About BRF and BRFplus

Applies to:

Business Rule Framework plus (BRFplus) shipped with SAP NetWeaver 7.0 Enhancement Package 2.

For a detailed introduction to BRFplus please visit Business Rules Management at the SDN network and select 'BRFplus' in the Related Content section.

Summary

This paper explains how the established BRF relates to the new BRFplus. It explains differences in the concepts and compares the important artifacts of both products. Finally, it gives advice when to use which product.

Authors: Carsten Ziegler, Lars Biewald

Company: SAP AG

Created on: January 1st, 2011

Author Bio

Carsten Ziegler is the Architect and Product Owner of Business Rule Framework plus. He joined SAP in 2000. Since then he has been working in various projects as a developer, development architect and project lead.

Lars-Eric Biewald is a former BRF developer. He also worked in BRFplus for several months. Recently, he developed the tool for migration from BRF to BRFplus.

Table of Content

Prerequisites

- Basic knowledge of BRF
- Basic knowledge of BRFplus

Preface

Many customers ask questions about:

- future of BRF and BRFplus
- comparison of BRF with BRFplus
- criteria to decide either for BRF or for BRFplus
- reason for development of BRFplus
- migration strategies

This paper gives answers to these questions.

How It All Began

In the year 2005, several development architects had the task to investigate the usage of business rules in a future product with code name Vienna – nowadays called Business ByDesign. It turned out that business rules are a very important aspect of many modern business applications. Consequently, the investigation project turned into a development project. None of the existing tools and engines was powerful enough or had an architecture that allowed developing it to the envisioned world class rules engine. That was the time when Carsten Ziegler got the responsibility for the project and the architecture of BRFplus.

At that time BRFplus had the working title “Formula & Derivation Tool” (FDT). This is the reason why all development objects such as database tables and classes still use the abbreviation FDT. Some objects use FDT in the description even today.

Although there were limitations and problems with the existing tools and engines, it was always very clear that these tools were hosting an impressive wealth of experience and expertise that had to be preserved. There were many use cases and different approaches to solve the problems that had to be understood before the design work for BRFplus could be started. The following list contains the important tools and engines at that point in time:

- Business Rules Framework (BRF)
- Condition Technique
- Costing Formula Builder
- Derivation Tool (and clones)
- Formula Builder (Fobu)
- Hierarchical Derivation Service (HDS)
- Internet Pricing and Configurator (IPC)
- Rule Modeler
- Validation, Substitutes, Rules (VSR)
- Workflow Rules

A new architecture must not encompass all the features and approaches. Instead, it must include all the approaches and features that turned out to work fine and leave out those that create problems. Very quickly, BRF was identified as the framework with the best approach since it was built in a way that allows extending it easily by additional functionality. Extension is part of the basic architecture of BRF. The possibility of modular extension is essential for many users. A generic component will never be able to serve well all possible use cases because the developers of the generic component simply cannot tackle all of these uses cases at the time of development.

Therefore, BRF was the tool that had gained great acceptance in many different applications although it had originally been developed for an insurance solution. So Carsten Ziegler asked the developers of BRF (Matthias Jordan and Lars Biewald) to support the new project and discuss the BRFplus features in more

detail and how BRFplus could leverage the BRF learnings. Lars Biewald even did some of the very early developments in BRFplus.

At an earlier planning stage of the BRFplus project, it was intended to offer a migration tool for migrating rules modeled in BRF to BRFplus. However, after some research and tests, we have canceled this approach. It turned out that, for technical and conceptual reasons, a 1:1 migration would not lead to a desirable result. You would not have been able to leverage the capabilities of BRFplus, and the result would not have been satisfying.

With BRFplus, you can easily reengineer the business rules modeled in BRF, resulting in a comprehensive model that allows for best usage of all BRFplus capabilities. The rules logic can be reused. This makes the manual reengineering process a minor effort and provides much better results than any migration tool could offer.

Releases

BRFplus first was shipped in **SAP NetWeaver 7.0**. This shipment included only few pieces of the infrastructure and had no user interface. It served some pilot customers but was never intended to be used on a large scale. Since then, many incompatible changes were introduced. We therefore highly discourage to use BRFplus in that release.

SAP NetWeaver 7.0 EHP1 included the next shipment of BRFplus. The basic infrastructure could be completed and a UI was provided. But still, there were quite some limitations, such as missing features, poor usability, and stability of some components. Simple scenarios can be supported by this version, using functions, data objects and the expression types Decision Table, Formula, and Case. For more complex scenarios, the recommendation is to use BRF and consider a migration to BRFplus as soon as SAP NetWeaver 7.0 EHP2 is available. If in doubt, you may get back to the authors of this paper and discuss your use case.

You may also find this blog interesting: http://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/11632

SAP NetWeaver 7.0 EHP2 is the first release with a complete and unrestricted shipment of BRFplus. It can be used for use cases of any degree of complexity. Based on this release, the migration pilot projects have been completed and are shipped as part of SAP ERP 6.0 Enhancement Package 5

The pilot projects are:

- Tax and Revenue Management
- Social Case Management
    - Social Application Processing
    - Social Service Plan Processing
    - Deduction Plan Processing

From this release on, the recommendation is not to use BRF for new projects anymore but to use BRFplus instead.

Comparison

Although many concepts used in BRFplus have been taken over from BRF, BRFplus was written from scratch. It does not share any code with BRF. Hence, a clean slate approach was possible. There was no reason to compromise on UI or API level.

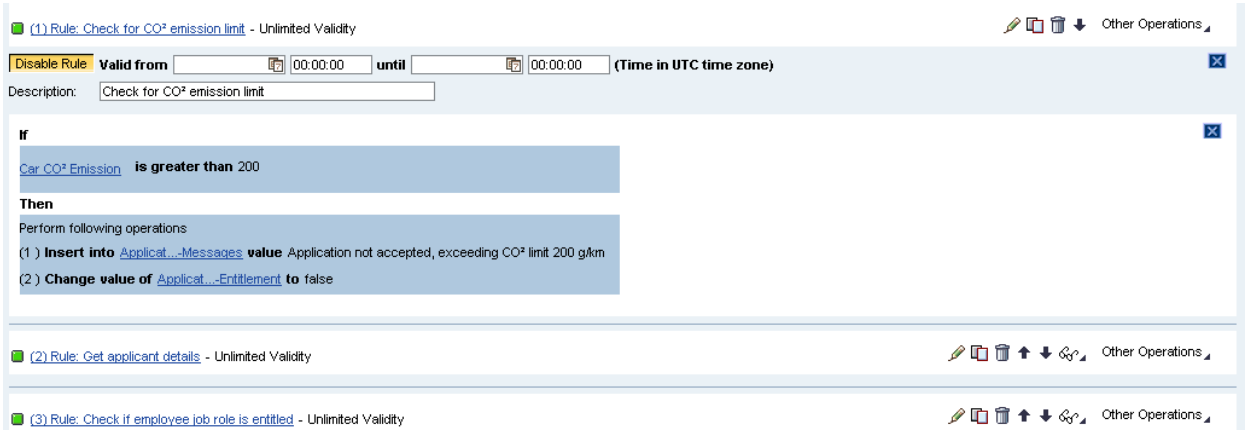
Concepts

Before we take a look at the artifacts, let's first shed some light on important differences in concept.

User Interface

BRF	Dynpro
BRFplus	WebDynpro ABAP

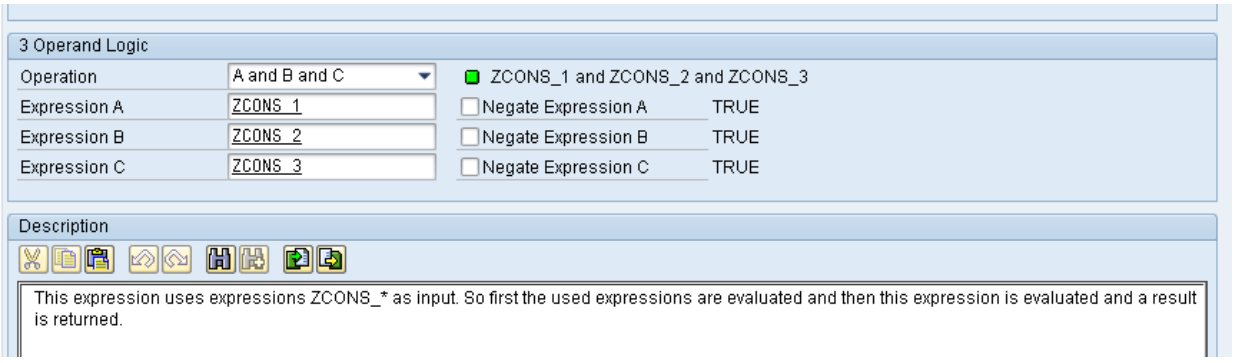
At the time when the major part of developing BRF was done, WebDynpro ABAP was not yet available and the applications using BRF all had Dynpro UIs. In the meantime, SAP has invested heavily into development of new UI technologies such as WebDynpro ABAP. New applications are often built using the new technologies. Consequently, BRFplus has a UI that is implemented in WebDynpro ABAP. However, this is not only a question about running in a web browser or not. WebDynpro ABAP also provides great new features to build highly flexible UIs such as needed for dynamic composition of rule sentences. BRFplus is for sure one of the most dynamic SAP applications built with WebDynpro ABAP.



Rule Execution

BRF	Interpretation
BRFplus	Code generation and interpretation

BRF has an infrastructure to combine artifacts in a flexible way. For example, a Three Operand Arithmetic expression may nest another one. At execution time, the nested expression is evaluated first and its result is returned to the nesting expression. Then, the nesting expression is evaluated.



This concept has been taken over to BRFplus. However, the BRFplus architecture adds the capability of code generation following a similar approach. Each expression returns a piece of code. The generation framework makes sure the pieces fit together. Matthias Jordan, the architect of BRF, always dreamt of

adding code generation capabilities to BRF. This dream has become real with BRFplus. The built-in code generation allows drastically reduced processing times for the execution of rule sets in the range of microseconds. This is totally impossible with any non-generating approach. Usually, data collection as a preparation for the execution call into BRFplus takes more time than execution of the rules themselves.

Further new feature of BRFplus are the ability to have detailed execution traces in both interpreted as well as generated rules, and the so-called “time-travelling” feature. This feature allows processing of rules with definitions as they were in effect in the past. The caller provides a timestamp and all definitions that have been valid at that point in time are used for rule execution.

Context and Result

BRF	Context access by nested expressions
BRFplus	Direct context access

Early versions of BRF did not know a context. Instead, all the data needed for rule execution had to be read with function modules by definition of expressions. Each of the expressions returned exactly one piece of data. Although a so-called context has been introduced in the meantime, the concept is still fundamentally different from the context used in BRFplus. In BRF, the context has to be accessed with the help of expressions. This approach results in a big number of helper expressions that serve the only purpose of accessing the data in the context.

In BRFplus, input data is called context and result data of the rules execution is called result. All actions and expressions can directly use context and result data. There is no need to define helper expressions. Technically, for an expression instance, accessing a data object in the context is not different than the usage of the result from a nested expression. By this approach, the usability for the rule modeling is improved and users have a better overview of the used expressions. Helper expressions are not needed at all.

Persistence Layer

BRF	DB tables with delivery class E; client-dependent, manual transport recording; no reuse
BRFplus	DB tables with delivery classes S,C, and A; local and non-local, client-dependent and cross-client, manual and automatic transport recording; reuse possible

BRF saves content in database tables with delivery class E (control table, SAP and customer have separate key areas). An object is uniquely identified by its combined key: application, category, class ID, object name. The object name is a technical name that needs to fit to the pattern defined for the E tables (name space separation). A reuse between applications is not possible.

BRFplus is able to save content in the following delivery classes:

- A - Application table, master and transaction data
  Properties: local, client-dependent, no transport recording
- C - Customizing table
  Properties: optionally local, client-dependent, transport recording depending on client settings
- S – System table
  Properties: optionally local, client-independent, transport recording depending on client settings

An object is uniquely identified by its UUID. Objects can have names according to customer requirements. All objects are created as part of an application. The application defines how and where the object is saved as well as the transport behavior. A reuse between applications is easily possible and the visibility can be defined on single object level.

Data Types

BRF	Elementary types (various, similar to ABAP)
BRFplus	Elementary (various. simplified), structure and table types; support of deeply nested types

In BRF, only elementary data types are supported as result of an expression. The types are a selection of the ABAP types, such as Integer Number, Numerical Character, Packed Number, Characters, Floating Point Number, String. Additionally, there is a Boole type which does not exist in ABAP.

In BRFplus, the type system is simplified compared to ABAP. The following types exist:

- Boolean
- Number (definition of length, decimals, sign is optional)
- Text (optional definition of length)
- Amount (2 components: number and currency, length, decimals, sign is optional)
- Quantity (2 components: number and currency, length, decimals, sign, dimension is optional)
- Timepoint (several components for optional definition of time point types such as date, time, date and time, timestamp)

The simplification of the types reduces the complexity for the users. Hence, users do not need to care for the difference between String and Character or Packed and Float.

API

BRF	Mixture of UI and Backend in the API
BRFplus	Decoupling of UI and backend logic by clearly defined API

BRF has an API which can be used for the maintenance of BRF objects. However, there is no clear separation in the API between UI and backend methods which creates some additional complexity when working with the API only.

In BRFplus UI and backend are decoupled and the API can be reused very easily for rule maintenance in background or creation of an alternative UI. Nearly all methods used in the BRFplus standard UI are disclosed for usage.

Building Blocks

Not only are there changes in the concepts but also the main building blocks are different in BRF and BRFplus. The following table maps the terms used in BRF and BRFplus.

BRF	**BRFplus**
Application Class	Application
Action/Expression Type	Action/Expression Type
Event	Function
Rule Set	Rule Set
Rule	Rule
Object Group	Catalog

Application Class and Application

The BRF application class is very similar to the BRFplus application. The only difference is that a BRF application class can contain objects from SAP and from customers whereas in BRFplus, an application (including its objects) has exactly one origin. BRFplus does allow for reuse across the application border. This is not supported in BRF. Additionally, in BRFplus there are multiple attributes to define default settings for newly created objects.

Action/Expression Type

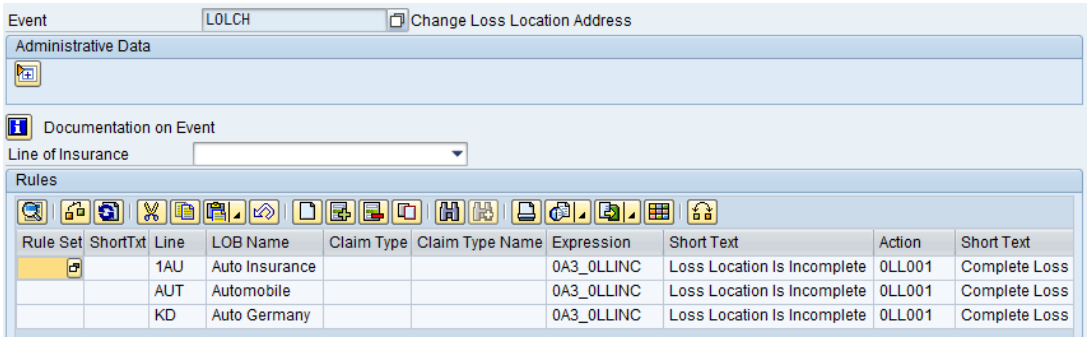
Expression types define the computational power of BRF and BRFplus. Each expression type defines a self-contained computational unit with a well-defined logic. An expression can be considered to be an instance of an expression type behaving according to the expression type's logic. Actions are special expression types for definition of interactive parts. Both BRF and BRFplus support the creation of custom action types and expression types which is the prerequisite for powerful enhancements and optimal adaptation to use cases. The concept of action types and expression types is very similar in BRF and BRFplus. The BRFplus architecture is an improved copy of the BRF architecture.

A difference between BRF and BRFplus is the ability not only to nest expressions (take the output of one expression as input to another one) but also to directly use context in BRFplus. For the implementation, it is not relevant and does not lead to a different behavior if an expression is nested or data from the context. This shows already that there are changes in the implementation of action and expression types between BRF and BRFplus that go far beyond changes in inheritance or the level of service provided for redefinitions.

Another obvious difference is the number and the capabilities of expression and action types provided in the standard. Although BRFplus types have been inspired by BRF types, the features provided in BRFplus are by far richer and have a better usability than in BRF (such as BRFplus expression types Decision Table, Table Operation, Procedure Call). Finally, BRF expressions can have an elementary result. In BRFplus, an expression may return anything up to deeply nested tables.

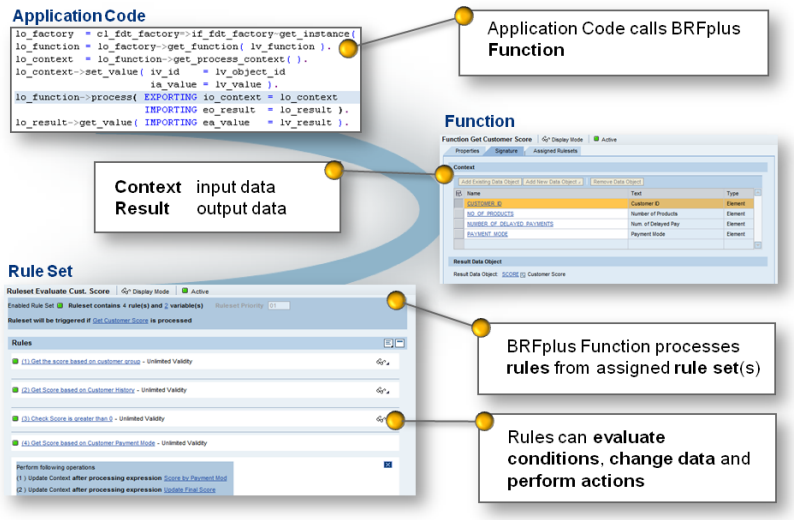
Event and Function

In BRF, the event is the central entry point for the execution of business rules. Any number of rules can be defined for an event. Rules can also be grouped in rule sets and assigned to events. The number of rules which have to be processed during runtime is determined dynamically.



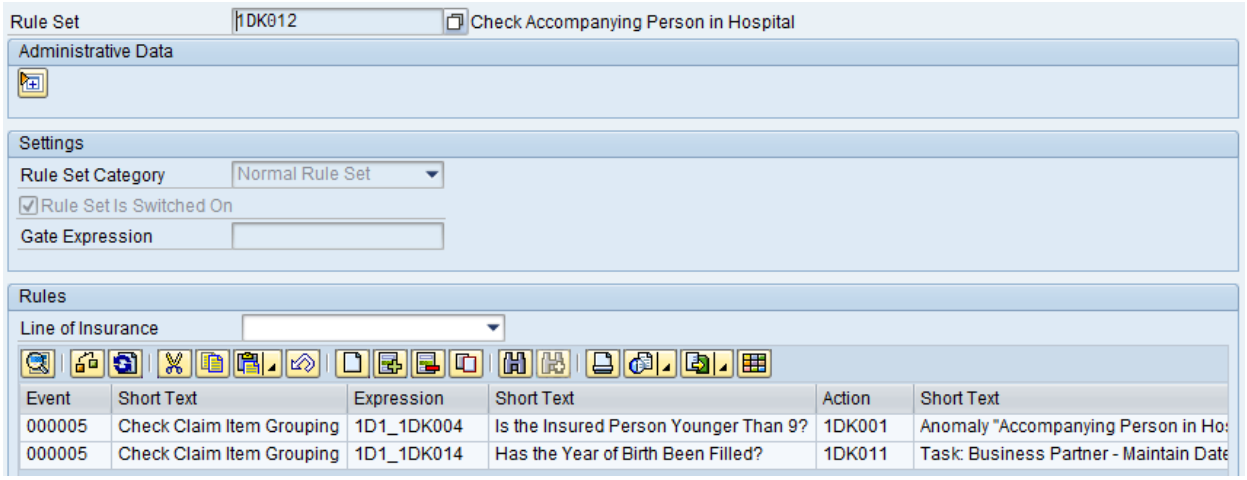
Rule Set	ShortTxt	Line	LOB Name	Claim Type	Claim Type Name	Expression	Short Text	Action	Short Text
		1AU	Auto Insurance			0A3_0LLINC	Loss Location Is Incomplete	0LL001	Complete Loss
		AUT	Automobile			0A3_0LLINC	Loss Location Is Incomplete	0LL001	Complete Loss
		KD	Auto Germany			0A3_0LLINC	Loss Location Is Incomplete	0LL001	Complete Loss

In BRFplus, a function is the link between application code and BRFplus rules. It defines an interface consisting of ingoing data called context and returned data called result. Whenever a function is called, all rules in the assigned rule sets are evaluated. The separation of rule set and event provides the possibility to have more than one rule set assigned to a function or to have rule sets assigned from other applications and even with a different lifecycle behavior. You may create the function as a system object but create a rule set as a customizing object.



Rule Set and Rule

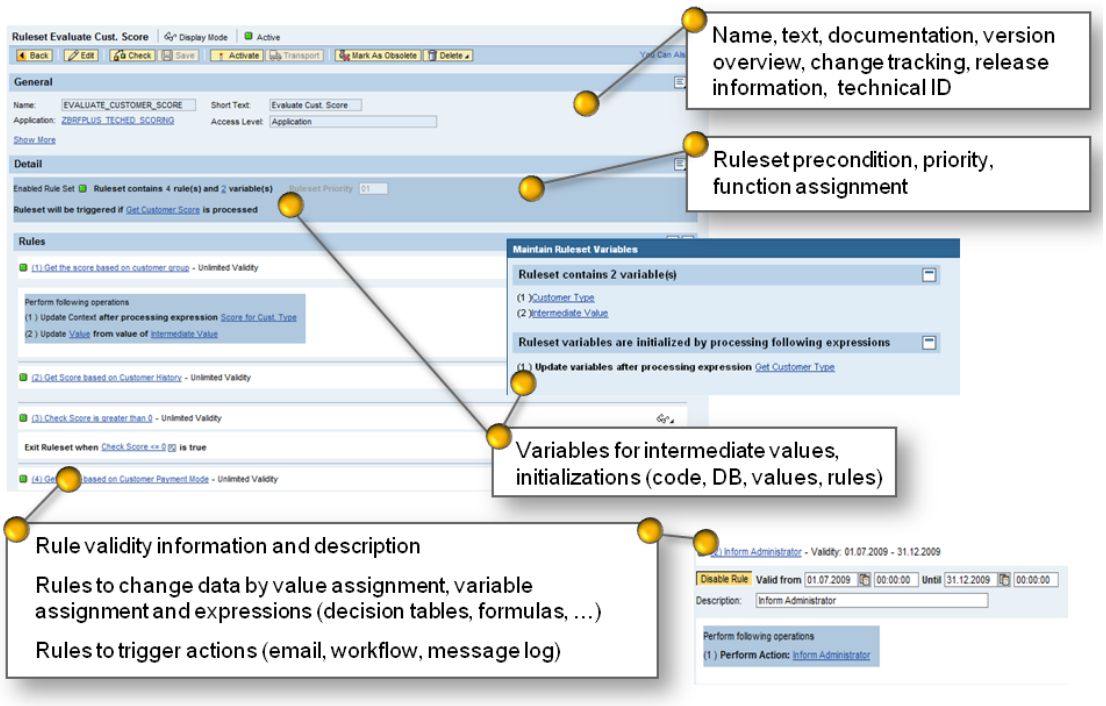
A rule set in BRF can be used to group rules that belong together from a business point of view. The rules may be used in several events. There a different types of rule sets for different runtime behavior including date expressions and switches. A rule in BRF has basically three attributes: the event, a Boolean expression and an action. When the rule is called in the corresponding event and the Boolean condition is true then the action part is performed. Rules in BRF are not independent objects that can be reused.



In BRFplus, the idea of the rule set is very similar to that in BRF. A rule set contains a collection of rules. As opposed to BRF, in BRFplus the complete rule set is assigned to a function. The rule has some more attributes such as switch (enabled/disabled), precondition, variable definitions, and initialization expressions. A rule in BRFplus consists of an optional condition (expression with Boolean result or context object of Boolean type). If evaluated to true, or in case there is no condition, the Then part of the rule is performed. If evaluated to false, the Else part of the rule is performed. The rule is able to perform the following operations:

- Trigger action
- Process expression and the result can be used to update context/result
- Initialize or assign values to the context/result

As opposed to BRF, a rule in BRFplus is an independent object. It may be used in more than one rule set.

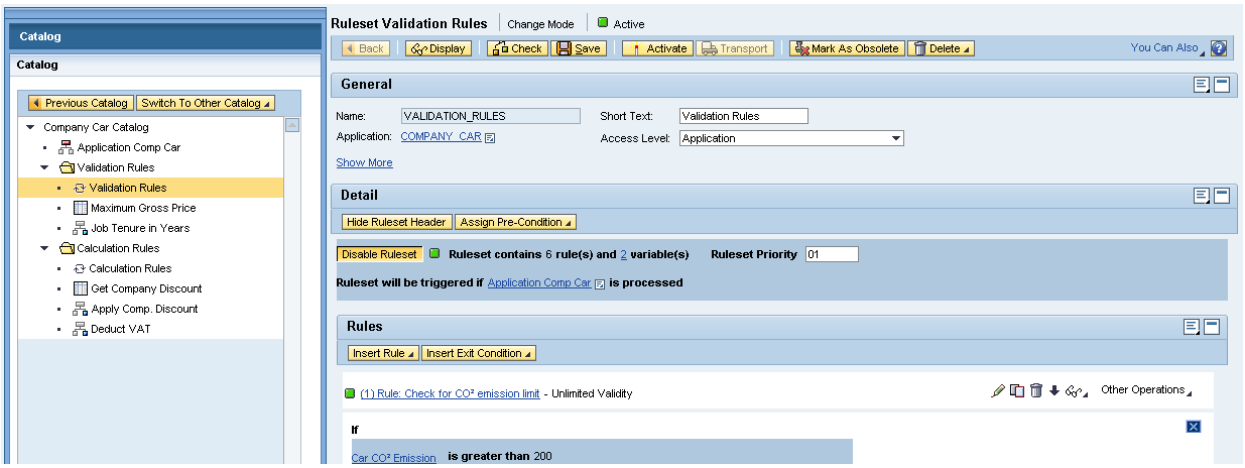


Object Group and Catalogs

Object groups are used in BRF to organize BRF objects to simplify usage and navigation of the objects. Groups can be nested. They are not relevant for the rule execution.

Application Class	Application Class
FLIGHT	Application Class:
All Groups	All Groups
Group :0CONTEXT	
Group :0DECISIONTREE	
Actions	Actions
Events	Events
TESTEVENT_AW	
Expressions	Expressions
0AMAOUNT_C0	
0AMAOUNT_C1	
0AMAOUNT_C2	
0AMOUNT	
0AMOUNT_C3	
0AMOUNT_F1	Posting total < 1000 Euro
0AMOUNT_F2	Posting total < 5000
0AMOUNT_F3	Posting total >= 25000
0AMOUNT_F4	
0AMOUNT_F4_TEST	
0DECISION	Simple decision tree
0FL_PLANETYPE	Aircraft Type
0PLANETYPE_F1	
0PLANETYPE_F2	
0SIZE_FUNNY	Posting total funny size
0SIZE_HUGE	Posting total huge size
0SIZE_LARGE	Posting total large size
0SIZE_MEDIUM	Posting total mid size
0SIZE_SMALL	Posting total small size
0SIZE_UNKNOWN	Posting total unknown
3423412	
TEST_AW	
Rule Sets	Rule Sets
Contexts	Contexts
Group :0FLIGHT	Flight
Group :0FLIGHTPLAN	Flight

In BRFplus, there is a comparable concept called catalog. A BRFplus catalog allows organizing BRFplus objects from one or many applications. Similar to the BRF group, a BRFplus catalog supports links with which other catalogs or parts of them can be nested. Additionally, a BRFplus catalog also provides the possibility to create structure nodes (folders) for hierarchical structuring, as well as the attachment of attributes on node level. The main idea of the catalog is simplification of the user interface for users who are more focused on business topics than on technical details. Exposing the complete BRFplus repository to a business expert may be very confusing. Consequently, a BRFplus catalog is shown in a separate view in the navigation panel with the option to switch off the other views and take out complexity by removing technical artifacts, features and complex screens.



Related Content

BRFplus

Carsten Ziegler's Blog

Business Rules Management

For a detailed introduction to BRFplus please visit Business Rules Management at the SDN network and select 'BRFplus' in the Related Content section.

Copyright