

# **Section 4 - Configuration/Rules Overview**

## Table of Contents

<b>Configuration Overview .....</b>	<b>3</b>
<b>Business Rules .....</b>	<b>4</b>
Global Rules and Overrides.....	4
Choosing the Right Override .....	4
Rule Types.....	5
Copy Book.....	5
File .....	6
Function .....	6
Screen .....	6
System .....	7
Table File .....	7
User Defined .....	7
Business Rule Configuration .....	7
Business Rule Functions .....	7
Calculation Functions available with Janino.....	7
<b>Fund .....</b>	<b>50</b>
<b>Transactions.....</b>	<b>51</b>
<b>SegmentName (Segments).....</b>	<b>53</b>
<b>File .....</b>	<b>53</b>
<b>Inquiry .....</b>	<b>53</b>
<b>Batch .....</b>	<b>53</b>

## Configuration Overview

The OIPA system is a rules driven system. The core processing within the system is controlled by a set of rules. These rules are maintained in the form of XML payloads. When a business process is invoked, the appropriate rule is loaded, parsed, and used to provide information for the functionality.

The configuration of the rule is stored in the database. The table it is stored in depends on the Type of Rule.

### Rule Type

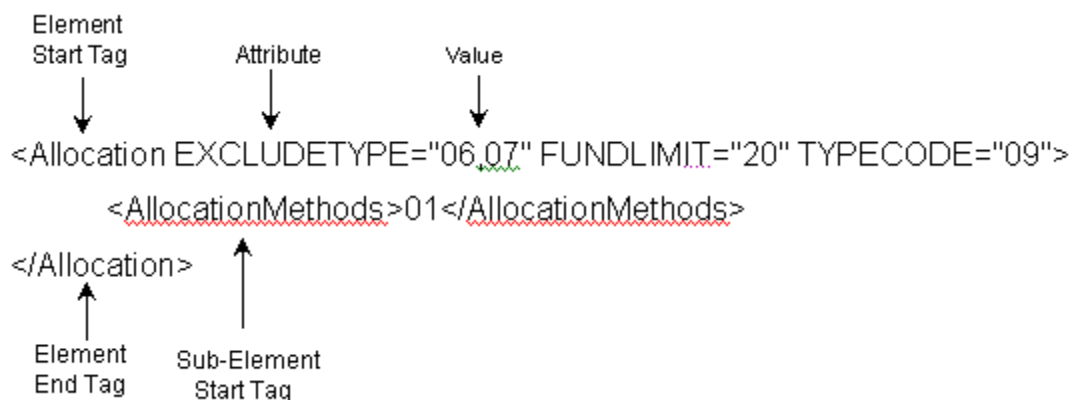
Business Rule  
Transaction  
Segment  
File  
Inquiry  
Batch

### Table Name

AsBusinessRules  
AsTransaction  
AsSegmentName  
AsFile  
AsInquiry  
AsBatch

The Rules are configured with Elements that apply to that rule. Some Elements allow for further functionality through Attributes or Sub-Elements. The Attributes and the Sub-Elements can be configured with one of many values. This configuration determines the rule processing. Some Elements and Attributes are required and some are optional for a Rule.

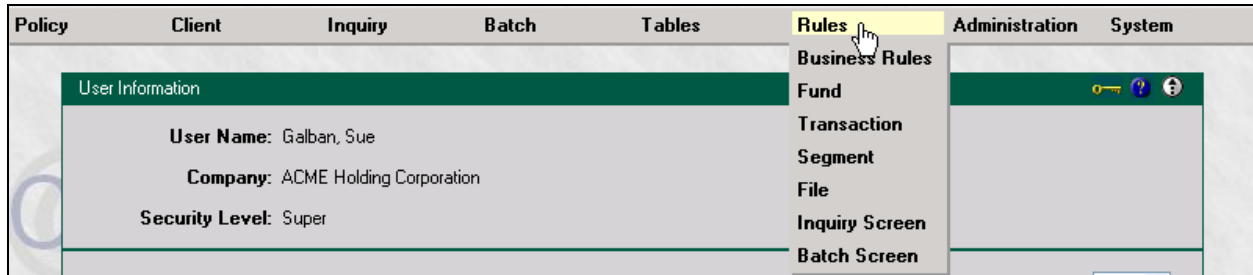
Below is sample configuration from the Allocation Screen Business Rule:



- Each Element and Sub-Element are configured with a Start and End Tag. In this case, `<Allocation></Allocation>` and `<AllocationMethods></AllocationMethods>`.
- `EXCLUDETYPE`, `FUNDLIMIT` AND `TYPECODE` are attributes of `Allocation`. Generally the 'Attribute' is configured in uppercase text and the 'Value', if a string, is configured in camel case. The Value is enclosed in double quotes ( " ). Also, if multiple values are allowed they will be configured in a comma separated list.

- The Value of the AllocationMethods Sub-Element is configured inside of the tag. It is possible for a sub-element to have attributes as well.

All configuration screens are accessed through the Rule Screen from the Main Menu.



## Business Rules

Business Rules are used for configuration for several different stages of business processing. They control the creation, display and validation of screens as well as provide additional processing to transactions.

## Global Rules and Overrides

All the business rules in the system begin their lives as global rules. If a function is common throughout the company, the XML payload can contain the detail needed for the business functionality. However, in most cases functionality changes based on a plan, fund or transaction. Overrides are used in these instances. Additionally, information may change based on an associated state. In these cases, the relevant state code is used, coupled with the plan, fund, or transaction. For example, MinimumIssueAge may differ by plan, but within a plan there still may be a difference based on the state where the policy is issued. Here the override will contain the plan identifier and the state code. This coupling is only allowed when there is a state level override.

### Rule Structure (and example):

Rule Name	Company	PlanGroup	Plan	Fund	Transaction	State	Client	Policy	Segment	XML Payload
RolesExist	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
RolesExist	NULL	NULL	Term	NULL	Premium	NULL	NULL	NULL	NULL	<...20000
RolesExist	NULL	NULL	UL	NULL	Premium	NULL	NULL	NULL	NULL	<...15000

## Choosing the Right Override

Objects providing business functionality use business rules to function properly. When an object is invoked, in most cases it loads the most suitable override of the rule it needs. More precisely, it calls on the business rules object to provide it with the 'best-fit' rule. The business functionality object must provide the business rules object with its exact context in order to get the right business rule. For

example, the business functionality object that processes MinimumIssueAge will have to provide the rule name, plan identifier, and the issue state. Meanwhile, the business rules object will locate all the rules that match the rule name and isolate the 'best-fit'. In other words, it will 'carve-out' a subgroup that matches the plan identifier and then compare the issue state code with the state/plan couple. If an exact match is not found, the rule with just the plan match is returned.

The override priority follows this order:

- Activity
- Segment
- Policy
- Client
- State
- Transaction
- Fund
- Plan
- PlanGroup
- Company

## Rule Types

There are seven rule types. The purpose of rule types is to categorize and define the actions of the Business Rule. To choose rule types, access the Business Rules through the **Rules** menu dropdown list. This will open the **Business Rules** screen. Choose the **Rule Type** from the **Type Code** dropdown list.

Company	Plan	Fund	State
(Global)			>>

## Copy Book

The CopyBook type Business Rule allows for plans, transactions and other rules to share common functionality. Maintenance for actions is made simpler as the rule's XML data is held in one place. This XML data can contain data for the creation of fields for screens, actions for math and spawns or the entire configuration for a transaction.

When a transaction or rule references a CopyBook the contents of that CopyBook are resolved and inserted inside that rule or transaction.

CopyBook Configuration Standards include:

- CopyBook names should begin with 'CopyBook-'
  - CopyBook-*copybookname*
  - Example: CopyBook-AddressChange

## **File**

File rules are used to identify Report Files. The ReportFileGUID is configured in the rule. When the report is generated the system uses the ReportFileGUID configured in the Rule.

## **Function**

The Function type Business Rule is used to create common calculations or functionality that can be broken out into a discrete and logical piece. Functions are often chosen over CopyBooks for specific calculations since they offer increased readability as the Inputs and Outputs are clearly defined. Also, if a Function is used multiple times in a transaction/business rule it will be resolved and compiled once no matter how many times it is used in that rule or transaction.

Function Configuration Standards include:

- Function names should begin with 'Function-'
  - Function-*functionname*
  - Example: Function-ModalFactors
- In the Function itself:
  - Parameters in the Function itself should begin with p for input variables
    - *p"parametername"*
    - pPaymentMethod
  - Output variables should begin with o
    - *o"parametername"*
    - oModalFactor
  - Return variable should begin with r
    - *r"parametername"*
    - rModalPremium

## **Screen**

The Screen Business Rule type allows for configuration of entry fields and validations of most AdminServer entry and search screens. Screen rules allow for manipulation of fixed fields (predetermined fields) as well as the creation of dynamic fields.

## System

System Rule types are Business Rule that must be present in order for the system to process correctly.

## Table File

Table File Business Rules direct the system to a record in the AsUploadRate table. The table file GUID is generated when data is uploaded through the Business Rule Screen.

**Note:** This process is no longer used in J2EE, it has been replaced by the new Rate upload process (Tables/Rates).

## User Defined

User Defined rules are optional rules used to enhance processing. Rules are overridden at the Transaction Level (included in the Transaction Business Rule Packet are examples of User Defined business rules).

## Business Rule Configuration

The configuration for Business Rules varies by Rule . Similar rules use similar configuration. See *Appendix 1 – Business Rule Configuration* for details on possible configuration of the core system rules.

## Business Rule Functions

### Calculation Functions available with Janino

In the Function definitions, use the following table to equate MathVariable data types to the Java equivalent shown in the definitions:

Math DATATYPE	Java Data Type
INTEGER	int
DECIMAL	double
DATE	Date
TEXT	String
OBJECT	Object

Many Functions have multiple definitions for handling different combinations of data types. When Object is used for the data type of a parameter, it means that it will accept a variable of any data type. One must make sure, in any case that the content of a variable make sense in the context of a specific function call.

Function Summaries		
Return DataType	Function Definition	
DECIMAL	<p><b>AbsDiffOf</b>(double sNumberOne, double sNumberTwo)</p> <p>Computes the absolute value of the difference of two numbers</p>	<p><b>AbsDiffOf</b></p> <p>final double <b>AbsDiffOf</b>(double sNumberOne, double sNumberTwo) Computes the absolute value of the difference of two numbers</p> <p><b>Parameters:</b> sNumberOne - sNumberTwo -</p> <p><b>Returns:</b></p>
DECIMAL	<p><b>AbsDiffOf</b>(double sNumberOne, int sNumberTwo)</p> <p>Computes the absolute value of the difference of two numbers</p>	<p><b>AbsDiffOf</b></p> <p>final double <b>AbsDiffOf</b>(double sNumberOne, int sNumberTwo) Computes the absolute value of the difference of two numbers</p> <p><b>Parameters:</b> sNumberOne - sNumberTwo -</p> <p><b>Returns:</b></p>
DECIMAL	<p><b>AbsDiffOf</b>(int sNumberOne, double sNumberTwo)</p> <p>Computes the absolute value of the difference of two numbers</p>	<p><b>AbsDiffOf</b></p> <p>final double <b>AbsDiffOf</b>(int sNumberOne, double sNumberTwo) Computes the absolute value of the difference of two numbers</p> <p><b>Parameters:</b> sNumberOne - sNumberTwo -</p> <p><b>Returns:</b></p>
INTEGER	<p><b>AbsDiffOf</b>(int sNumberOne, int sNumberTwo)</p> <p>Computes the absolute value of the difference of two numbers</p>	<p><b>AbsDiffOf</b></p> <p>final int <b>AbsDiffOf</b>(int sNumberOne, int sNumberTwo) Computes the absolute value of the difference of two numbers</p> <p><b>Parameters:</b> sNumberOne - sNumberTwo -</p> <p><b>Returns:</b></p>



Function Summaries		
Return DataType	Function Definition	
DECIMAL	<p><b>AbsOf</b>(double value)</p> <p>Returns the absolute value of the given value</p>	<p><b>AbsOf</b></p> <p>final double <b>AbsOf</b>(double value) Returns the absolute value of the given value</p> <p><b>Parameters:</b> value -</p> <p><b>Returns:</b></p>
INTEGER	<p><b>AbsOf</b>(int value)</p> <p>Returns the absolute value of the given value</p>	<p><b>AbsOf</b></p> <p>final int <b>AbsOf</b>(int value) Returns the absolute value of the given value</p> <p><b>Parameters:</b> value -</p> <p><b>Returns:</b></p>
TEXT	<p><b>AddZero</b>(int number)</p> <p>Converts in integer to a string and prepends "0", (zero), to it if it has only one digit.</p>	<p><b>AddZero</b></p> <p>final String <b>AddZero</b>(int number) Converts in integer to a string and prepends "0", (zero), to it if it has only one digit. Primarily used to (re)construct numeric codes when the leading zero may have been stripped off.</p> <p><b>Parameters:</b> number - the given integer</p> <p><b>Returns:</b> the new string</p>
TEXT	<p><b>AddZero</b>(String number)</p> <p>Prepends "0", (zero), to the given string if it is numeric and has only one digit.</p>	<p><b>AddZero</b></p> <p>final String <b>AddZero</b>(String number) Prepends "0", (zero), to the given string if it is numeric and has only one digit.</p> <p>Primarily used to (re)construct numeric codes when the leading zero may have been stripped off.</p> <p><b>Parameters:</b> number - the given string</p> <p><b>Returns:</b> the new string</p>

Function Summaries		
Return DataType	Function Definition	
INTEGER	<p><b>AgeOf</b>(Date dateOfBirth, Date dateOfComparison)</p> <p>Computes the age in years given two dates.</p>	<p><b>AgeOf</b></p> <p>final int <b>AgeOf</b>(Date dateOfBirth, Date dateOfComparison) Computes the age in years given two dates.</p> <p>This is a straight calculation based on the exact dates. e.g. if dateOfBirth = March 15, 1972 and dateOfComparison = March 10, 1982, then the age will be 9 since the March 15th anniversary has not been reached in the final year.</p> <p>If dateOfComparison is before dateOfBirth then the result will be a negative number.</p> <p><b>Parameters:</b> dateOfBirth - the starting date dateOfComparison - the ending date</p> <p><b>Returns:</b> the number of years between the two dates</p>
INTEGER	<p><b>AgeOf</b>(Date dob, Object compareDate)</p> <p>Computes the age in years given two dates.</p>	<p><b>AgeOf</b></p> <p>final int <b>AgeOf</b>(Date dob, Object compareDate) Computes the age in years given two dates.</p> <p>This is a straight calculation based on the exact dates. e.g. if dateOfBirth = March 15, 1972 and dateOfComparison = March 10, 1982, then the age will be 9 since the March 15th anniversary has not been reached in the final year.</p> <p>If dateOfComparison is before dateOfBirth then the result will be a negative number.</p> <p><b>Parameters:</b> dateOfBirth - the starting date dateOfComparison - the ending date</p> <p><b>Returns:</b> the number of years between the two dates</p>
INTEGER	<p><b>AgeOf</b>(Object dob, Date compareDate)</p> <p>Computes the age in years given two dates.</p>	<p><b>AgeOf</b></p> <p>final int <b>AgeOf</b>(Object dob, Date compareDate) Computes the age in years given two dates.</p> <p>This is a straight calculation based on the exact</p>

Function Summaries		
Return DataType	Function Definition	
		<p>dates. e.g. if dateOfBirth = March 15, 1972 and dateOfComparison = March 10, 1982, then the age will be 9 since the March 15th anniversary has not been reached in the final year.</p> <p>If dateOfComparison is before dateOfBirth then the result will be a negative number.</p> <p><b>Parameters:</b> dateOfBirth - the starting date dateOfComparison - the ending date</p> <p><b>Returns:</b> the number of years between the two dates</p>
INTEGER	<b>AgeOf</b> (Object dateOfBirth, Object dateOfComparison)	<p><b>AgeOf</b></p> <p>final int <b>AgeOf</b>(Object dateOfBirt Object dateOfComparison)</p>
INTEGER	<p><b>ANBAgeOf</b>(Date birthDate, Date asOfDate)</p> <p>Computes the age in years given two dates.</p>	<p><b>ANBAgeOf</b></p> <p>final int <b>ANBAgeOf</b>(Date birthDate, Date asOfDate) Computes the age in years given two dates.</p> <p>This calculation based on moving the ending date to the nearest anniversary of the starting date. e.g. if dateOfBirth = March 15, 1972 and dateOfComparison = December 20, 1981, then the age will be 10 since the December 20, 1981 is closer to March 15, 1982 than it is to March 15, 1981.</p> <p>If dateOfComparison is before dateOfBirth then the result will be a negative number.</p> <p><b>Parameters:</b> dateOfBirth - the starting date dateOfComparison - the ending date</p> <p><b>Returns:</b> the number of years between the two dates</p>
INTEGER	<p><b>ANBAgeOf</b>(Date birthDate, Object asOfDate)</p> <p>Computes the age in years given two dates.</p>	<p><b>ANBAgeOf</b></p> <p>final int <b>ANBAgeOf</b>(Date birthDate, Object asOfDate)</p>

Function Summaries		
Return DataType	Function Definition	
		<p>Computes the age in years given two dates.</p> <p>This calculation based on moving the ending date to the nearest anniversary of the starting date. e.g. if dateOfBirth = March 15, 1972 and dateOfComparison = December 20, 1981, then the age will be 10 since the December 20, 1981 is closer to March 15, 1982 than it is to March 15, 1981.</p> <p>If dateOfComparison is before dateOfBirth then the result will be a negative number.</p> <p><b>Parameters:</b> dateOfBirth - the starting date dateOfComparison - the ending date</p> <p><b>Returns:</b> the number of years between the two dates</p>
INTEGER	<p><b>ANBAgeOf</b>(Object birthDate, Date asOfDate)</p> <p>Computes the age in years given two dates.</p>	<p><b>ANBAgeOf</b></p> <p>final int <b>ANBAgeOf</b>(Object birthDate, Date asOfDate) Computes the age in years given two dates.</p> <p>This calculation based on moving the ending date to the nearest anniversary of the starting date. e.g. if dateOfBirth = March 15, 1972 and dateOfComparison = December 20, 1981, then the age will be 10 since the December 20, 1981 is closer to March 15, 1982 than it is to March 15, 1981.</p> <p>If dateOfComparison is before dateOfBirth then the result will be a negative number.</p> <p><b>Parameters:</b> dateOfBirth - the starting date dateOfComparison - the ending date</p> <p><b>Returns:</b> the number of years between the two dates</p>
INTEGER	<p><b>ANBAgeOf</b>(Object birthDate, Object asOfDate)</p> <p>Computes the age in years given two dates.</p>	<p><b>ANBAgeOf</b></p> <p>final int <b>ANBAgeOf</b>(Object birthDate, Object asOfDate) Computes the age in years given two dates.</p> <p>This calculation based on moving the ending date to the nearest anniversary of the starting date.</p>

Function Summaries		
Return DataType	Function Definition	
		<p>e.g. if dateOfBirth = March 15, 1972 and dateOfComparison = December 20, 1981, then the age will be 10 since the December 20, 1981 is closer to March 15, 1982 than it is to March 15, 1981.</p> <p>If dateOfComparison is before dateOfBirth then the result will be a negative number.</p> <p><b>Parameters:</b> dateOfBirth - the starting date dateOfComparison - the ending date</p> <p><b>Returns:</b> the number of years between the two dates</p>
DATE	<p><b>AsDateAdd</b>(Date date, int days, int months, int years)</p> <p>Adds the specified number of days, months and years to a date.</p>	<p><b>AsDateAdd</b></p> <p>final Date <b>AsDateAdd</b>(Date date, int days, int months, int years)</p> <p>Adds the specified number of days, months and years to a date.</p> <p><b>Parameters:</b> date - the starting date days - the number of days to add months - the number of months to add years - the number of years to add</p> <p><b>Returns:</b> the new date</p>
DATE	<p><b>AsDateAdd</b>(Object date, int days, int months, int years)</p> <p>Adds the specified number of days, months and years to a date.</p>	<p><b>AsDateAdd</b></p> <p>final Date <b>AsDateAdd</b>(Object date, int days, int months, int years)</p> <p>Adds the specified number of days, months and years to a date.</p> <p><b>Parameters:</b> date - the starting date days - the number of days to add months - the number of months to add years - the number of years to add</p> <p><b>Returns:</b> the new date</p>
INTEGER	<p><b>CalendarMonthsDiffOf</b>(Date startDate, Date endDate)</p> <p>Computes the number of months between two dates</p>	<p><b>CalendarMonthsDiffOf</b></p> <p>final int <b>CalendarMonthsDiffOf</b>(Date startDate, Date endDate)</p> <p>Computes the number of months between two</p>

Function Summaries		
Return DataType	Function Definition	
		<p>dates</p> <p><b>Parameters:</b> startDate - endDate -</p> <p><b>Returns:</b> the number of months</p>
INTEGER	<p><b>CalendarMonthsDiffOf</b>(Date startDate, Object endDate)</p> <p>Computes the number of months between two dates</p>	<p><b>CalendarMonthsDiffOf</b></p> <p>final int <b>CalendarMonthsDiffOf</b>(Date startDate, Object endDate) Computes the number of months between two dates</p> <p><b>Parameters:</b> startDate - endDate -</p> <p><b>Returns:</b> the number of months</p>
INTEGER	<p><b>CalendarMonthsDiffOf</b>(Object startDate, Date endDate)</p> <p>Computes the number of months between two dates</p>	<p><b>CalendarMonthsDiffOf</b></p> <p>final int <b>CalendarMonthsDiffOf</b>(Object startDate, Date endDate) Computes the number of months between two dates</p> <p><b>Parameters:</b> startDate - endDate -</p> <p><b>Returns:</b> the number of months</p>
INTEGER	<p><b>CalendarMonthsDiffOf</b>(Object startDate, Object endDate)</p> <p>Computes the number of months between two dates</p>	<p><b>CalendarMonthsDiffOf</b></p> <p>final int <b>CalendarMonthsDiffOf</b>(Object startDate, Object endDate) Computes the number of months between two dates</p> <p><b>Parameters:</b> startDate - endDate -</p> <p><b>Returns:</b> the number of months</p>
DATE	<p><b>CalendarQuarter</b>(Date date)</p> <p>Returns the end date of the quarter that contains the given date</p>	<p><b>CalendarQuarter</b></p> <p>final Date <b>CalendarQuarter</b>(Date date) Returns the end date of the quarter that contains the given date</p>

Function Summaries		
Return DataType	Function Definition	
		<b>Parameters:</b> date - the starting date <b>Returns:</b> the end date of the quarter containing date
DATE	<b>CalendarQuarter</b> (Object date)  Returns the end date of the quarter that contains the given date	<b>CalendarQuarter</b>  final Date <b>CalendarQuarter</b> (Object date) Returns the end date of the quarter that contains the given date  <b>Parameters:</b> date - the starting date <b>Returns:</b> the end date of the quarter containing date
INTEGER	<b>CalendarYearsDiffOf</b> (Date startDate, Date endDate)  Computes the number of years between two dates.	<b>CalendarYearsDiffOf</b>  final int <b>CalendarYearsDiffOf</b> (Date startDate, Date endDate) Computes the number of years between two dates.  <b>Parameters:</b> startDate - endDate - <b>Returns:</b>
INTEGER	<b>CalendarYearsDiffOf</b> (Date startDate, Object endDate)  Computes the number of years between two dates.	<b>CalendarYearsDiffOf</b>  final int <b>CalendarYearsDiffOf</b> (Date startDate, Object endDate) Computes the number of years between two dates.  <b>Parameters:</b> startDate - endDate - <b>Returns:</b>
INTEGER	<b>CalendarYearsDiffOf</b> (Object startDate, Date endDate)	<b>CalendarYearsDiffOf</b>  final int <b>CalendarYearsDiffOf</b> (Object startDate, Date endDate)
INTEGER	<b>CalendarYearsDiffOf</b> (Object startDate, Object endDate)  Computes the number of years between two dates.	<b>CalendarYearsDiffOf</b>  final int <b>CalendarYearsDiffOf</b> (Object startDate, Object endDate) Computes the number of years between two dates.  <b>Parameters:</b>

Function Summaries		
Return DataType	Function Definition	
		startDate - endDate - <b>Returns:</b>
INTEGER	<b>DayOf</b> (Date date)  Returns the day portion of the given date	<b>DayOf</b>  final int <b>DayOf</b> (Date date) Returns the day portion of the given date  <b>Parameters:</b> date - the date <b>Returns:</b> the day portion of date
INTEGER	<b>DayOf</b> (Object date)  Returns the day portion of the given date	<b>DayOf</b>  final int <b>DayOf</b> (Object date) Returns the day portion of the given date  <b>Parameters:</b> date - the date <b>Returns:</b> the day portion of date
INTEGER	<b>DayOf</b> (String date)  Returns the day portion of the given date	<b>DayOf</b>  final int <b>DayOf</b> (String date) Returns the day portion of the given date  <b>Parameters:</b> date - the date <b>Returns:</b> the day portion of date
DATE	<b>DaysAdd</b> (Date date, double days)  Adds a number of days to a date.	<b>DaysAdd</b>  final Date <b>DaysAdd</b> (Date date, double days) Adds a number of days to a date. If days is negative, the resulting date will be before the given date.  <b>Parameters:</b> date - the given date days - the number of days to add <b>Returns:</b> a new date that is days number of days from date
DATE	<b>DaysAdd</b> (Date date, int days)  Adds a number of days to a date.	<b>DaysAdd</b>  final Date <b>DaysAdd</b> (Date date, int days) Adds a number of days to a date. If days is negative, the resulting date will be before



Function Summaries		
Return DataType	Function Definition	
		<p>the given date.</p> <p><b>Parameters:</b> date - the given date days - the number of days to add</p> <p><b>Returns:</b> a new date that is days number of days from date</p>
DATE	<p><b>DaysAdd</b>(Object date, double days)</p> <p>Adds a number of days to a date.</p>	<p><b>DaysAdd</b></p> <p>final Date <b>DaysAdd</b>(Object date, double days) Adds a number of days to a date. If days is negative, the resulting date will be before the given date.</p> <p><b>Parameters:</b> date - the given date days - the number of days to add</p> <p><b>Returns:</b> a new date that is days number of days from date</p>
DATE	<p><b>DaysAdd</b>(Object date, int days)</p> <p>Adds a number of days to a date.</p>	<p><b>DaysAdd</b></p> <p>final Date <b>DaysAdd</b>(Object date, int days) Adds a number of days to a date. If days is negative, the resulting date will be before the given date.</p> <p><b>Parameters:</b> date - the given date days - the number of days to add</p> <p><b>Returns:</b> a new date that is days number of days from date</p>
TEXT	<p><b>DaysAdd</b>(String sDate, double days)</p> <p>Adds a number of days to a date.</p>	<p><b>DaysAdd</b></p> <p>final String <b>DaysAdd</b>(String sDate, double days) Adds a number of days to a date. If days is negative, the resulting date will be before the given date.</p> <p><b>Parameters:</b> date - the given date days - the number of days to add</p> <p><b>Returns:</b> a new date that is days number of days from date</p>
TEXT	<p><b>DaysAdd</b>(String sDate, int days)</p> <p>Adds a number of days to a date.</p>	<p><b>DaysAdd</b></p> <p>final String <b>DaysAdd</b>(String sDate, int days) Adds a number of days to a date. If days is negative, the resulting date will be before</p>

Function Summaries		
Return DataType	Function Definition	
		<p>the given date.</p> <p><b>Parameters:</b> date - the given date days - the number of days to add</p> <p><b>Returns:</b> a new date that is days number of days from date</p>
INTEGER	<p><b>DaysDiffOf</b>(Date startDate, Date endDate)</p> <p>Computes the number of days between two dates.</p>	<p><b>DaysDiffOf</b></p> <p>final int <b>DaysDiffOf</b>(Date startDate, Date endDate) Computes the number of days between two dates.</p> <p>If endDate is before startDate the result is a negative number.</p> <p><b>Parameters:</b> startDate - start date endDate - end date</p> <p><b>Returns:</b> the number of days difference</p>
INTEGER	<p><b>DaysDiffOf</b>(Object startDate, Object endDate)</p> <p>Computes the number of days between two dates.</p>	<p><b>DaysDiffOf</b></p> <p>final int <b>DaysDiffOf</b>(Object startDate, Object endDate) Computes the number of days between two dates.</p> <p>If endDate is before startDate the result is a negative number.</p> <p><b>Parameters:</b> startDate - start date endDate - end date</p> <p><b>Returns:</b> the number of days difference</p>
INTEGER	<p><b>DurationOf</b>(Date dob, Date compareDate)</p> <p>Alias for AgeOf().</p>	<p><b>DurationOf</b></p> <p>final int <b>DurationOf</b>(Date dob, Date compareDate) Alias for AgeOf().</p> <p><b>Parameters:</b> dob - start date compareDate - end date</p> <p><b>Returns:</b> number of years</p> <p><b>See Also:</b> AgeOf</p>

Function Summaries		
Return DataType	Function Definition	
INTEGER	<p><b>DurationOf</b>(Date dob, String compareDate)</p> <p>Alias for AgeOf().</p>	<p><b>DurationOf</b></p> <p>final int <b>DurationOf</b>(Date dob, String compareDate) Alias for AgeOf().</p> <p><b>Parameters:</b> dob - start date compareDate - end date</p> <p><b>Returns:</b> number of years</p> <p><b>See Also:</b> AgeOf</p>
INTEGER	<p><b>DurationOf</b>(Object dob, Object compareDate)</p> <p>Alias for AgeOf().</p>	<p><b>DurationOf</b></p> <p>final int <b>DurationOf</b>(Object dob, Object compareDate) Alias for AgeOf().</p> <p><b>Parameters:</b> dob - start date compareDate - end date</p> <p><b>Returns:</b> number of years</p> <p><b>See Also:</b> AgeOf</p>
INTEGER	<p><b>DurationOf</b>(String dob, Date compareDate)</p> <p>Alias for AgeOf().</p>	<p><b>DurationOf</b></p> <p>final int <b>DurationOf</b>(String dob, Date compareDate) Alias for AgeOf().</p> <p><b>Parameters:</b> dob - start date compareDate - end date</p> <p><b>Returns:</b> number of years</p> <p><b>See Also:</b> AgeOf</p>
INTEGER	<p><b>DurationOf</b>(String dob, String compareDate)</p> <p>Alias for AgeOf().</p>	<p><b>DurationOf</b></p> <p>final int <b>DurationOf</b>(String dob, String compareDate) Alias for AgeOf().</p> <p><b>Parameters:</b> dob - start date compareDate - end date</p> <p><b>Returns:</b></p>

Function Summaries		
Return DataType	Function Definition	
		number of years <b>See Also:</b> AgeOf
INTEGER	<b>FullMonthsDiffOf</b> (Date startDate, Date endDate)  Computes the number of months between two dates.	<b>FullMonthsDiffOf</b>  final int <b>FullMonthsDiffOf</b> (Date startDate, Date endDate) Computes the number of months between two dates.  Does not include partial months.  <b>Parameters:</b> startDate - endDate - <b>Returns:</b> the number of months
INTEGER	<b>FullMonthsDiffOf</b> (Object startDate, Object endDate)	<b>FullMonthsDiffOf</b>  final int <b>FullMonthsDiffOf</b> (Object startDate, Object endDate)
DATE	<b>GivesBestDay</b> (Date startDate, int bestDay)  Computes a date having the same month and year as the given date and with the specified day of the month. If the specified day causes the new date to roll over into the next month, the day is decreased until that does not occur.	<b>GivesBestDay</b>  final Date <b>GivesBestDay</b> (Date startDate, int bestDay) Computes a date having the same month and year as the given date and with the specified day of the month. If the specified day causes the new date to roll over into the next month, the day is decreased until that does not occur.  For example, GivesBestDate( "2/15/2007", 30 ) returns "2/28/2007".  <b>Parameters:</b> startDate - the given date bestDay - the day of the month <b>Returns:</b> a new date with the same month and year as startDate and the same, (or lesser), day as bestDay
DATE	<b>GivesBestDay</b> (Object startDate, int bestDay)  Computes a date having the same month and year as the given date and with the	<b>GivesBestDay</b>  final Date <b>GivesBestDay</b> (Object startDate, int bestDay)

Function Summaries		
Return DataType	Function Definition	
	<p>specified day of the month. If the specified day causes the new date to roll over into the next month, the day is decreased until that does not occur.</p>	<p>Computes a date having the same month and year as the given date and with the specified day of the month. If the specified day causes the new date to roll over into the next month, the day is decreased until that does not occur.</p> <p>For example, GivesBestDate( "2/15/2007", 30 ) returns "2/28/2007".</p> <p><b>Parameters:</b> startDate - the given date bestDay - the day of the month</p> <p><b>Returns:</b> a new date with the same month and year as startDate and the same, (or lesser), day as bestDay</p>
DATE	<p><b>GivesBestDay</b>(Object startDate, Object bestDay)</p> <p>Computes a date having the same month and year as the given date and with the specified day of the month. If the specified day causes the new date to roll over into the next month, the day is decreased until that does not occur.</p>	<p><b>GivesBestDay</b></p> <p>final Date <b>GivesBestDay</b>(Object startDate, Object bestDay) Computes a date having the same month and year as the given date and with the specified day of the month. If the specified day causes the new date to roll over into the next month, the day is decreased until that does not occur.</p> <p>For example, GivesBestDate( "2/15/2007", 30 ) returns "2/28/2007".</p> <p><b>Parameters:</b> startDate - the given date bestDay - the day of the month</p> <p><b>Returns:</b> a new date with the same month and year as startDate and the same, (or lesser), day as bestDay</p>
boolean	<p><b>hasMethod</b>(String name)</p> <p>Determines whether or not a function of the given name exists in this class.</p>	<p><b>hasMethod</b></p> <p>final boolean <b>hasMethod</b>(String name) Determines whether or not a function of the given name exists in this class.</p> <p><b>Parameters:</b> name - name of the desired function</p> <p><b>Returns:</b> true if the function exists</p>

Function Summaries		
Return DataType	Function Definition	
INTEGER	<p><b>IntegerDivide</b>(double dividend, double divisor)</p> <p>Divides two numbers and returns the integer portion of the result</p>	<p><b>IntegerDivide</b></p> <p>Final int <b>IntegerDivide</b>(double dividend, double divisor) Divides two numbers and returns the integer portion of the result</p> <p><b>Parameters:</b> dividend - numerator divisor - denominator</p> <p><b>Returns:</b> integer value of the division</p>
INTEGER	<p><b>IntegerDivide</b>(double dividend, int divisor)</p> <p>Divides two numbers and returns the integer portion of the result</p>	<p><b>IntegerDivide</b></p> <p>final int <b>IntegerDivide</b>(double dividend, int divisor) Divides two numbers and returns the integer portion of the result</p> <p><b>Parameters:</b> dividend - numerator divisor - denominator</p> <p><b>Returns:</b> integer value of the division</p>
INTEGER	<p><b>IntegerDivide</b>(int dividend, double divisor)</p> <p>Divides two numbers and returns the integer portion of the result</p>	<p><b>IntegerDivide</b></p> <p>final int <b>IntegerDivide</b>(int dividend, double divisor) Divides two numbers and returns the integer portion of the result</p> <p><b>Parameters:</b> dividend - numerator divisor - denominator</p> <p><b>Returns:</b> integer value of the division</p>
INTEGER	<p><b>IntegerDivide</b>(int dividend, int divisor)</p> <p>Divides two integers</p>	<p><b>IntegerDivide</b></p> <p>final int <b>IntegerDivide</b>(int dividend, int divisor) Divides two integers</p> <p><b>Parameters:</b> dividend - numerator divisor - denominator</p> <p><b>Returns:</b> integer value of the division</p>
DECIMAL	<p><b>InterestCalc</b>(double Amount, double InterestRate, Date StartDate, Date EndDate, Date BasisDateForLeapYear)</p>	<p><b>InterestCalc</b></p> <p>final double <b>InterestCalc</b>(double Amount, double InterestRate, Date StartDate, Date EndDate, Date BasisDateForLeapYear)</p>

Function Summaries		
Return DataType	Function Definition	
	Calculate interest for the period between two dates.	<p>Calculate interest for the period between two dates.</p> <p><b>Parameters:</b>  Amount - principle amount  InterestRate - interest rate. If &lt;-1 or &gt;1, assumed to be in percent format  StartDate - beginning date  EndDate - ending date  BasisDateForLeapYearUsed - to determine when an extra day's interest should be included</p> <p><b>Returns:</b>  calculated interest</p>
boolean	<p><b>IsNotNull</b>(double base)</p> <p>Determines if the given parameter is not null.</p>	<p><b>IsNotNull</b></p> <p>boolean <b>IsNotNull</b>(double base)  Determines if the given parameter is not null. Since double can't be null always return true.</p> <p><b>Parameters:</b>  base - Suppress the 'unused' warning since there is no need to read the variable</p> <p><b>Returns:</b></p>
boolean	<p><b>IsNotNull</b>(int base)</p> <p>Determines if the given parameter is not null.</p>	<p><b>IsNotNull</b></p> <p>boolean <b>IsNotNull</b>(int base)  Determines if the given parameter is not null. Since int can't be null always return true.</p> <p><b>Parameters:</b>  base - Suppress the 'unused' warning since there is no need to read the variable</p> <p><b>Returns:</b></p>
boolean	<p><b>IsNotNull</b>(Object base)</p> <p>Determines if the given parameter is not null.</p>	<p><b>IsNotNull</b></p> <p>boolean <b>IsNotNull</b>(Object base)  Determines if the given parameter is not null.</p> <p><b>Parameters:</b>  base -</p> <p><b>Returns:</b></p>
boolean	<p><b>IsNull</b>(double base)</p> <p>Determines if the given parameter is null.</p>	<p><b>IsNull</b></p> <p>boolean <b>IsNull</b>(double base)  Determines if the given parameter is null. Since double can't be null always return false.</p> <p><b>Parameters:</b></p>

Function Summaries		
Return DataType	Function Definition	
		base - Suppress the 'unused' warning since there is no need to read the variable <b>Returns:</b>
boolean	<b>IsNull</b> (int base)  Determines if the given parameter is null.	<b>IsNull</b>  boolean <b>IsNull</b> (int base) Determines if the given parameter is null. Since int can't be null always return false.  <b>Parameters:</b> base - Suppress the 'unused' warning since there is no need to read the variable <b>Returns:</b>
boolean	<b>IsNull</b> (Object base)  Determines if the given parameter is null.	<b>IsNull</b>  boolean <b>IsNull</b> (Object base) Determines if the given parameter is null.  <b>Parameters:</b> base - <b>Returns:</b>
DATE	<b>MaxDateOf</b> (Date date1, Date date2)  Returns the greater of two dates	<b>MaxDateOf</b>  final Date <b>MaxDateOf</b> (Date date1, Date date2) Returns the greater of two dates <b>Parameters:</b> date1 - first date date2 - second date <b>Returns:</b> d1 if d1 is after d2, otherwise d2
DATE	<b>MaxDateOf</b> (Date d1, Object d2)  Returns the greater of two dates	<b>MaxDateOf</b>  final Date <b>MaxDateOf</b> (Date d1, Object d2) Returns the greater of two dates  <b>Parameters:</b> d1 - first date d2 - second date <b>Returns:</b> d1 if d1 is after d2, otherwise d2
DATE	<b>MaxDateOf</b> (Object d1, Date d2)  Returns the greater of two dates	<b>MaxDateOf</b>  final Date <b>MaxDateOf</b> (Object d1, Date d2) Returns the greater of two dates  <b>Parameters:</b> d1 - first date



Function Summaries		
Return DataType	Function Definition	
		d2 - second date <b>Returns:</b> d1 if d1 is after d2, otherwise d2
DATE	<b>MaxDateOf</b> (Object d1, Object d2)  Returns the greater of two dates	<b>MaxDateOf</b>  final Date <b>MaxDateOf</b> (Object d1, Object d2) Returns the greater of two dates  <b>Parameters:</b> d1 - first date d2 - second date <b>Returns:</b> d1 if d1 is after d2, otherwise d2
DECIMAL	<b>MaxOf</b> (double a, double b)  Returns the larger of two numbers.	<b>MaxOf</b>  final double <b>MaxOf</b> (double a, double b) Returns the larger of two numbers. Always returns the second in the case of equality.  <b>Parameters:</b> a - b - <b>Returns:</b>
INTEGER	<b>MaxOf</b> (int a, int b)	<b>MaxOf</b>  final int <b>MaxOf</b> (int a, int b)
DATE	<b>MinDateOf</b> (Date d1, Date d2)  Returns the lesser of two dates	<b>MinDateOf</b>  final Date <b>MinDateOf</b> (Date d1, Date d2) Returns the lesser of two dates  <b>Parameters:</b> d1 - first date d2 - second date <b>Returns:</b> d1 if d2 is after d1, otherwise d2
DATE	<b>MinDateOf</b> (Date d1, Object d2)  Returns the lesser of two dates	<b>MinDateOf</b>  final Date <b>MinDateOf</b> (Date d1, Object d2) Returns the lesser of two dates  <b>Parameters:</b> d1 - first date d2 - second date <b>Returns:</b> d1 if d2 is after d1, otherwise d2

Function Summaries		
Return DataType	Function Definition	
DATE	<p><b>MinDateOf</b>(Object d1, Date d2)</p> <p>Returns the lesser of two dates</p>	<p><b>MinDateOf</b></p> <p>final Date <b>MinDateOf</b>(Object d1, Date d2) Returns the lesser of two dates</p> <p><b>Parameters:</b> d1 - first date d2 - second date</p> <p><b>Returns:</b> d1 if d2 is after d1, otherwise d2</p>
DATE	<p><b>MinDateOf</b>(Object d1, Object d2)</p> <p>Returns the lesser of two dates</p>	<p><b>MinDateOf</b></p> <p>final Date <b>MinDateOf</b>(Object d1, Object d2) Returns the lesser of two dates</p> <p><b>Parameters:</b> d1 - first date d2 - second date</p> <p><b>Returns:</b> d1 if d2 is after d1, otherwise d2</p>
DECIMAL	<p><b>MinOf</b>(double a, double b)</p> <p>Returns the Minimum result of comparing two numbers.</p>	<p><b>MinOf</b></p> <p>final double <b>MinOf</b>(double a, double b) Returns the Minimum result of comparing two numbers. Always returns the second in the case of equality.</p> <p><b>Parameters:</b> a - b -</p> <p><b>Returns:</b></p>
INTEGER	<p><b>MinOf</b>(int a, int b)</p>	<p><b>MinOf</b></p> <p><b>final int MinOf(int a, int b)</b></p>
DATE	<p><b>MonthAdd</b>(Date date)</p> <p>Adds one month to a date.</p>	<p><b>MonthAdd</b></p> <p>final Date <b>MonthAdd</b>(Date date) Adds one month to a date.</p> <p><b>Parameters:</b> date - the given date</p> <p><b>Returns:</b> a new date that is one month from date</p>
DATE	<p><b>MonthAdd</b>(Object date)</p> <p>Adds one month to a date.</p>	<p><b>MonthAdd</b></p> <p>final Date <b>MonthAdd</b>(Object date) Adds one month to a date.</p> <p><b>Parameters:</b> date - the given date</p>

Function Summaries		
Return DataType	Function Definition	
		<b>Returns:</b> a new date that is one month from date
TEXT	<b>MonthAdd</b> (String date)  Adds one month to a date.	<b>MonthAdd</b>  final String <b>MonthAdd</b> (String date) Adds one month to a date.  <b>Parameters:</b> date - the given date <b>Returns:</b> a new date that is one month from date
INTEGER	<b>MonthOf</b> (Date date)  Returns the month portion of the given date	<b>MonthOf</b>  final int <b>MonthOf</b> (Date date) Returns the month portion of the given date  <b>Parameters:</b> date - the date <b>Returns:</b> the month portion of date
INTEGER	<b>MonthOf</b> (Object date)  Returns the month portion of the given date	<b>MonthOf</b>  final int <b>MonthOf</b> (Object date) Returns the month portion of the given date  <b>Parameters:</b> date - the date <b>Returns:</b> the month portion of date
INTEGER	<b>MonthOf</b> (String date)  Returns the month portion of the given date	<b>MonthOf</b>  final int <b>MonthOf</b> (String date) Returns the month portion of the given date  <b>Parameters:</b> date - the date <b>Returns:</b> the month portion of date
DATE	<b>MonthsAdd</b> (Date date, double months)  Adds a number of months to a date.	<b>MonthsAdd</b>  final Date <b>MonthsAdd</b> (Date date, double months) Adds a number of months to a date. If months is negative, the resulting date will be before the given date.  <b>Parameters:</b> date - the given date

Function Summaries		
Return DataType	Function Definition	
		months - the number of months to add <b>Returns:</b> a new date that is months number of months from date
DATE	<b>MonthsAdd</b> (Date date, int months)  Adds a number of months to a date.	<b>MonthsAdd</b>  final Date <b>MonthsAdd</b> (Date date, int months) Adds a number of months to a date. If months is negative, the resulting date will be before the given date.  <b>Parameters:</b> date - the given date months - the number of months to add <b>Returns:</b> a new date that is months number of months from date
DATE	<b>MonthsAdd</b> (Object date, double months)  Adds a number of months to a date.	<b>MonthsAdd</b>  final Date <b>MonthsAdd</b> (Object date, double months) Adds a number of months to a date. If months is negative, the resulting date will be before the given date.  <b>Parameters:</b> date - the given date months - the number of months to add <b>Returns:</b> a new date that is months number of months from date
DATE	<b>MonthsAdd</b> (Object date, int months)  Adds a number of months to a date.	<b>MonthsAdd</b>  final Date <b>MonthsAdd</b> (Object date, int months) Adds a number of months to a date. If months is negative, the resulting date will be before the given date.  <b>Parameters:</b> date - the given date months - the number of months to add <b>Returns:</b> a new date that is months number of months from date
TEXT	<b>MonthsAdd</b> (String date, double months)  Adds a number of months to a date.	<b>MonthsAdd</b>  final String <b>MonthsAdd</b> (String date, double months)

Function Summaries		
Return DataType	Function Definition	
		<p>Adds a number of months to a date. If months is negative, the resulting date will be before the given date.</p> <p><b>Parameters:</b> date - the given date months - the number of months to add</p> <p><b>Returns:</b> a new date that is months number of months from date</p>
TEXT	<p><b>MonthsAdd</b>(String sDate, int months)</p> <p>Adds a number of months to a date.</p>	<p><b>MonthsAdd</b></p> <p>final String <b>MonthsAdd</b>(String sDate, int months) Adds a number of months to a date. If months is negative, the resulting date will be before the given date.</p> <p><b>Parameters:</b> date - the given date months - the number of months to add</p> <p><b>Returns:</b> a new date that is months number of months from date</p>
INTEGER	<p><b>MonthsDiffOf</b>(Date startDate, Date endDate)</p> <p>Computes the number of months between two dates.</p>	<p><b>MonthsDiffOf</b></p> <p>final int <b>MonthsDiffOf</b>(Date startDate, Date endDate) Computes the number of months between two dates.</p> <p>Does not include partial months.</p> <p><b>Parameters:</b> startDate - endDate -</p> <p><b>Returns:</b> the number of months</p>
INTEGER	<p><b>MonthsDiffOf</b>(Object startDate, Object endDate)</p> <p>Computes the number of months between two dates.</p>	<p><b>MonthsDiffOf</b></p> <p>final int <b>MonthsDiffOf</b>(Object startDate, Object endDate) Computes the number of months between two dates.</p> <p>Does not include partial months.</p> <p><b>Parameters:</b> startDate -</p>

Function Summaries		
Return DataType	Function Definition	
		endDate - <b>Returns:</b> the number of months
DATE	<b>NextBusinessDay</b> (Date startDate)  Computes the date of the next business day following the given date	<b>NextBusinessDay</b>  final Date <b>NextBusinessDay</b> (Date startDate) Computes the date of the next business day following the given date  <b>Parameters:</b> date - <b>Returns:</b> the date of the following business day
DATE	<b>NextBusinessDay</b> (Object date)  Computes the date of the next business day following the given date	<b>NextBusinessDay</b>  final Date <b>NextBusinessDay</b> (Object date) Computes the date of the next business day following the given date  <b>Parameters:</b> date - <b>Returns:</b> the date of the following business day
DATE	<b>NextDay</b> (Date date)  Computes the date of the next day following the given date	<b>NextDay</b>  final Date <b>NextDay</b> (Date date) Computes the date of the next day following the given date  <b>Parameters:</b> date - <b>Returns:</b> the date of the following day
DATE	<b>NextDay</b> (Object date)  Computes the date of the next day following the given date	<b>NextDay</b>  final Date <b>NextDay</b> (Object date) Computes the date of the next day following the given date  <b>Parameters:</b> date - <b>Returns:</b> the date of the following day
DATE	<b>NextMode</b> (Date startDate, int mode)  Returns a date that is one unit from the startDate.	<b>NextMode</b>  final Date <b>NextMode</b> (Date startDate, int mode)

Function Summaries																
Return DataType	Function Definition															
	The mode determines the size of the unit:	<div>Returns a date that is one unit from the startDate. The mode determines the size of the unit:</div> <table><tr><th>Mode</th><th>Unit</th></tr><tr><td>1</td><td>Year</td></tr><tr><td>2</td><td>Half Year</td></tr><tr><td>4</td><td>Quarter</td></tr><tr><td>12</td><td>Month</td></tr><tr><td>24</td><td>Bi- Week</td></tr><tr><td>52</td><td>Week</td></tr></table> <div><b>Parameters:</b> startDate - the starting date mode - the mode as indicated in the table above</div> <div><b>Returns:</b> a date that is one unit of time after the starting date</div>	Mode	Unit	1	Year	2	Half Year	4	Quarter	12	Month	24	Bi- Week	52	Week
Mode	Unit															
1	Year															
2	Half Year															
4	Quarter															
12	Month															
24	Bi- Week															
52	Week															
DATE	<div><b>NextMode</b>(Date startDate, String mode)</div> <div>Returns a date that is one unit from the startDate. The mode determines the size of the unit:</div>	<div><b>NextMode</b></div> <div>final Date <b>NextMode</b>(Date startDate, String mode) Returns a date that is one unit from the startDate. The mode determines the size of the unit:</div> <table><tr><th>Mode</th><th>Unit</th></tr><tr><td>1</td><td>Year</td></tr><tr><td>2</td><td>Half Year</td></tr><tr><td>4</td><td>Quarter</td></tr><tr><td>12</td><td>Month</td></tr><tr><td>24</td><td>Bi- Week</td></tr><tr><td>52</td><td>Week</td></tr></table> <div><b>Parameters:</b> startDate - the starting date mode - the mode as indicated in the table above</div> <div><b>Returns:</b> a date that is one unit of time after the starting date</div>	Mode	Unit	1	Year	2	Half Year	4	Quarter	12	Month	24	Bi- Week	52	Week
Mode	Unit															
1	Year															
2	Half Year															
4	Quarter															
12	Month															
24	Bi- Week															
52	Week															
DATE	<div><b>NextMode</b>(Object startDate, int mode)</div> <div>Returns a date that is one unit from the startDate. The mode determines the size of the unit:</div>	<div><b>NextMode</b></div> <div>final Date <b>NextMode</b>(Object startDate, int mode) Returns a date that is one unit from the startDate. The mode determines the size of the unit:</div> <table><tr><th>Mode</th><th>Unit</th></tr></table>	Mode	Unit												
Mode	Unit															

## Function Summaries

Return DataType	Function Definition															
		<table><tr><td>1</td><td>Year</td></tr><tr><td>2</td><td>Half Year</td></tr><tr><td>4</td><td>Quarter</td></tr><tr><td>12</td><td>Month</td></tr><tr><td>24</td><td>Bi- Week</td></tr><tr><td>52</td><td>Week</td></tr></table> <p><b>Parameters:</b> startDate - the starting date mode - the mode as indicated in the table above</p> <p><b>Returns:</b> a date that is one unit of time after the starting date</p>	1	Year	2	Half Year	4	Quarter	12	Month	24	Bi- Week	52	Week		
1	Year															
2	Half Year															
4	Quarter															
12	Month															
24	Bi- Week															
52	Week															
DATE	<p><b>NextMode</b>(Object startDate, String mode)</p> <p>Returns a date that is one unit from the startDate. The mode determines the size of the unit:</p>	<p><b>NextMode</b></p> <p>final Date <b>NextMode</b>(Object startDate, String mode) Returns a date that is one unit from the startDate. The mode determines the size of the unit:</p> <table><tr><th>Mode</th><th>Unit</th></tr><tr><td>1</td><td>Year</td></tr><tr><td>2</td><td>Half Year</td></tr><tr><td>4</td><td>Quarter</td></tr><tr><td>12</td><td>Month</td></tr><tr><td>24</td><td>Bi- Week</td></tr><tr><td>52</td><td>Week</td></tr></table> <p><b>Parameters:</b> startDate - the starting date mode - the mode as indicated in the table above</p> <p><b>Returns:</b> a date that is one unit of time after the starting date</p>	Mode	Unit	1	Year	2	Half Year	4	Quarter	12	Month	24	Bi- Week	52	Week
Mode	Unit															
1	Year															
2	Half Year															
4	Quarter															
12	Month															
24	Bi- Week															
52	Week															
TEXT	<p><b>NextMode</b>(String startDate, int mode)</p> <p>Returns a date that is one unit from the startDate. The mode determines the size of the unit:</p>	<p><b>NextMode</b></p> <p>final String <b>NextMode</b>(String startDate, int mode) Returns a date that is one unit from the startDate. The mode determines the size of the unit:</p> <table><tr><th>Mode</th><th>Unit</th></tr><tr><td>1</td><td>Year</td></tr><tr><td>2</td><td>Half</td></tr></table>	Mode	Unit	1	Year	2	Half								
Mode	Unit															
1	Year															
2	Half															



Function Summaries																
Return DataType	Function Definition															
		<table><tr><td></td><td>Year</td></tr><tr><td>4</td><td>Quarter</td></tr><tr><td>12</td><td>Month</td></tr><tr><td>24</td><td>Bi-Week</td></tr><tr><td>52</td><td>Week</td></tr></table> <p><b>Parameters:</b> startDate - the starting date mode - the mode as indicated in the table above</p> <p><b>Returns:</b> a date that is one unit of time after the starting date</p>		Year	4	Quarter	12	Month	24	Bi-Week	52	Week				
	Year															
4	Quarter															
12	Month															
24	Bi-Week															
52	Week															
TEXT	<p><b>NextMode</b>(String startDate, String mode)</p> <p>Returns a date that is one unit from the startDate. The mode determines the size of the unit:</p>	<p><b>NextMode</b></p> <p>final String <b>NextMode</b>(String startDate, String mode) Returns a date that is one unit from the startDate. The mode determines the size of the unit:</p> <table><tr><th>Mode</th><th>Unit</th></tr><tr><td>1</td><td>Year</td></tr><tr><td>2</td><td>Half Year</td></tr><tr><td>4</td><td>Quarter</td></tr><tr><td>12</td><td>Month</td></tr><tr><td>24</td><td>Bi-Week</td></tr><tr><td>52</td><td>Week</td></tr></table> <p><b>Parameters:</b> startDate - the starting date mode - the mode as indicated in the table above</p> <p><b>Returns:</b> a date that is one unit of time after the starting date</p>	Mode	Unit	1	Year	2	Half Year	4	Quarter	12	Month	24	Bi-Week	52	Week
Mode	Unit															
1	Year															
2	Half Year															
4	Quarter															
12	Month															
24	Bi-Week															
52	Week															
Date	<p><b>NextModeBestDay</b>(Date startDate, int mode, int bestDay)</p> <p>Computes the date of the next mode following the given date, but with the specified day of the month</p>	<p><b>NextModeBestDay</b></p> <p>final Date <b>NextModeBestDay</b>(Date startDate, int mode, int bestDay) Computes the date of the next mode following the given date, but with the specified day of the month</p> <p><b>Parameters:</b> startDate - the start date mode - the mode bestDay - the day of the month</p> <p><b>Returns:</b></p>														

Function Summaries																
Return DataType	Function Definition															
		the computed date <b>See Also:</b> GivesBestDay()														
DATE	<b>NextModeBestDay</b> (Object startDate, int mode, int bestDay)	<b>NextModeBestDay</b>  final Date <b>NextModeBestDay</b> (Object startDate, int mode, int bestDay)														
DATE	<b>NextModeBestDay</b> (Object startDate, Object mode, Object bestDay)  Computes the date of the next mode following the given date, but with the specified day of the month	<b>NextModeBestDay</b>  final Date <b>NextModeBestDay</b> (Object startDate, Object mode, Object bestDay) Computes the date of the next mode following the given date, but with the specified day of the month  <b>Parameters:</b> startDate - the start date mode - the mode bestDay - the day of the month <b>Returns:</b> the computed date <b>See Also:</b> GivesBestDay()														
DATE	<b>NextMultipleMode</b> (Date startDate, int mode, int number)  Returns a date that is a number of units from the startDate. The mode determines the size of the units:	<b>NextMultipleMode</b>  final Date <b>NextMultipleMode</b> (Date startDate, int mode, int number) Returns a date that is a number of units from the startDate. The mode determines the size of the units: <table><tr><th>Mode</th><th>Unit</th></tr><tr><td>1</td><td>Year</td></tr><tr><td>2</td><td>Half Year</td></tr><tr><td>4</td><td>Quarter</td></tr><tr><td>12</td><td>Month</td></tr><tr><td>24</td><td>Bi-Week</td></tr><tr><td>52</td><td>Week</td></tr></table> <b>Parameters:</b> startDate - the starting date mode - the mode as indicated in the table above number - the number of units to add to the date <b>Returns:</b> a date that is the specified number of units from the starting date	Mode	Unit	1	Year	2	Half Year	4	Quarter	12	Month	24	Bi-Week	52	Week
Mode	Unit															
1	Year															
2	Half Year															
4	Quarter															
12	Month															
24	Bi-Week															
52	Week															

Function Summaries																
Return DataType	Function Definition															
DATE	<div><b>NextMultipleMode</b>(Date startDate, String mode, int number)</div> <div>Returns a date that is a number of units from the startDate. The mode determines the size of the units:</div>	<div><b>NextMultipleMode</b></div> <div>final Date <b>NextMultipleMode</b>(Date startDate, String mode, int number) Returns a date that is a number of units from the startDate. The mode determines the size of the units:</div> <table><tr><th>Mode</th><th>Unit</th></tr><tr><td>1</td><td>Year</td></tr><tr><td>2</td><td>Half Year</td></tr><tr><td>4</td><td>Quarter</td></tr><tr><td>12</td><td>Month</td></tr><tr><td>24</td><td>Bi-Week</td></tr><tr><td>52</td><td>Week</td></tr></table> <div><b>Parameters:</b> startDate - the starting date mode - the mode as indicated in the table above number - the number of units to add to the date</div> <div><b>Returns:</b> a date that is the specified number of units from the starting date</div>	Mode	Unit	1	Year	2	Half Year	4	Quarter	12	Month	24	Bi-Week	52	Week
Mode	Unit															
1	Year															
2	Half Year															
4	Quarter															
12	Month															
24	Bi-Week															
52	Week															
DATE	<div><b>NextYear</b>(Date date)</div> <div>Adds one year to the given date</div>	<div><b>NextYear</b></div> <div>final Date <b>NextYear</b>(Date date) Adds one year to the given date</div> <div><b>Parameters:</b> date -</div> <div><b>Returns:</b></div>														
DATE	<div><b>NextYear</b>(Object date)</div> <div>Adds one year to the given date</div>	<div><b>NextYear</b></div> <div>final Date <b>NextYear</b>(Object date) Adds one year to the given date</div> <div><b>Parameters:</b> date -</div> <div><b>Returns:</b></div>														
DATE	<div><b>NextYear</b>(String date)</div> <div>Adds one year to the given date</div>	<div><b>NextYear</b></div> <div>final Date <b>NextYear</b>(String date) Adds one year to the given date</div> <div><b>Parameters:</b></div>														

Function Summaries		
Return DataType	Function Definition	
		date - <b>Returns:</b>
INTEGER	<p><b>PolMonthOf</b>(Date issueDate, Date processDate)</p> <p>Computes the month number of (a policy) at a specified point in time.</p>	<p><b>PolMonthOf</b></p> <p>final int <b>PolMonthOf</b>(Date issueDate, Date processDate) Computes the month number of (a policy) at a specified point in time.</p> <p><b>Parameters:</b> issueDate - the issue date of the policy processDate - the point in time for which to compute the policy month</p> <p><b>Returns:</b> the policy month as of processDate</p>
INTEGER	<p><b>PolMonthOf</b>(Date issueDate, String processDate)</p> <p>Computes the month number of (a policy) at a specified point in time.</p>	<p><b>PolMonthOf</b></p> <p>final int <b>PolMonthOf</b>(Date issueDate, String processDate) Computes the month number of (a policy) at a specified point in time.</p> <p><b>Parameters:</b> issueDate - the issue date of the policy processDate - the point in time for which to compute the policy month</p> <p><b>Returns:</b> the policy month as of processDate</p>
INTEGER	<p><b>PolMonthOf</b>(Object issueDate, Object processDate)</p> <p>Computes the month number of (a policy) at a specified point in time.</p>	<p><b>PolMonthOf</b></p> <p>final int <b>PolMonthOf</b>(Object issueDate, Object processDate) Computes the month number of (a policy) at a specified point in time.</p> <p><b>Parameters:</b> issueDate - the issue date of the policy processDate - the point in time for which to compute the policy month</p> <p><b>Returns:</b> the policy month as of processDate</p>
INTEGER	<p><b>PolMonthOf</b>(String issueDate, Date processDate)</p> <p>Computes the month number of (a policy) at a specified point in time.</p>	<p><b>PolMonthOf</b></p> <p>final int <b>PolMonthOf</b>(String issueDate, Date processDate) Computes the month number of (a policy) at a specified point in time.</p>

Function Summaries		
Return DataType	Function Definition	
		<b>Parameters:</b> issueDate - the issue date of the policy processDate - the point in time for which to compute the policy month <b>Returns:</b> the policy month as of processDate
INTEGER	<b>PolMonthOf</b> (String issueDate, String processDate)  Computes the month number of (a policy) at a specified point in time.	<b>PolMonthOf</b>  final int <b>PolMonthOf</b> (String issueDate, String processDate) Computes the month number of (a policy) at a specified point in time.  <b>Parameters:</b> issueDate - the issue date of the policy processDate - the point in time for which to compute the policy month <b>Returns:</b> the policy month as of processDate
DECIMAL	<b>PresentValue</b> (double rate, int numPayments, double payment, double futureValue, double type)  Calculates the present value of a number.	<b>PresentValue</b>  final double <b>PresentValue</b> (double rate, int numPayments, double payment, double futureValue, double type) Calculates the present value of a number.  <b>Parameters:</b> rate - interest rate numPayments - number of payments payment - the payment amount futureValue - future value of the number type - mode <b>Returns:</b> the present value
DECIMAL	<b>PresentValue</b> (Object rate, Object numPayments, Object payment, Object futureValue, Object type)  Calculates the present value of a number.	<b>PresentValue</b>  final double <b>PresentValue</b> (Object rate, Object numPayments, Object payment, Object futureValue, Object type) Calculates the present value of a number.  <b>Parameters:</b> rate - interest rate numPayments - number of payments payment - the payment amount futureValue - future value of the number type - mode <b>Returns:</b> the present value

Function Summaries		
Return DataType	Function Definition	
DATE	<p><b>QuarterLastBusinessDay</b>(Date startDate)</p> <p>Determines the last business day of the quarter containing the given date.</p>	<p><b>QuarterLastBusinessDay</b></p> <p>final Date  <b>QuarterLastBusinessDay</b>(Date startDate)  Determines the last business day of the quarter containing the given date.  Does not take into account holidays.</p> <p><b>Parameters:</b>  startDate - the date  <b>Returns:</b>  the last business day of the quarter.</p>
DATE	<p><b>QuarterLastBusinessDay</b>(Object startDate)</p>	<p><b>QuarterLastBusinessDay</b></p> <p>final Date  <b>QuarterLastBusinessDay</b>(Object startDate)</p>
TEXT	<p><b>QuarterLastBusinessDay</b>(String startDate)</p> <p>Determines the last business day of the quarter containing the given date.</p>	<p><b>QuarterLastBusinessDay</b></p> <p>final String  <b>QuarterLastBusinessDay</b>(String startDate)  Determines the last business day of the quarter containing the given date.  Does not take into account holidays.</p> <p><b>Parameters:</b>  startDate - the date  <b>Returns:</b>  the last business day of the quarter</p>
DATE	<p><b>QuartersAdd</b>(Date date, double quarters)</p> <p>Adds a number of quarters (3 months) to a date</p>	<p><b>QuartersAdd</b></p> <p>final Date <b>QuartersAdd</b>(Date date, double quarters)  Adds a number of quarters (3 months) to a date</p> <p><b>Parameters:</b>  sDate - the original date  quarters - the number or quarters to add  <b>Returns:</b>  the new date</p>
DATE	<p><b>QuartersAdd</b>(Date date, int quarters)</p> <p>Adds a number of quarters (3 months) to a date</p>	<p><b>QuartersAdd</b></p> <p>final Date <b>QuartersAdd</b>(Date date, int quarters)  Adds a number of quarters (3 months) to a date</p> <p><b>Parameters:</b>  sDate - the original date  quarters - the number or quarters to add</p>

Function Summaries		
Return DataType	Function Definition	
		<b>Returns:</b> the new date
DATE	<p><b><u>QuartersAdd</u></b>(Object date, double quarters)</p> <p>Adds a number of quarters (3 months) to a date</p>	<p><b>QuartersAdd</b></p> <p>final Date <b>QuartersAdd</b>(Object date, double quarters)  Adds a number of quarters (3 months) to a date</p> <p><b>Parameters:</b>  sDate - the original date  quarters - the number or quarters to add</p> <p><b>Returns:</b>  the new date</p>
DATE	<p><b>QuartersAdd</b>(Object date, int quarters)</p> <p>Adds a number of quarters (3 months) to a date</p>	<p><b>QuartersAdd</b></p> <p>final Date <b>QuartersAdd</b>(Object date, int quarters)  Adds a number of quarters (3 months) to a date</p> <p><b>Parameters:</b>  sDate - the original date  quarters - the number or quarters to add</p> <p><b>Returns:</b>  the new date</p>
TEXT	<p><b>QuartersAdd</b>(String date, double quarters)</p> <p>Adds a number of quarters (3 months) to a date</p>	<p><b>QuartersAdd</b></p> <p>final String <b>QuartersAdd</b>(String date, double quarters)  Adds a number of quarters (3 months) to a date</p> <p><b>Parameters:</b>  sDate - the original date  quarters - the number or quarters to add</p> <p><b>Returns:</b>  the new date</p>
TEXT	<p><b>QuartersAdd</b>(String sDate, int quarters)</p> <p>Adds a number of quarters (3 months) to a date</p>	<p><b>QuartersAdd</b></p> <p>final String <b>QuartersAdd</b>(String sDate, int quarters)  Adds a number of quarters (3 months) to a date</p> <p><b>Parameters:</b>  sDate - the original date  quarters - the number or quarters to add</p> <p><b>Returns:</b>  the new date</p>
TEXT	<p><b>RemoveLeadingZeros</b>(String originalString)</p> <p>Remove all zeros from the beginning of the</p>	<p><b>RemoveLeadingZeros</b></p>

Function Summaries		
Return DataType	Function Definition	
	given string value	final String <b>RemoveLeadingZeros</b> (String originalString)  Remove all zeros from the beginning of the given string value <b>Parameters:</b> originalString - the original string <b>Returns:</b> the string with leading zeros removed
DATE	<b>ReplaceDayOfMonth</b> (Date date, int day)  Replaces the day portion of a date	<b>ReplaceDayOfMonth</b>  final Date <b>ReplaceDayOfMonth</b> (Date date, int day) Replaces the day portion of a date  <b>Parameters:</b> date - the original date day - the new dayvalue <b>Returns:</b> the new date
DATE	<b>ReplaceDayOfMonth</b> (Object date, int day)  Replaces the day portion of a date	<b>ReplaceDayOfMonth</b>  final Date <b>ReplaceDayOfMonth</b> (Object date, int day) Replaces the day portion of a date  <b>Parameters:</b> date - the original date day - the new dayvalue <b>Returns:</b> the new date
TEXT	<b>ReplaceDayOfMonth</b> (String date, int day)  Replaces the day portion of a date	<b>ReplaceDayOfMonth</b>  final String <b>ReplaceDayOfMonth</b> (String date, int day) Replaces the day portion of a date  <b>Parameters:</b> date - the original date day - the new dayvalue <b>Returns:</b> the new date
DATE	<b>ReplaceMonth</b> (Date date, int month)  Replaces the month portion of a date	<b>ReplaceMonth</b>  final Date <b>ReplaceMonth</b> (Date date, int month) Replaces the month portion of a date  <b>Parameters:</b>



Function Summaries		
Return DataType	Function Definition	
		date - the original date month - the new month value <b>Returns:</b> the new date
DATE	<b>ReplaceMonth</b> (Object date, int month)  Replaces the month portion of a date	<b>ReplaceMonth</b>  final Date <b>ReplaceMonth</b> (Object date, int month) Replaces the month portion of a date  <b>Parameters:</b> date - the original date month - the new month value <b>Returns:</b> the new date
DATE	<b>ReplaceMonth</b> (Object date, Object month)  Replaces the month portion of a date	<b>ReplaceMonth</b>  final Date <b>ReplaceMonth</b> (Object date, Object month) Replaces the month portion of a date  <b>Parameters:</b> date - the original date month - the new month value <b>Returns:</b> the new date
TEXT	<b>ReplaceMonth</b> (String date, int month)  Replaces the month portion of a date	<b>ReplaceMonth</b>  final String <b>ReplaceMonth</b> (String date, int month) Replaces the month portion of a date  <b>Parameters:</b> date - the original date month - the new month value <b>Returns:</b> the new date
DATE	<b>ReplaceYear</b> (Date date, int year)  Replaces the year portion of a date	<b>ReplaceYear</b>  final Date <b>ReplaceYear</b> (Date date, int year) Replaces the year portion of a date  <b>Parameters:</b> date - the original date year - the new year value <b>Returns:</b> the new date
DATE	<b>ReplaceYear</b> (Object date, int year)  Replaces the year portion of a date	<b>ReplaceYear</b>

Function Summaries		
Return DataType	Function Definition	
		<p>final Date <b>ReplaceYear</b>(Object date, int year) Replaces the year portion of a date</p> <p><b>Parameters:</b> date - the original date year - the new year value</p> <p><b>Returns:</b> the new date</p>
DATE	<p><b>ReplaceYear</b>(Object date, Object year)</p> <p>Replaces the year portion of a date</p>	<p><b>ReplaceYear</b></p> <p>final Date <b>ReplaceYear</b>(Object date, Object year) Replaces the year portion of a date</p> <p><b>Parameters:</b> date - the original date year - the new year value</p> <p><b>Returns:</b> the new date</p>
TEXT	<p><b>ReplaceYear</b>(String date, int year)</p> <p>Replaces the year portion of a date</p>	<p><b>ReplaceYear</b></p> <p>final String <b>ReplaceYear</b>(String date, int year) Replaces the year portion of a date</p> <p><b>Parameters:</b> date - the original date year - the new year value</p> <p><b>Returns:</b> the new date</p>
DECIMAL	<p><b>TableMult</b>(String tableRate)</p> <p>Determines the rate multiplier for the given table rating code.</p>	<p><b>TableMult</b></p> <p>final double <b>TableMult</b>(String tableRate) Determines the rate multiplier for the given table rating code.</p> <p><b>Parameters:</b> tableRate - the table rating code</p> <p><b>Returns:</b> the rate multiplier</p>
DATE	<p><b>ToDate</b>(Date date)</p> <p>Converts the given value to a Date object</p>	<p><b>ToDate</b></p> <p>final Date <b>ToDate</b>(Date date) Converts the given value to a Date object</p> <p><b>Parameters:</b> value - the value to a date convert</p> <p><b>Returns:</b> <b>Throws:</b></p>

Function Summaries		
Return DataType	Function Definition	
		<u>EngineExceptionUtil</u> - if the value cannot be converted
DATE	<b>ToDate</b> (Object objDate) Converts the given value to a Date object	<b>ToDate</b> final Date <b>ToDate</b> (Object objDate) Converts the given value to a Date object <b>Parameters:</b> value - the value to convert <b>Returns:</b> a date <b>Throws:</b> <u>EngineExceptionUtil</u> - if the value cannot be converted
DATE	<b>ToDate</b> (String sDate) Converts the given value to a Date object	<b>ToDate</b> final Date <b>ToDate</b> (String sDate) Converts the given value to a Date object <b>Parameters:</b> value - the value to convert <b>Returns:</b> a date <b>Throws:</b> <u>EngineExceptionUtil</u> - if the value cannot be converted
DECIMAL	<b>ToDecimal</b> (double value) Converts the given value to a double value	<b>ToDecimal</b> final double <b>ToDecimal</b> (double value) Converts the given value to a double value <b>Parameters:</b> value - the value to convert <b>Returns:</b> a double <b>Throws:</b> <u>EngineExceptionUtil</u> - if the value cannot be converted
DECIMAL	<b>ToDecimal</b> (Double value) Converts the given value to a double value	<b>ToDecimal</b> final double <b>ToDecimal</b> (Double value) Converts the given value to a double value <b>Parameters:</b> value - the value to convert <b>Returns:</b> a double <b>Throws:</b>

Function Summaries		
Return DataType	Function Definition	
		<u>EngineExceptionUtil</u> - if the value cannot be converted
DECIMAL	<b>ToDecimal</b> (int value) Converts the given value to a double value	<b>ToDecimal</b> final double <b>ToDecimal</b> (int value) Converts the given value to a double value <b>Parameters:</b> value - the value to convert <b>Returns:</b> a double <b>Throws:</b> <u>EngineExceptionUtil</u> - if the value cannot be converted
DECIMAL	<b>ToDecimal</b> (Integer value) Converts the given value to a double value	<b>ToDecimal</b> final double <b>ToDecimal</b> (Integer value) Converts the given value to a double value <b>Parameters:</b> value - the value to convert <b>Returns:</b> a double <b>Throws:</b> <u>EngineExceptionUtil</u> - if the value cannot be converted
DECIMAL	<b>ToDecimal</b> (Object value) Converts the given value to a double value	<b>ToDecimal</b> final double <b>ToDecimal</b> (Object value) Converts the given value to a double value <b>Parameters:</b> value - the value to convert <b>Returns:</b> a double <b>Throws:</b> <u>EngineExceptionUtil</u> - if the value cannot be converted
DECIMAL	<b>ToDecimal</b> (String value) Converts the given value to a double value	<b>ToDecimal</b> final double <b>ToDecimal</b> (String value) Converts the given value to a double value <b>Parameters:</b> value - the value to convert <b>Returns:</b> a double <b>Throws:</b>

Function Summaries		
Return DataType	Function Definition	
		<u>EngineExceptionUtil</u> - if the value cannot be converted
INTEGER	<b>ToInt(double d)</b>  Returns the integer portion of the given double value.	<b>ToInt</b>  final int <b>ToInt</b> (double d) Returns the integer portion of the given double value.  <b>Parameters:</b> d - double <b>Returns:</b> the integer portion of the given double value
INTEGER	<b>ToInt(int i)</b>  Returns the given integer	<b>ToInt</b>  final int <b>ToInt</b> (int i) Returns the given integer  <b>Parameters:</b> i - int <b>Returns:</b> the value given
INTEGER	<b>ToInt(Object value)</b>  Converts the given object to an integer	<b>ToInt</b>  final int <b>ToInt</b> (Object value) Converts the given object to an integer  <b>Parameters:</b> value - the object <b>Returns:</b> the integer portion of the given object  <b>Parameters:</b> dateOfBirth - the starting date dateOfComparison - the ending date <b>Returns:</b> the number of years between the two dates
INTEGER	<b>ToInt(String string)</b>  Works pretty much like the JavaScript parseInt() function with the exception of only dealing with base 10 numbers. Looks at the beginning of the string and finds anything that looks like a number.	<b>ToInt(String string)</b>  Works pretty much like the JavaScript parseInt() function with the exception of only dealing with base 10 numbers. Looks at the beginning of the string and finds anything that looks like a number. Any non-digit, non-decimal point characters following the initial digit sequence are ignored. Fractional portions of the number are likewise ignored. Defaults to 0, (zero), if no number is found.

Function Summaries		
Return DataType	Function Definition	
		<b>Parameters:</b> string: - String - contains the string to be parsed. <b>Returns:</b> int: the integer parsed from the beginning of the given string, or 0 if none is found.
TEXT	<b>ToString</b> (Date date)  Converts the given Date to a String	<b>ToString</b>  final String <b>ToString</b> (Date date) Converts the given Date to a String  <b>Parameters:</b> date - the Date object <b>Returns:</b> the Date formatted as a string
TEXT	<b>ToString</b> (double doubleValue)  Converts the given decimal value to a String	<b>ToString</b>  final String <b>ToString</b> (double doubleValue) Converts the given decimal value to a String  <b>Parameters:</b> doubleValue - the decimal value <b>Returns:</b> the decimal formatted as a string
TEXT	<b>ToString</b> (int integer)  Converts the given integer to a String	<b>ToString</b>  final String <b>ToString</b> (int integer) Converts the given integer to a String  <b>Parameters:</b> integer - the integer <b>Returns:</b> the integer formatted as a string
TEXT	<b>ToString</b> (String string)  Returns the string	<b>ToString</b>  final String <b>ToString</b> (String string) Returns the string <b>Parameters:</b> string - <b>Returns:</b>
DECIMAL	<b>TotalAnnualCOI</b> (double annualCOI, double increasePercent, double yearsLeft)  Calculates the total annual cost of insurance	<b>TotalAnnualCOI</b>  final double <b>TotalAnnualCOI</b> (double annualCOI, double increasePercent, double yearsLeft) Calculates the total annual cost of insurance  <b>Parameters:</b> annualCOI - the cost of insurance for year 1

Function Summaries		
Return DataType	Function Definition	
		<p>increasePercent - the percent the cost increases each year  yearsLeft - the number of years  <b>Returns:</b>  the total cost of insurance for the given number of years</p>
DECIMAL	<p><b>TotalAnnualCOI</b>(double annualCOI, double increasePercent, int yearsLeft)</p> <p>Calculates the total annual cost of insurance</p>	<p><b>TotalAnnualCOI</b></p> <p>public final double  <b>TotalAnnualCOI</b>(double annualCOI, double increasePercent, int yearsLeft)  Calculates the total annual cost of insurance</p> <p><b>Parameters:</b>  annualCOI - the cost of insurance for year 1  increasePercent - the percent the cost increases each year  yearsLeft - the number of years  <b>Returns:</b>  the total cost of insurance for the given number of years</p>
DATE	<p><b>YearBeginOf</b>(Date date)</p> <p>Determines the first day of the year of the given date.</p>	<p><b>YearBeginOf</b></p> <p>final Date <b>YearBeginOf</b>(Date date)  Determines the first day of the year of the given date.</p> <p><b>Parameters:</b>  date - the initial date  <b>Returns:</b>  the first day of the year</p>
DATE	<p><b>YearBeginOf</b>(Object date)</p> <p>Determines the first day of the year of the given date.</p>	<p><b>YearBeginOf</b></p> <p>final Date <b>YearBeginOf</b>(Object date)  Determines the first day of the year of the given date.</p> <p><b>Parameters:</b>  date - the initial date  <b>Returns:</b>  the first day of the year</p>
TEXT	<p><b>YearBeginOf</b>(String date)</p> <p>Determines the first day of the year of the given date.</p>	<p><b>YearBeginOf</b></p> <p>final String <b>YearBeginOf</b>(String date)  Determines the first day of the year of the given date.</p>

Function Summaries		
Return DataType	Function Definition	
		<b>Parameters:</b> date - the initial date <b>Returns:</b> the first day of the year
DATE	<b>YearEndOf</b> (Date date)  Determines the last day of the year of the given date.	<b>YearEndOf</b>  final Date <b>YearEndOf</b> (Date date) Determines the last day of the year of the given date.  <b>Parameters:</b> date - the initial date <b>Returns:</b> the last day of the year
DATE	<b>YearEndOf</b> (Object date)  Determines the last day of the year of the given date.	<b>YearEndOf</b>  final Date <b>YearEndOf</b> (Object date) Determines the last day of the year of the given date.  <b>Parameters:</b> date - the initial date <b>Returns:</b> the last day of the year
TEXT	<b>YearEndOf</b> (String date)  Determines the last day of the year of the given date.	<b>YearEndOf</b>  final String <b>YearEndOf</b> (String date) Determines the last day of the year of the given date.  <b>Parameters:</b> date - the initial date <b>Returns:</b> the last day of the year
INTEGER	<b>YearOf</b> (Date date)  Returns the year portion of the given date	<b>YearOf</b>  final int <b>YearOf</b> (Date date) Returns the year portion of the given date  <b>Parameters:</b> date - the date <b>Returns:</b> the year portion of date
INTEGER	<b>YearOf</b> (Object date)  Returns the year portion of the given date	<b>YearOf</b>  final int <b>YearOf</b> (Object date) Returns the year portion of the given date



Function Summaries		
Return DataType	Function Definition	
		<b>Parameters:</b> date - the date <b>Returns:</b> the year portion of date
INTEGER	<b>YearOf</b> (String date)  Returns the year portion of the given date	<b>YearOf</b>  final int <b>YearOf</b> (String date) Returns the year portion of the given date  <b>Parameters:</b> date - the date <b>Returns:</b> the year portion of date
DATE	<b>YearsAdd</b> (Date date, int years)  Adds a number of years to a date.	<b>YearsAdd</b>  final Date <b>YearsAdd</b> (Date date, int years) Adds a number of years to a date. If years is negative, the resulting date will be before the given date.  <b>Parameters:</b> date - the given date years - the number of years to add <b>Returns:</b> a new date that is years number of years from date
DATE	<b>YearsAdd</b> (Object date, int years)  Adds a number of years to a date.	<b>YearsAdd</b>  final Date <b>YearsAdd</b> (Object date, int years) Adds a number of years to a date. If years is negative, the resulting date will be before the given date.  <b>Parameters:</b> date - the given date years - the number of years to add <b>Returns:</b> a new date that is years number of years from date
TEXT	<b>YearsAdd</b> (String sDate, int years)  Adds a number of years to a date.	<b>YearsAdd</b>  final String <b>YearsAdd</b> (String sDate, int years) Adds a number of years to a date. If years is negative, the resulting date will be before the given date.  <b>Parameters:</b> date - the given date years - the number of years to add

Function Summaries		
Return DataType	Function Definition	
		<b>Returns:</b> a new date that is years number of years from date
INTEGER	<b>YearsDiffOf</b> (Date startDate, Date endDate)  Computes the number of years between two dates.	<b>YearsDiffOf</b>  final int <b>YearsDiffOf</b> (Date startDate, Date endDate) Computes the number of years between two dates.  <b>Parameters:</b> startDate - endDate - <b>Returns:</b>
INTEGER	<b>YearsDiffOf</b> (Object startDate, Object endDate)  Computes the number of years between two dates.	<b>YearsDiffOf</b>  final int <b>YearsDiffOf</b> (Object startDate, Object endDate) Computes the number of years between two dates.  <b>Parameters:</b> startDate - endDate - <b>Returns:</b>

## Fund

Although Fund is listed under Rules no Fund information is configured under this option. This menu option loads the Fund Entry Screen. The actual configuration for this screen is configured in the FundScreen Business Rule. See the *Business Rule Configuration* Section for details on the FundScreen configuration.

Plans	
Company:	Acme Life
Plan Group:	( All Plan Groups )
Plan:	( All Plans )
EffectiveDate:	
Fund(s)	
Fund:	(New Fund)
Page 1 of 1	Page 1
Maximum Results:	5
Effective Date	Status Code
Name:	Status: Active
Type: DCA+ Fund	Effective Date:
Removal Precedence:	Removal Method:
ACME Fund Code:	Oracle Fund Number:
<div>Save</div> <div>New</div> <div>Close</div>	

## Transactions

An advanced feature of the system is the way in which transaction behavior is handled. Administration systems have always maintained separate programs and often-separate files for financial events. For example, an administration system may have one program for Premiums and one program for Withdrawals. The OIPA system uses XML scripting to determine the behavior of a given transaction. All of the business logic exists as XML scripts in the database.

See *Appendix 2 - Transaction Configuration* for detailed information regarding Transaction configuration.

Transactions are the Blueprint for activities. Transactions are configured to meet business needs including entry, accounting, validations, and valuation. Like Business Rules, configuration consists of Elements, Sub-Elements, Attributes and Values. Unlike Business Rules, Transactions cannot be overridden at different levels. If a plan requires a Transaction it must be created for that plan. However, if a transaction works similarly for multiple plans each transaction can be configured to point to the same CopyBook.

Individual Business Rules can also be attached to Transactions for additional processing functionality. These rules are created/overridden at the appropriate Transaction level and 'attached' to the transaction via the TransactionBusinessRulePacket Business Rule.

A Premium Transaction XML example is shown below. This rule exists as a transaction record with the name Premium. Business rules define the cosmetics for this transaction, such as icons and colors, and the XML describes its behavior. Comments in red describe the Elements (Sub-Elements, Attributes and Values)

<Transaction>
<i>Starting element. See Transaction XML Examples for more information.</i>
<EffectiveDate>Enabled</EffectiveDate> <i>Describes the Effective Date Field.</i>
<FundAllocation>Yes</FundAllocation> <i>Describes Fund Allocation Entry.</i>
<DefaultAllocation>Yes</DefaultAllocation> <i>Controls the default values</i>
<Valuation> <i>Runs Valuation prior to transaction processing</i> <EffectiveDateNUVMustExist>Yes</EffectiveDateNUVMustExist> <SystemDateNUVMustExist>No</SystemDateNUVMustExist> </Valuation>
<Fields> <i>Entry Fields including Name, Display, and Type</i> <Field> <Name>GrossAmount</Name> <Display>Gross Amount</Display> <DataType>Money</DataType> </Field> </Fields>
<Spawns> <i>One transaction 'spawns' none, one or many other transactions. It passes the 'spawned' transaction it's entry fields since they won't be keyed</i> <i>Premium spawns another activity – a PremiumBonus.</i> <Spawn> <Transaction SPAWNCODE="01">PremiumBonus</Transaction> <SpawnFields> <SpawnField> <From>PremiumBonusAmount</From> <To>BonusAmount</To> <DataType>Money</DataType> </SpawnField> </SpawnFields> <Allocation>Parent</Allocation> <i>PremiumBonus gets its allocation details from Premium.</i> </Spawn> <i>Premium also spawns a confirm.</i> <Spawn> <Transaction SPAWNCODE="01">ActivityConfirm</Transaction> </Spawn> </Spawns>
<Math> - ** See FIELD Attributed description below <i>Math variables used by the activity.</i> <MathVariables> <MathVariable VARIABLENAME="FrontLoad" TYPE="RULE">FrontLoad</MathVariable> <MathVariable VARIABLENAME="PremiumBonus" TYPE="RULE">BonusPercent</MathVariable> <MathVariable VARIABLENAME="FrontLoadAmount" TYPE="VALUE">0</MathVariable> </MathVariables>
<Assignment TYPE="Apply"> <GrossAmount>01</GrossAmount> <PremiumTaxAmount>02</PremiumTaxAmount> <FrontLoadAmount>03</FrontLoadAmount> </Assignment> <i>Describes how values should be assigned to the policy and written to AsValuation. (See</i>

<i>Assignment Elements section for additional information)</i>
</Math>
</Transaction> <i>Ending element.</i>

## SegmentName (Segments)

SegmentName dictates how the Segment(s) are configured for each Plan. It is configured similar to Screen rules but allows for additional validations and can invoke 'Calculate' Business Rules.

See *Appendix 3 - SegmentName Configuration* for detailed information regarding SegmentName configuration.

## File

This Rule type is configured to translate incoming data and insert it directly into the OIPA database. It is also used to selectively insert records, data and activities based on the state of a particular request.

The configuration for each File has two sections: XMLData and XSLT.

The XMLData processes the incoming request by assigning values to variables that will be used for processing, when transformed by the XSLT.

See *Appendix 4 - File Configuration* for detailed information regarding File configuration.

## Inquiry

Inquiry Rules can be configured to enable you to access information regarding Clients, Policies, Plan, etc. This Rule can be configured to query the database, use math and run valuation. The Rules can be configured to allow inputs that will dictate the returned results as well as how the information is displayed.

See *Appendix 5 - Inquiry Configuration* for detailed information regarding Inquiry configuration.

## Batch

The Batch rule is configured to identify the name of the Batch, the inputs necessary to create the batch of policies, the Transaction to be processed and the activity fields that can be overridden.

See *Appendix 6 - Batch Activity Configuration* for detailed information regarding Batch rule configuration.