with SAP NW CE 7.3 onwards. For more information, visit the [Business Rules Management homepage](#).

Summary

This article provides details of the **best practices** that are to be followed by any user of SAP NW Business Rules Management product.

Author: Santosh Giri Govind Marthi

Company: SAP Labs India Pvt. Ltd.

Created on: 15 September 2011

Author Bio



Santosh Giri has been part of the NW BRM team for about 5 years 6 months and has been involved in topics relating to NW BRM component like creating of customer POCs, Development of Rules Manager (Web based business user tool), Testing and also handling customers with issues on NW BRM component

Table of Contents

Introduction	3
BRM Best Practices	3
Separate Decision Logic and Application Logic:.....	3
While modeling rules choose wisely if RETE or Flow is appropriate:.....	3
Choose correct artifact for your requirement:	4
Write decision tables with less than 20,000 rows and 1,00,000 cells if it is inevitable:	4
Synchronize with runtime version of rules before modifying in NWDS:.....	7
Calculate optimal input batch size for processing:.....	7
Use of Lean Rule Engine:	7
Design rules keeping Business User in mind:	7
Related Content.....	8
Copyright.....	9

Introduction

This article assumes that the reader has enough knowledge of the SAP NW Business Rules Management tools available with SAP NW CE 7.3 onwards.

Before implementing rules in BRM, question yourself the following and then proceed with the implementation.

- How frequently rules will change?
- Do we need visibility to these rules?
- Will implementing rules in BRM take more time than in application to execute?

And basing on the answers to the above questions, we can decide whether to use BRM for the rules implementation or not. Once the decision is made to use BRM for the implementation of rules, any business user implementing rules using SAP NW BRM should consider the following best practices so that he will be able to leverage the full abilities of the product.

BRM Best Practices

Following are the best practices that will allow the user implementing rules using SAP NW BRM to generate good quality rules implementation:

Separate Decision Logic and Application Logic:

Clearly separate decision making logic in application design and use BRM to ONLY support decision making. Do not pollute rules with other application logic like calling web services, external systems or database tables. This causes maintenance problems.

Guidelines:

- Use BRM to ONLY support decision making
- Provide input data required to BRM Engine and BRM will update some part of the data based on the rules and returns it back to the invoking application.

While modeling rules choose wisely if RETE or Flow is appropriate:

We recommend customers to use *Rete Rulesets* when he needs to write validation rules. We can also use *Flow Rulesets* to write validation rules but, *Rete Rulesets* are executed by the RETE engine which is implemented based on the RETE algorithm. RETE algorithm provides optimized performance in executing the rules. In case of validation rules, where the non-satisfaction of even one rule is sufficient for conclusion, RETE execution of rules would be most appropriate. And hence we recommend the usage of *Rete Rulesets* for validation rules.

You can find details about the RETE algorithm and its usage in BRMS in the following article:
<http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/10dea1d3-fbef-2d10-0e89-a7447f95bc0e>

And we recommend customers to use *Flow Rulesets* when he needs to write calculation rules or other complex rules which needs some kind of looping/complex logic to be implemented in the rules. As *RuleFlows* provide WYSIWYG (What You See Is What You Get) format of designing rules, this will provide good visibility to the users.

For example, let us consider the “Loan Approval Rules”, after the validation of the application is completed, the interest rate calculation involves some complex logic and in that case, *Flow Ruleset* provides the flexibility to the developer to write those complex rules using the *RuleScripts* and *RuleFlows* (both of them provides looping and other features which are not available in *Rete Rulesets*). And provides the business user with the visibility of which rule has been executed after which one, which enables him to respond to the customer requests like “Why is the rate of interest high for him?”

You can find details about the concepts of BRM in the below link:

http://help.sap.com/saphelp_nw73/helpdata/en/81/097529e0e545e58bb2d7a34e4d0a2a/frameset.htm

Please navigate to “Modeling Rules with Rules Composer” -> “Concepts” and you will find the details about all the artifacts that are there in BRM that can be used to write rules.

Guidelines:

- RETE Ruleset:
 - Execution based on RETE algorithm, BRM Engine takes control of execution
 - For validation rules like Claim validation, Credit Card Application validation
- FLOW Ruleset:
 - You model the flow of execution using FLOWS
 - For looping through list of objects
 - For calculation rules like claim settlement, student rank calculation, etc.

Choose correct artifact for your requirement:

Before implementing rules, please understand the existing artifacts available in the BRM and then choose wisely the artifact to be used for implementing your rules.

Guidelines:

- Rule:
 - For simple conditions and actions
 - One level condition check
 - Create rules with common conditions and reuse them across ruleset
 - Reuse conditions by using Pre-condition feature
 - Is editable in Rules Manager
- Decision Table:
 - Gives you better visibility
 - Powerful MS-Excel integration with Import/Export feature
 - Cannot perform action types ASSERT, RETRACT and RE-EVALUATE
 - Is editable in Rules Manager
- Rulescript:
 - For multilevel (nested) conditions
 - Loop through list of objects
 - For complex calculations
 - Is non-editable in Rules Manager

Write decision tables with less than 20,000 rows and 1,00,000 cells if it is inevitable:

Decision tables are one of the major features provided by the SAP NW BRM which provides the flexibility to group similar rules which have to be checked for different set of values and take actions based on the condition values.

If the decision table is too big then that causes serious performance problems and if the scenario needs the decision table to be that big, then we recommend “Splitting the decision table into smaller decision tables within the above mentioned size” so that the performance will not get hit.

For example, if we consider “Credit Card Application Rules” then Credit Score calculation table will look like:

Decision Table: Standard Card Eligibility Table

Documentation and Properties

Condition Action... Change Layout Context Menu Export... Import...

Credit History Of Applicant		GOOD	MODERATE	BAD
Annual Income Of Appli...	Total Assets Of Applicant	CreditScore	CreditScore	CreditScore
< 50000	< 100000	42	37	30
	Between 100000 and 500000	50	44	37
	> 500000	58	52	45
Between 50000 and 80000	< 100000	53	47	40
	Between 100000 and 500000	60	54	47
	> 500000	68	62	55
Between 80000 and 100000	< 100000	58	52	45
	Between 100000 and 500000	65	59	52
	> 500000	73	67	60
> 100000	< 100000	63	57	50
	Between 100000 and 500000	70	62	57
	> 500000	78	72	65

In the above case, we can divide the decision table into three smaller decision tables by removing the Horizontal Condition and creating three separate decision tables for "CreditScore" calculation for three different values of the Credit History condition.

Decision Table: StandardCardEligibility_GOOD

Documentation and Properties

Condition Action... Change Layout Context Menu Export... Import...

Annual Income Of Applicant	Total Assets Of Applicant	CreditScore
< 50000	< 100000	42
	Between 100000 and 500...	50
	> 500000	58
Between 50000 and 80000	< 100000	53
	Between 100000 and 500...	60
	> 500000	68
Between 80000 and 100000	< 100000	58
	Between 100000 and 500...	65
	> 500000	73
> 100000	< 100000	63
	Between 100000 and 500...	70
	> 500000	78

Decision Table: StandardCardEligibility_BAD

Documentation and Properties

Condition Action... Change Layout Context Menu Export... Import...

Annual Income Of Applicant	Total Assets Of Applicant	CreditScore
< 50000	< 100000	30
	Between 100000 and 500...	37
	> 500000	45
Between 50000 and 80000	< 100000	40
	Between 100000 and 500...	47
	> 500000	55
Between 80000 and 100000	< 100000	45
	Between 100000 and 500...	52
	> 500000	60
> 100000	< 100000	50
	Between 100000 and 500...	57
	> 500000	65

We can also divide the decision table by reducing the number of rows and moving them to another decision table and invoke them one after the other continuously.

In the above case, we can divide the decision table into two different decision tables by reducing the number of rows as follows:

Decision Table: StandardCardEligibilityTable 1

Documentation and Properties

Condition Action... Change Layout Context Menu Export... Import...

Credit History Of Applicant		GOOD	MODERATE	BAD
Annual Income Of Appli...	Total Assets Of Applicant	CreditScore	CreditScore	CreditScore
< 50000	< 100000	42	37	30
	Between 100000 and 500000	50	44	37
	> 500000	58	52	45
Between 50000 and 80000	< 100000	53	47	40
	Between 100000 and 500000	60	54	47
	> 500000	68	62	55

And

Decision Table: StandardCardEligibilityTable 2

Documentation and Properties

Condition Action... Change Layout Context Menu Export... Import...

Credit History Of Applicant		GOOD	MODERATE	BAD
Annual Income Of Appli...	Total Assets Of Applicant	CreditScore	CreditScore	CreditScore
Between 80000 and 100000	< 100000	58	52	45
	Between 100000 and 500000	65	59	52
	> 500000	73	67	60
> 100000	< 100000	63	57	50
	Between 100000 and 500000	70	62	57
	> 500000	78	72	65

And while invoking the Decision tables, invoke them in the same order to get the same effect.

Note: The above described scenario is simple one so that understanding and showing the recommendation will be easy. But if the performance is a key factor then keeping the Decision Table to a small size would help.

Note: If a ruleset contains a big Decision Table, then it is suggested that you only have one Decision Table per Ruleset as, when a ruleset is invoked then the whole ruleset will be loaded into the memory and if we have more than one huge Decision Tables in that ruleset, all those Decision Tables are loaded which might affect the performance.

Guidelines:

- Limit size of decision table to 20,000 rows and 100,000 cells. Better to have small Decision Table sizes to avoid any performance issues due to the huge sizes of the Decision Tables.
- In case the Decision Table size has to be huge and it is inevitable, then it is better to have one such big Decision Table per Ruleset. Reason being, if a ruleset is invoked then whole Ruleset will be loaded into the memory and if the Ruleset contains more than one such huge Decision Table, then it might affect the performance of the rules execution.
- Split large decision tables into logically related smaller decision tables

- Definition and alias names should be distinct from condition and action values in DT
- Cannot perform action types ASSERT, RETRACT and RE-EVALUATE
- Methods with multiple arguments not supported in DT Actions

Synchronize with runtime version of rules before modifying in NWDS:

We recommend the developer/business user to synchronize the rules content with the runtime version before modifying the rules in NWDS so that the modifications will be done on the latest rules content rather than on obsolete rules content. By doing so, you make sure that the changes made by the business users are not lost.

In general, we use Rules Manager for the changes to be done by the business users. And in some cases, these changes are huge/many and also involves modifying Flows, Rulescripts, etc. which are non editable in the Rules Manager application (which is a limitation for this tool). And in such cases the modifications are to be done using the Rules Composer tool and in that scenario, we want to make modifications to the latest rules and do not want to lose on the changes done by the business users previously. And this can be achieved using the “Download Runtime Version” feature of NW BRM Rules Composer tool.

Guidelines:

- Always work on the latest rules so that the changes of the business users are not lost
- In NWDS, always synchronize with runtime version of the rules content before modifying and deploying

Calculate optimal input batch size for processing:

This applies only for the RETE Rulesets as the execution of the RETE Rules are based on the RETE algorithm and the BRM engine takes care of the control of the execution. When done with writing rules, test the rules with different sizes of input and measure the performance and then decide the batch size of the input to the rules.

In a customer scenario, the requirement was to process 10 million records, and the customer passed 1000 records each time to the rules and the BRM Engine took 44 secs to process these 1000 records. This means it takes ~5 days to process 10 Million records.

And then we tested with a batch size of 50 records each time and BRM Engine took 2-3 secs to process 1000 records, which effectively reduced the total processing time for 10 million records to 6-7 Hours.

Guidelines:

- Applicable only for RETE Rulesets
- Plan number of input objects to Ruleset and model rules to process one or multiple input objects
- Rules modeled on java classes perform better
- Test rules using different sizes of input to measure performance

Use of Lean Rule Engine:

In distributed deployment scenarios always use local rule execution with Lean Rule Engine rather than making remote calls to the central BRM instance.

Design rules keeping Business User in mind:

In scenarios where the business users use SAP NW BRM Rules Manager extensively for modifying the rules, and the changes to the rules are also frequently done, then it is better the rules are modeled using If-Then rules or *Decision Tables* and not using *RuleScripts* or *RuleFlows*. As modifying *RuleScripts* and *RuleFlows* is not possible using SAP NW BRM Rules Manager, this kind of modeling will compel the business users to use SAP NW BRM Rules Composer to make modifications.

Related Content

www.sdn.sap.com

www.bpx.sap.com

www.boc.sap.com

[SAP NetWeaver and SAP BOBJ topics](#)

[Refer to How-to-Guides.](#)

[podcasts designed to enhance your skills.](#)

<http://www.facebook.com/pages/SAP-RIG/119256894764191?ref=ts>

<http://twitter.com/saprig>

[Business Rules in SAP](#)

[Sneak view into the features of SAP NetWeaver BRM 7.2](#)

[Working with Rules Manager on SDN](#)

For more information, visit the [Business Rules Management homepage](#).

Copyright

© Copyright 2011 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Oracle Corporation.

JavaScript is a registered trademark of Oracle Corporation, used under license for technology invented and implemented by Netscape.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects S.A. in the United States and in other countries. Business Objects is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.