Business Rules Framework plus – The Very Basics

Applies to:

Business Rules Framework plus shipped with **enhancement package 1 for SAP NetWeaver 7.0**.

Summary

The paper introduces the Business Rules Framework plus. The core entities, concepts and terminology such as function, expression, context, result, ruleset, and rule are described in the paper.

Author: Carsten Ziegler, Thomas Albrecht

Company: SAP AG

Created on: 22 June 2008

About the Authors

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

Thomas Albrecht is a Developer in the Business Rules Framework plus team. Since joining SAP in 2003, he has worked in and driven several development projects in ERP Travel Management with focus on architecture.

He joined the BRFplus development team in 2008.

Table of Contents

Business Rules

Business rules are logical statements that determine and control the business functions of an organization. They enable an organization to achieve its goals by describing the operations, definitions and constraints. Business rules include basically everything that runs a business, for instance, business habits, manuals, policies, lines of computer code, and minds of experienced employees.

Managing business rules, which are subject to frequent changes because of an agile environment, is the biggest challenge organizations face today. The evolvement of a Business Rules Management System (BRMS) helps organizations maintain flexibility in administration and maintenance and a real-time responsiveness to changes in business requirements.

BRMS allows you to capture, store and manage rules in a transparent way. Business rules are separated from the application code. With the externalization of rules, organizations no longer have to depend upon IT experts to apply changes to the policies.

The use of a BRMS leads to the following advantages:

- Higher degree of transparency
- Less coordination efforts between IT experts and business experts
- Less testing and maintenance costs
- Reduced modification cycle times
- Quicker response to changes
- No downtime needed because code change is not required
- Clear segregation of duties – code (IT department) and business rules (business expert)
- Quicker application development because business rules are a powerful means for flexible application customizing
- Convenient graphical representation

Usage Scenarios

Applications that make use of BRMS such as Business Rules Framework plus (BRFplus) are designed in such a way that the stable parts remain in the application and the parts that are subject to frequent changes are delegated to the BRMS. The stable parts may include core processes and abilities of the application. The **flexible parts** may comprise some of the following:

- **Decision Services**
  Automating procedures such as claims processing, customer service management, credit approval
- **Data Validation and Error Detection**
  Diagnosis with monitoring/alerting and detection of invalid data and states
- **Classification and Derivation**
  Classifying and deriving customers, products, or risks
- **Matching**
  Matching responsibilities, suitable products, or locations
- **Calculation**
  Calculation of costs, overheads, risk, surcharge

The significance of BRMS can be gauged from what experts (Gartner's Jim Sinur) have to say:

- "…customers are keen to move to service-oriented architectures capable of rapidly implementing process change by using rules-based engines."
- "Just as yesterday we separated client-server interface from process and from data, we are today separating rules from workflows and services."
- "Rules can soften the blow from unstoppable change …"
- "Rules can exploit incremental accumulating knowledge instantly"

Business Rule Framework plus

BRFplus is a rule engine developed in ABAP. It provides a comprehensive application programming interface (API) and user interface (UI) for defining and processing business rules. It allows rules to be modelled in an intuitive way. Rules can be reused in different applications. BRFplus supports features like simulation, trace, transport, XML export and import.

Applications use BRFplus in the following scenarios:

- Validation of data and detection of invalid data and states
- Matching responsibilities, suitable products, and locations
- Calculation of costs, overhead, and risks
- As a technical configuration engine

BRFplus comes with a component-based UI built on WebDynpro ABAP that suits both business users and IT experts. It provides two modes: workbench and catalog browser.

The workbench enables both developers and business rules experts to maintain rules. In the workbench, one can access the *Repository* view in which all objects are visible and accessible. The workbench provides the complete functionality of BRFplus.

In the catalog browser mode, a business user can browse through the content of a catalog and maintain assigned business rules. Developers and business rules experts define the context of the catalogs for a business case. The catalog browser hides the *Repository* view.

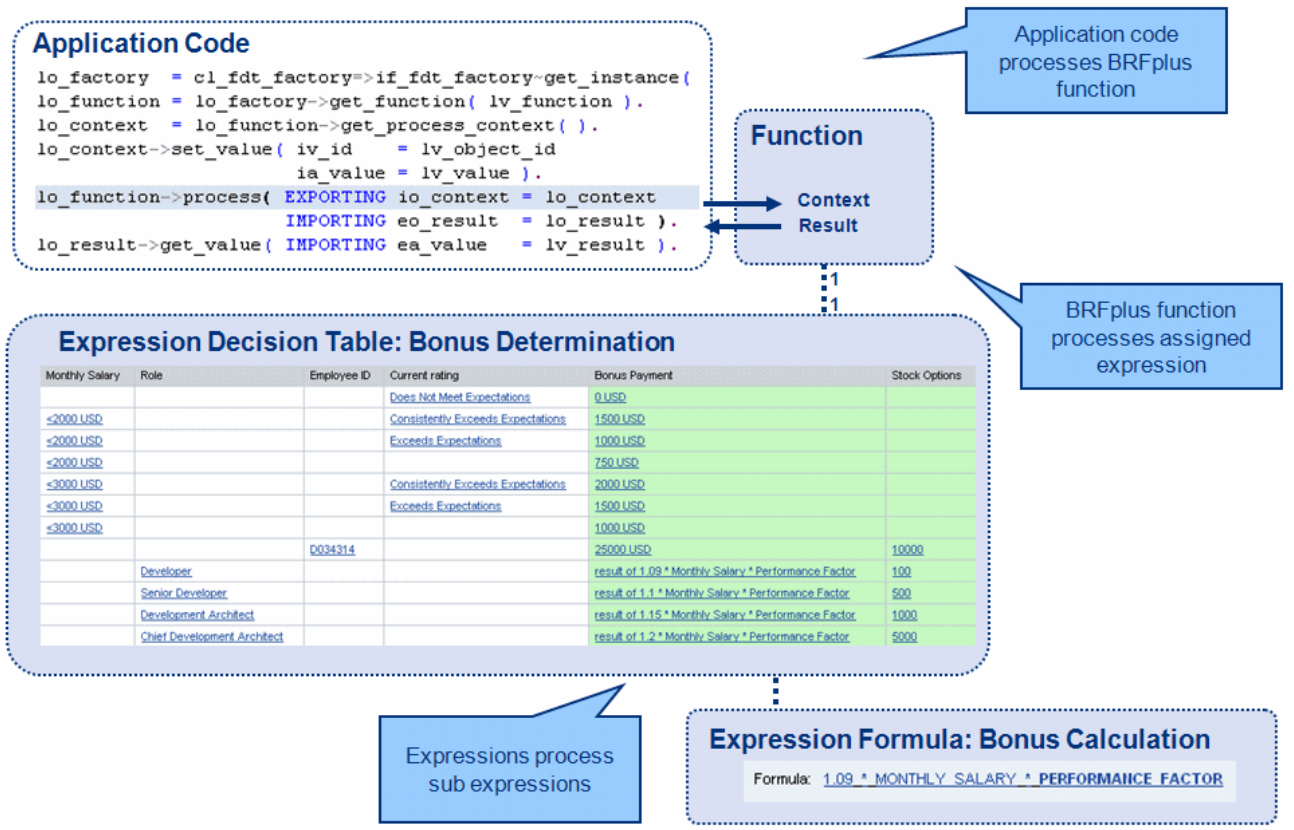
Mode of Operation

In BRFplus, a rule is invoked through a function. The function uses a context, a collection of data objects, that represents the interface for data exchange with the using application.

Expressions that are assigned to the function define the actual outcome of the business rule.

BRFplus provides three modes of operation for processing a function. They are **Functional mode**, **Event mode**, and **Functional and Event Mode**.

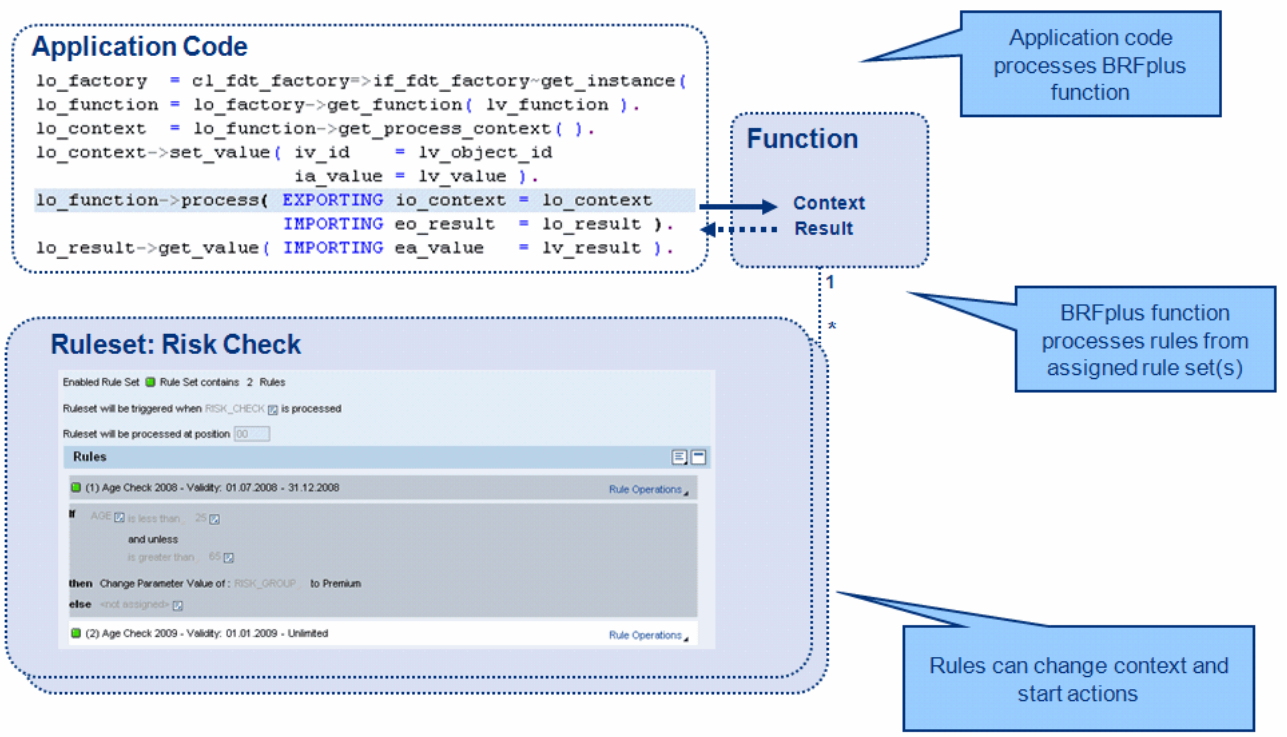
Functional mode requires a top expression to be assigned to the function. During processing, the function triggers and evaluates the top expression. The top expression may use a number of other expressions to find a result. The result of the top expression is the result generated for the function. For example, a decision table can be assigned as a top expression to a function. In the input columns of the decision table, conditions are formulated based on the context data objects. For a matching condition, the decision table may return a constant or nests a formula expression. The result of the decision table is returned as the function result. The functional mode is typically used for more simple use cases with a direct output of the function.



The image above shows a function in the functional mode. The function calculates the bonus of an employee. The function has a decision table, *Bonus Determination* as its top expression. Inside the decision table there are data objects of element type, result of structure type, case expression and formula expression. The result that is returned for the decision table acts as the result of the function.

In the **Event mode**, a function does not directly process a top expression. Instead, it triggers an event that can be subscribed by a number of rulesets. A ruleset is a collection of rules whose validity can be restricted to a specific time period. A rule consists of a condition and two possible action definitions – depending on whether the condition is met or not. An action can change values in the context or it can trigger additional actions, such as starting a workflow or writing error messages to a log.

Triggering a function in the event mode results in the processing of a number of subscribed rules and their actions, provided that the corresponding conditions are met. The event mode is generally used for more complex use cases where a direct result is not needed.

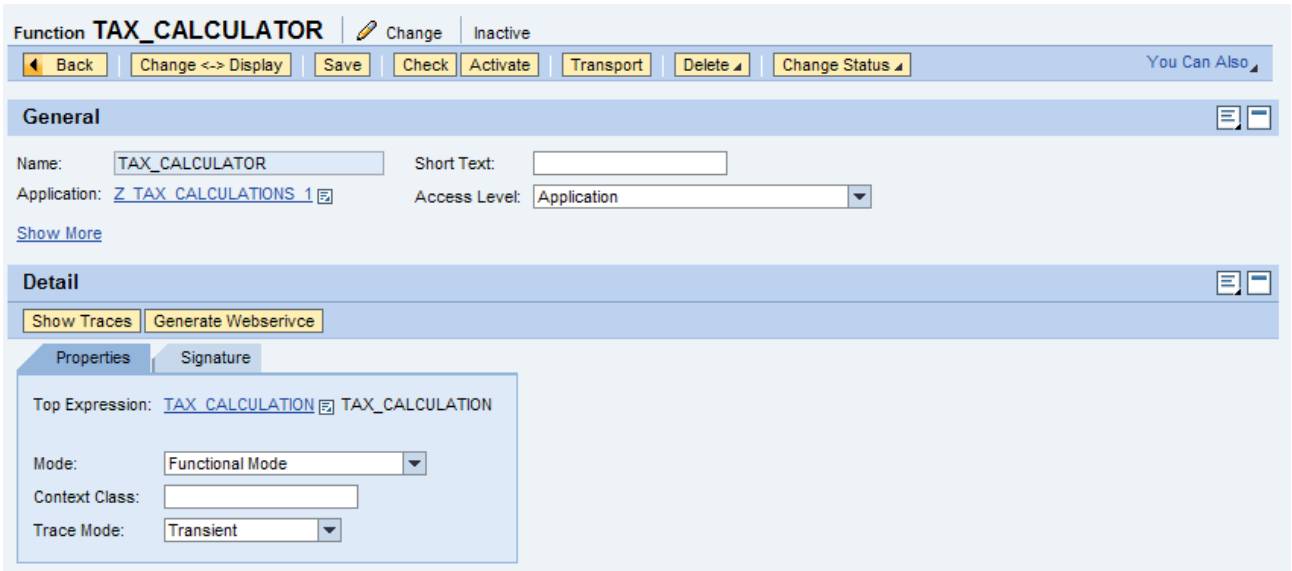


The image above shows a function in the event mode. The function determines the risk factor of a credit card applicant. Instead of a top expression, the function processes rules from the assigned ruleset.

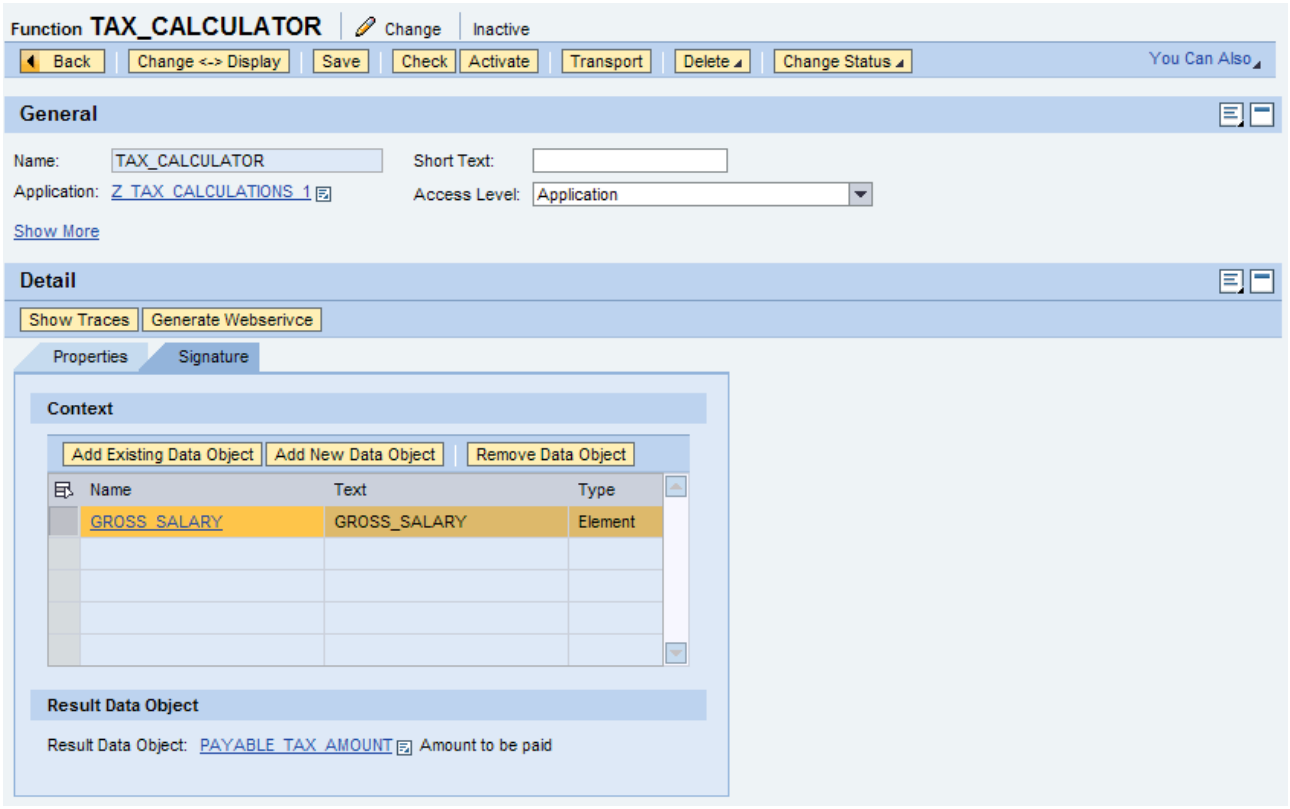
Functional and Event mode is a mixture of the functional mode and the event mode. It works with a top expression and a result is returned. However, rules can be fired so that actions change the context.

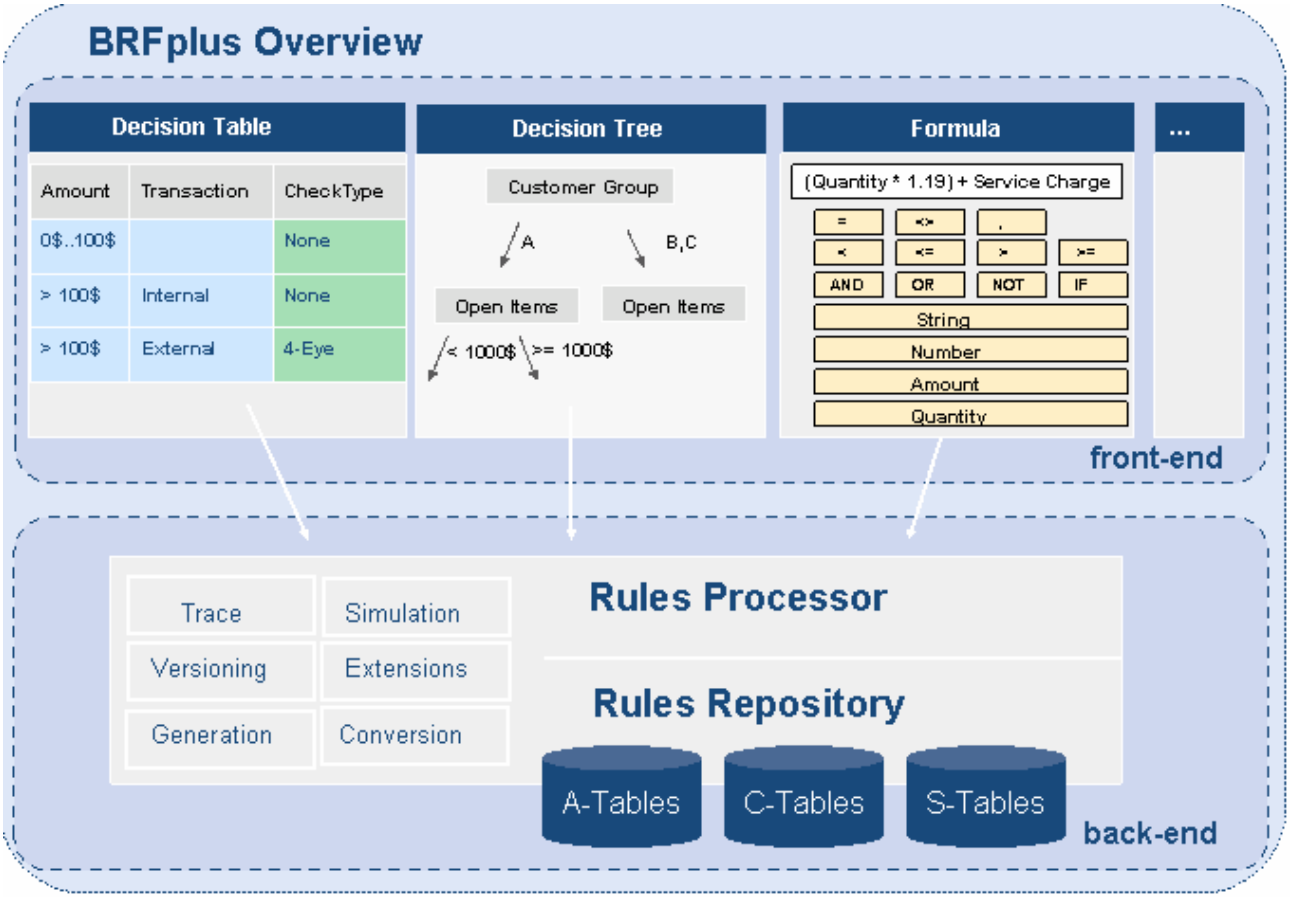
Application to Rules Interface

The function is the interface between the application code and the business rules. The application does not need to take care of the business logic defined in the business rules. It provides the data according to the function definition and may receive the result. The function can be compared to an ABAP function module definition, whereas the rules (the expressions that make the rules) are similar to the ABAP code inside the function module.



Note: The image above shows a function in the functional mode. A decision table has been assigned as the top expression for the function. The image below shows the context and the result assigned to the function.


Function TAX_CALCULATOR
Change
Inactive
Back
Change <-> Display
Save
Check
Activate
Transport
Delete
Change Status
You Can Also
General
Name:
TAX_CALCULATOR
Short Text:
Application:
Z_TAX_CALCULATIONS_1
Access Level:
Application
Show More
Detail
Show Traces
Generate Webserivce
Properties
Signature
Context
Add Existing Data Object
Add New Data Object
Remove Data Object
Name
Text
Type
GROSS_SALARY
GROSS_SALARY
Element
Result Data Object
Result Data Object:
PAYABLE_TAX_AMOUNT
Amount to be paid


BRFplus Overview
Decision Table
Amount
Transaction
CheckType
0$..100$
None
> 100$
Internal
None
> 100$
External
4-Eye
Decision Tree
Customer Group
A
B,C
Open Items
Open Items
< 1000$
>= 1000$
Formula
(Quantity * 1.19) + Service Charge
=
<>
<
<=
>
>=
AND
OR
NOT
IF
String
Number
Amount
Quantity
...
front-end
Trace
Simulation
Versioning
Extensions
Generation
Conversion
Rules Processor
Rules Repository
A-Tables
C-Tables
S-Tables
back-end

Components

Application

An application is a top level container for all kinds of different BRFplus objects. All BRFplus objects are assigned to exactly one application.

The application bundles objects involved in a specific usage scenario. It imposes some common technical settings such as development package, software component and application component.

The application component and software component have to be the same as defined for the development package. The application and software component are automatically derived from the development package if they are not set.

Application resides at the highest level in the classification hierarchy for objects. Application-dependent objects such as functions, expressions, or actions are created in the context of an application. All objects are assigned to the application.

There are three types of applications – **Customizing** application, **Master Data** application, and **System** application. **Customizing** and **System** applications can be local, but **Master Data** application is always local. All three types of application can be assigned to a development package. The type of application can be set when a new application is created.

Customizing Application

Customizing applications are objects in which users are allowed to create or change (customize) the objects in their system environment. Technically, the meta data for customizing objects is always stored in client-dependent tables of delivery class C.

Master Data Application

Master data applications are always local objects. The user is allowed to create or change the objects but the objects cannot be transported. The meta data is stored in client-dependent tables of delivery class A. Master data objects can refer to system and customizing applications but master data applications cannot be referred to by customizing or system applications.

System Application

System applications do not allow users to make changes to the objects. The meta data is stored in client-independent tables of delivery class S. This meta data can only be transported via a workbench transport.

All system object DB table names end with an S. System application cannot refer to customizing or master data applications but they can be referred to by customizing or master data applications.

Data Objects

A data object behaves like a variable in a programming language. It defines a data type and is associated with a name. In BRFplus, data objects work as a data carrier in the context or as storage for the result of an expression or action. Data objects can be of element type, structure type or table type.

Context and Result

Context consists of a set of data objects. They are typically used as input parameters for expressions and actions, but can also be manipulated during processing (for example, Context-Change actions).

For the result of an expression or function, the data object is used as an output parameter.

Expression Types

Expression types define the computational power of BRFplus. Each expression type defines a self-contained computational unit with a well-defined logic.

BRFplus comes with a set of common expression types and will be continuously enhanced by new expression types. You can create your own expression types and use them in rules.

Types of Expression

Case	The case expression is similar to the ABAP CASE statement. The expression takes a case parameter and checks its value against a number of test parameters.
Constant	The constant expression is the most simple expression. It returns a fixed value that may be used by other expressions.
Random	The random number expression provides two different functionalities. On the one hand, the expression can be used to return a random number between a minimum and a maximum value. On the other hand, it can check whether a specified probability between 0 and 1 is met and accordingly returns a boolean value.
Static Method Call	The static method call expression calls a static method on an ABAP class and implements a special interface.
Value Range	Value range expressions check if the value of a test parameter lies within a certain range. The result of a range expression is always boolean. The expression returns the value as true or false depending on whether the value falls within the defined range.
BRMS Connector	The BRMS connector enables one to use the SAP NetWeaver BRM or any other external rules engine with BRFplus. From the context and result data definition, an XML schema can be generated and imported into the external BRMS. At runtime, the expression uses an appropriate, remote-enabled function module to invoke the external rules engine. The communication is based on the provided XML schema.
Decision Table	The decision table expression sequentially processes rows of a table that consists of condition and result columns. If all conditions in a row are met, the corresponding result values are returned. The decision table can work in two modes: single match and multiple match modes. In the single match mode, the decision table stops processing at the first matching row, thus returning a single result. In the multiple match mode, all rows are processed and the results of all matching rows are collected into a table.
Decision Tree	The decision tree expression allows the definition of a binary search tree. It consists of two types of

	nodes: condition and leaf nodes. Condition nodes use boolean expressions to fork to one of two sub-nodes at runtime. Leaf nodes are the end in a decision chain and provide the result value for the expression.
Formula	The formula expression allows you to perform a wide range of mathematical, boolean and other kind of calculations. It can use any other expression or context data objects to determine the result.
Function Call	The function call expression calls a specified BRFplus function and thus allows functions to be nested into each other. This can result in better reuse and may facilitate modifications for complex expressions. The expression defines a mapping for the context data objects and simply returns the result of the function at runtime.
Rule	A rule consists of a condition and two actions, true action and false action. If the result of the expression is true then the true action is processed. If the result of the expression is false then the false action is processed. A rule checks a condition and fires an action depending on the result. The condition can be a context data object or an expression that provides a boolean result value.
Search Tree	The search tree is built similar to the decision tree. Each node of the search tree has a condition. The leaf nodes have an assigned result. However, the nodes of the search tree can have an arbitrary number of sub-nodes. The search tree is evaluated in a top down approach for matching conditions. If the condition of a node is true then all its sub-nodes are processed. For the leaf nodes, the result is returned. The search tree can be processed in two modes – first match mode and multiple match mode.
Step Sequence	Step sequence processes a number of single sequence steps. The results of the single sequence steps are written to a central work area. The central work area is a data object of type structure. Automatic context mapping of the results of the single steps to the result data object takes place. The data objects are broken down to element level, and the corresponding values from the context are written to the central work area. A pre-condition is evaluated before a step is processed. This pre-condition is an expression which returns a boolean value. If a true value is evaluated, the corresponding step is skipped. During the processing of a single step, the context

	of the corresponding function is filled with values taken from the work area. The result of the single step is written back into the work area.
XSL Transformation	This expression performs an XSL transformation to determine a result. A list of parameters (context data object or nested expression) is passed to the XSL transformation as input. The transformation can either be referenced by name (the transformation has to exist in the system) or it can be derived from a nested expression or context data object value.

An expression can be considered to be an instance of an expression type, behaving according to the expression type's logic. Expressions access context data objects and may nest other expressions to calculate or determine a result. They are the building blocks for rule and function as definitions.

Actions are expressions of an action type. As an additional feature, each action can have an arbitrary number of follow-up actions to define an action chain. The result of an action expression is always a table containing the identifiers of all performed (follow-up) actions.

Expressions and actions may be shipped by SAP or created by partners or customers.

Action Types

Action types are special expression types that define the interactive part of BRFplus. Action types do not have any output. Instead, they define some result-independent processing, based on incoming data. Since action types are highly application-dependent, BRFplus provides only a few generic ones.

Each action can have an arbitrary number of follow-up actions to define an action chain. An action returns a table with all performed actions. Actions may be shipped by SAP or created by partners or customers. Each action belongs to an action type that defines the class and the interface for the action.

Types of Action

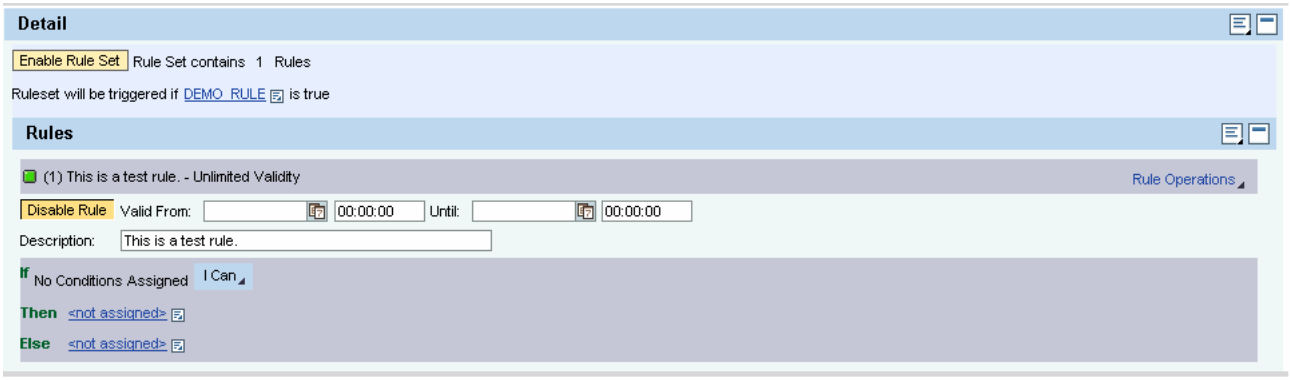
Context Change	The context change action sets the value of a context data object at runtime. Expressions and other context data objects can be used to provide the value to be set.
Message Log	The message log action is used to write messages into an application log. Messages are created from message class definitions in the backend or by entering free text with some parameters. The parameters of the messages can be filled directly or may be derived from context data objects or nested expressions at runtime.
Static Method Call	The static method call action takes a class which implements a special interface and calls a specific static method at runtime. To determine a result, the method can access the context and an optional set of (sub) expressions.

Ruleset

A ruleset handles the processing for a collection of rules. A rule in a ruleset can be enabled or disabled in general. Its validity can also be restricted to a certain time period.

The ruleset subscribes to a number of functions (in event mode) that trigger the processing of the ruleset. As for the embedded rules, the complete ruleset can be enabled or disabled in general. .

A ruleset has no connection to other rulesets that are subscribed to the same function(s). This means when a function in event mode is processed, all rulesets having subscribed to the function as its trigger are processed independently from each other. It is not possible to define an explicit order for the processing of multiple rulesets.



Catalog

The catalog simplifies the rules maintenance process for the business user by hiding the *Repository* view. Only the relevant objects which are necessary for the maintenance of rules in a given business scenario are displayed. The catalog tree structure is composed of nodes of different types: structure nodes, leaf nodes and link nodes. These nodes define the structure of the catalog. The main structure of the catalog is built from the structure nodes which in turn link to an arbitrary number of subnodes. The leaf nodes point to a specific BRFplus object. The content of another catalog can be referred to through link nodes.

Simulation

Simulation allows you to check the outcome of a BRFplus function. It works similar to a function module testing in the ABAP workbench.

Test data can be provided for all the context data objects (importing parameters) of the function. The simulation can be run in two modes:

- **Show Only Result Mode**

  The final result of the complete rules execution is shown.

- **Show also Results of Intermediate Steps Mode**

  A step-by-step analysis of the internal function processing along with the result is shown.

Trace

Trace helps users to access information that is logged during the processing of a function or an expression. The trace results in a step-by-step explanation of the internal processing of a function. The information generated is helpful for error analysis and may even be required for legal reasons. The trace output can be saved at runtime.

Versioning

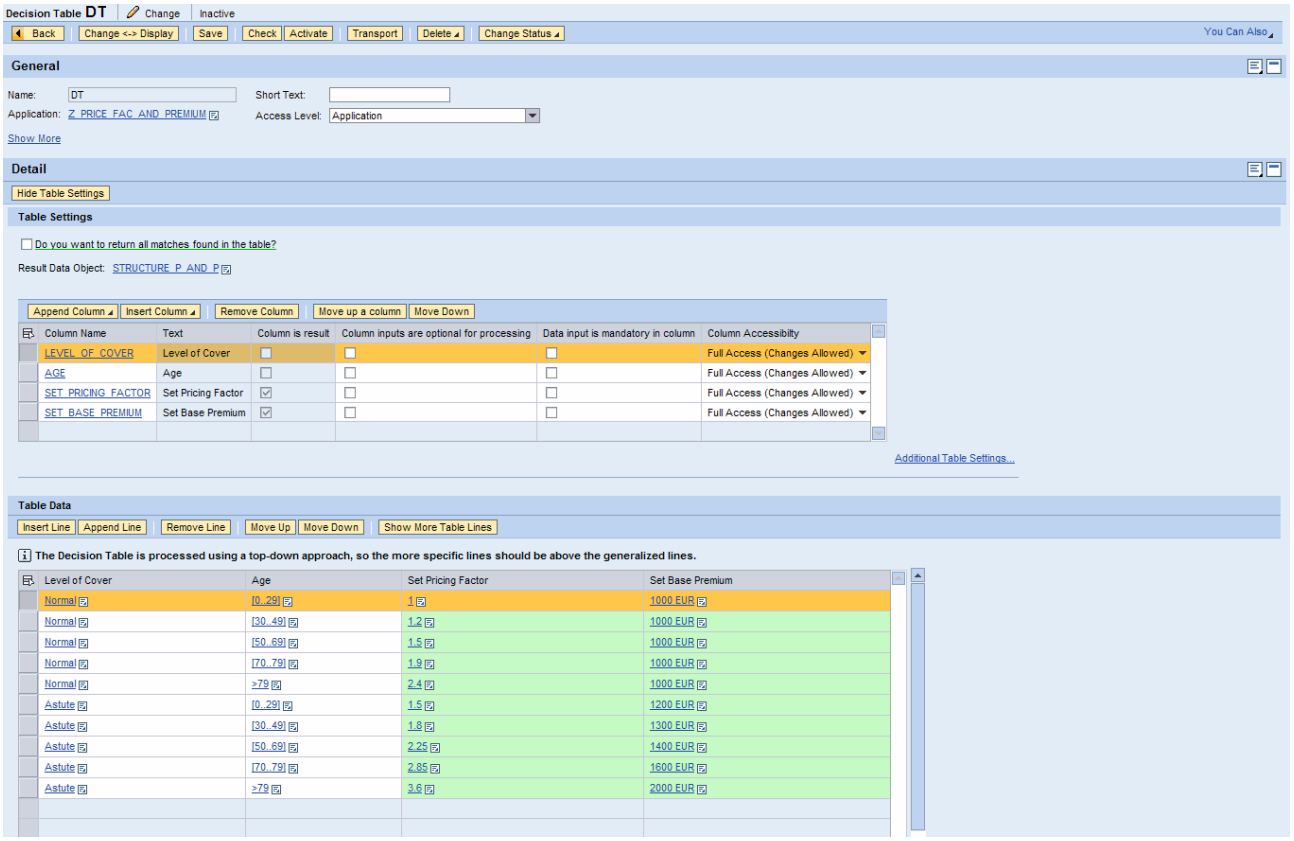
Versioning enables users to track the changes done in the BRFplus objects. If the versioning button is switched on, each time an object is saved and activated, a new version instance for the object is generated automatically.
With versioning, it is also possible to run older versions of an object at a given time point. A typical use case would be the change in a policy with the requirement that older data still needs to be processed in the earlier way and new data shall be processed according to the new policy.

Features

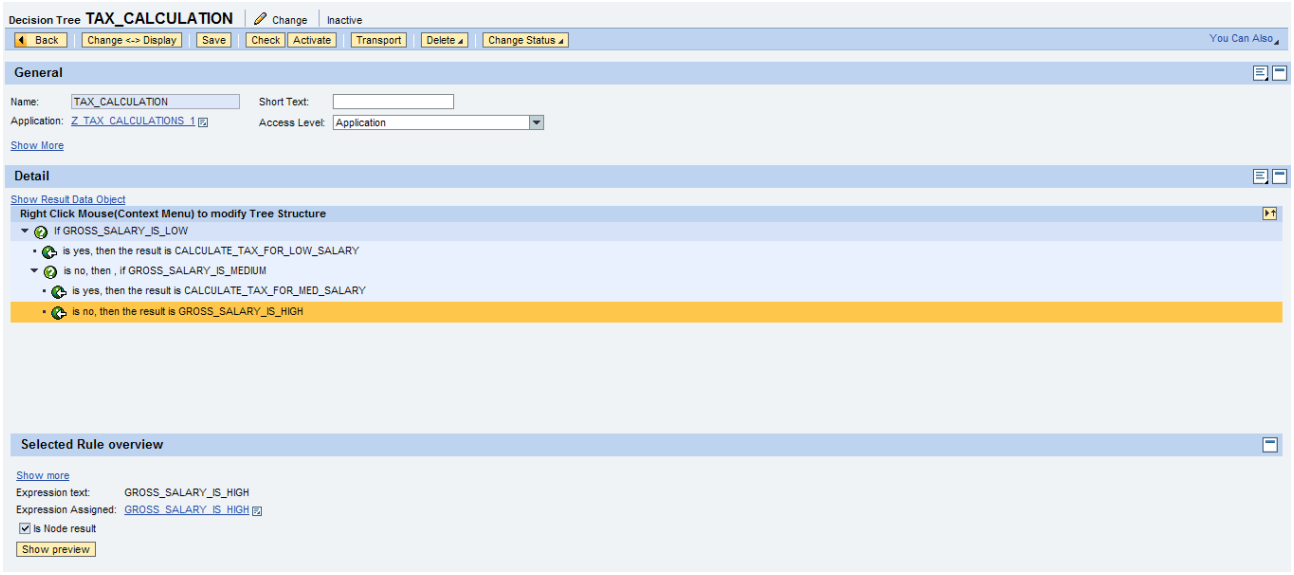
- Various visualization means
  - Decision Table
  - Decision/Search Tree
  - Sequence
  - Formula
  - IF-THEN Rules
- High performance runtime (Code Generation)
- Suits business users and IT experts (can be plugged into other UIs)
- Versioning
- Simulation
- Trace (for legal requirements and for the explanation of results)
- Transport, XML import & export
- Implicit unit and currency conversion
- Object catalogs and filters to organize the rule repository
- Third Party Connector

Screenshots

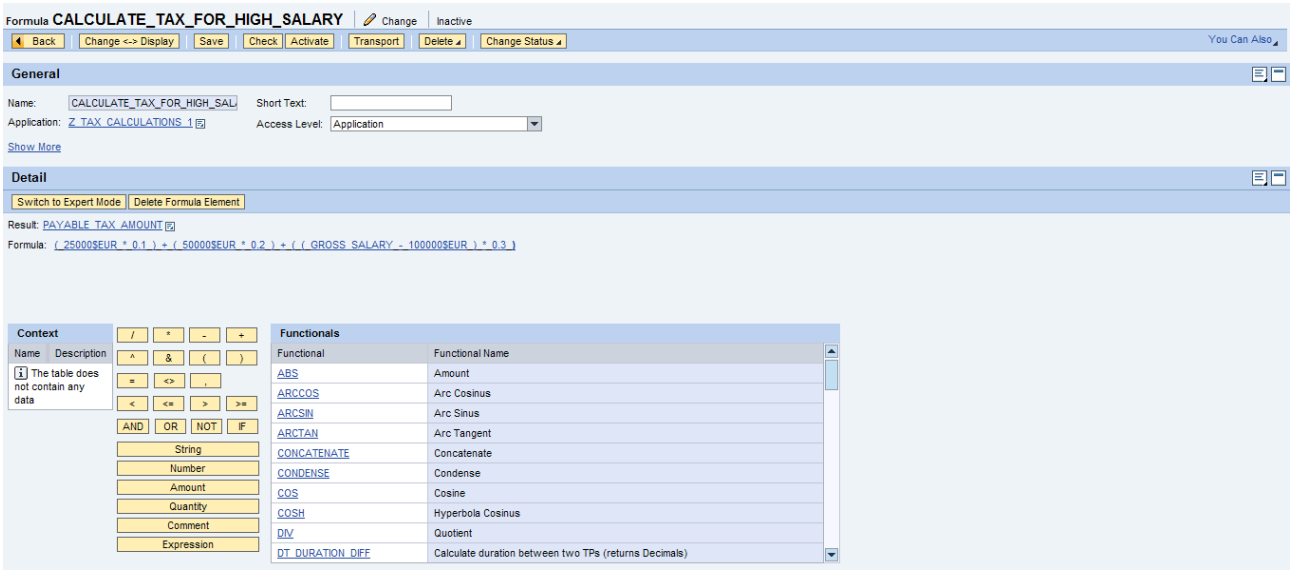
Decision Table Expression



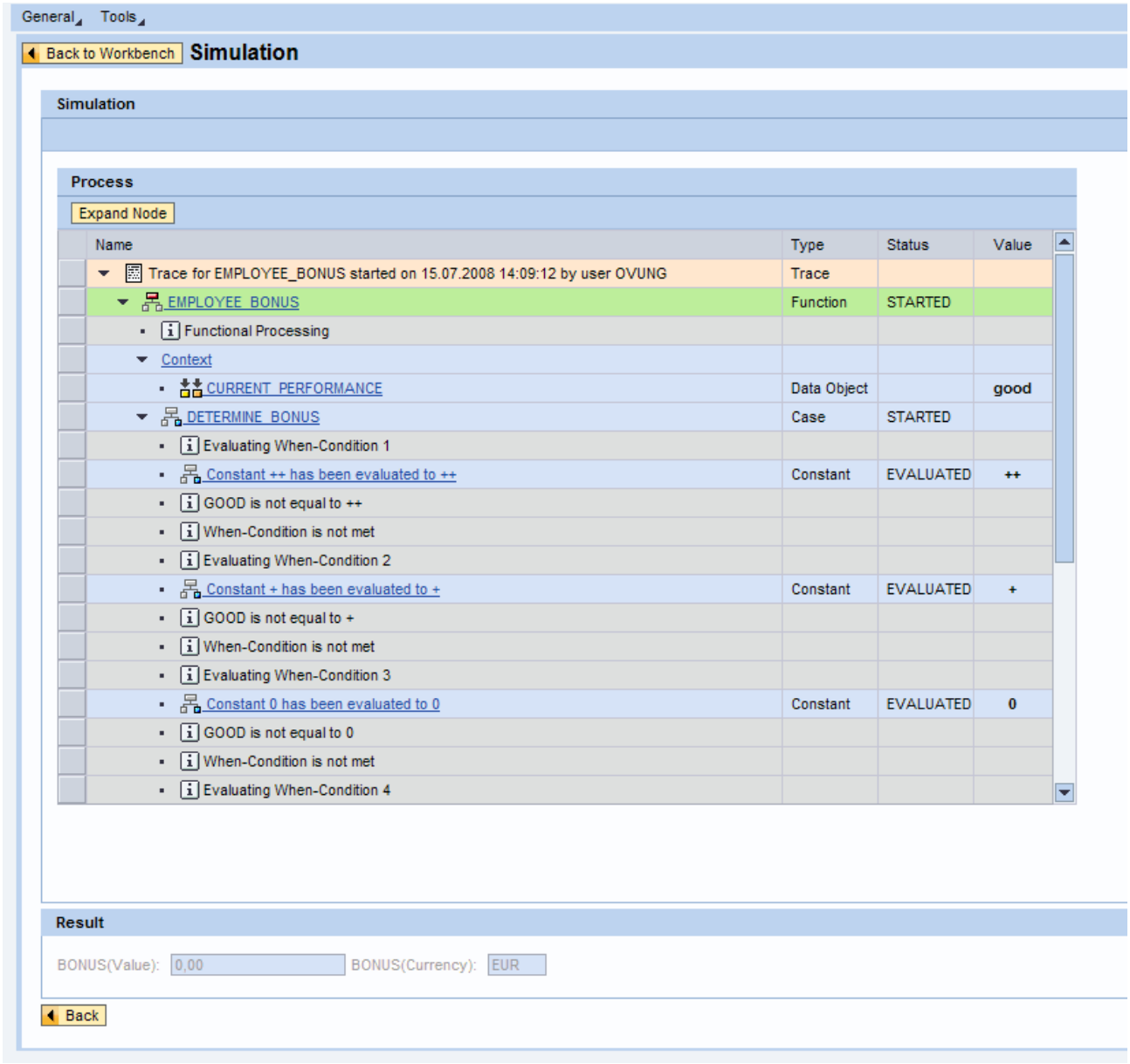
Decision Tree Expression



Formula Expression



Simulation



Related Content

- Formula Functions
- Using BRFplus with a Third-Party Rules Engine
- Wikipedia, Business Rules: http://en.wikipedia.org/wiki/Business_rules
- Wikipedia, Business Rule Management System: http://en.wikipedia.org/wiki/Business_Rule_Management_System
- Carsten Ziegler, About Business Rules: https://www.sdn.sap.com/irj/sdn/weblogs?blog=/pub/wlg/9713
- Carsten Ziegler, BRFplus a Business Rule Engine written in ABAP, https://www.sdn.sap.com/irj/sdn/weblogs?blog=/pub/wlg/8889
- Carsten Ziegler, Important Information for Using BRFplus

  https://www.sdn.sap.com/irj/sdn/weblogs?blog=/pub/wlg/11632
- Rajagopalan Narayanan, Business Rules and Software Requirements, https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/40aae118-42c2-2a10-fcaf-fdd9d30bcb1a
- Rajagopalan Narayanan, Seven Tips for Your First Business Rules Project, https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/201a9e3d-3ec2-2a10-85b2-ce56d276dd7a
- Rajagopalan Narayanan, Real World Return of Investment Scenarios with Business Rules Management, https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/b050905e-3cc2-2a10-979a-81a57a787f56
- Rajagopalan Narayanan, Five Reasons to Build Agile Systems Using Business Rules Management Functionality, https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/504486eb-43c2-2a10-f5a7-e84ef3fd45be
- Rajagopalan Narayanan, How Business Rules Management Functionality Helps Tariff Plans Management in Transportation and Shipping, https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/40a9cf69-40c2-2a10-8a8b-969fb311dd31
- Rajagopalan Narayanan, Getting Started with Business Rules Management, https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/70c669d8-3ac2-2a10-0e96-c7c3786168f0

Copyright