ORACLE®
INSURANCE

Oracle® Insurance Policy Administration

Documentation Updated for OIPA Release 9.6.1.0

Version 9.6.1.0

Documentation Part Number: E35883_01

June, 2013

ORACLE®

Table of Contents

UPDATES TO THE RULES PALETTE HELP

This section contains pages from the Rules Palette Help that were updated for the 9.6.1.0 release.

General Structure and Best Practices

Sequence of Transaction Elements

The general sequence of the transaction elements should be as follows (follow the links for more information on best practices for configuring these elements):

1. EffectiveDate
2. Allocations (if applicable)
3. Valuation (if applicable)
4. Suspense
5. ValuesBlock
6. Withholding
7. AllowComments
8. Transitions
9. MultiSuspense
10. Address
11. Fields
12. Events
13. MultiFields
14. Math
15. Assignments (if applicable)
16. Spawns

Syntax and Configuration Standards for Transaction Elements

EffectiveDate

The `EffectiveDate` element should be the first tag in a transaction.
The `EffectiveDate` of spawned transactions, when that spawned activity is only ever SYSTEM generated, should be Disabled.

Allocations (if applicable)

Allocation order of elements should be:

- FundAllocation
- AllocationFrom
- Allocation
- DefaultAllocation

Suspense

Use the OVERRIDABLE attribute to allow users to bypass suspense requirements.

Withholding

When withholding is needed, use the `<Withholding> </Withholding>` tags instead of creating fields for withholding.

Valuation

Do not include the <Valuation> tag in a transaction, if valuation is not necessary.

Fields

1. The order of field tags is:
    - <Name>
    - <Display>
    - <DataType>
    - <Group>
    - <Currency>
    - <Value>
    - <DefaultCurrency>
    - <DefaultValue>
    - <Query>
    - <Calculated>
    - <Required>
    - <Disabled>
    - <Hidden>
    - <Expanded>
    - <ClearOnRecycle>
2. The <DefaultValue> tag and the <Calculated> tag may not be present at the same time. Use one or the other.
3. If a field is hidden, do not include a <Disabled> tag.
4. If a field is disabled, do not include a <Hidden> tag.
5. If a field is required, do not include <Hidden> and <Disabled> tags.
6. If a field is either hidden or disabled, do not include a <Required> tag.
7. If a field is not hidden, do not include <Hidden> tag.
8. If a field is not disabled, do not include <Disabled> tag.
9. Hidden fields that are independent of other fields should be placed at the end of the <Fields> section.
   Note: An exception would be when the value of a hidden field is needed as a source of data in a non-hidden field.
10. The Required element applies only to the following screens:
    - Activities
    - Address
    - Client
    - ClientGroup
    - ClientWithholding
    - Inquiry
    - Policy
    - PolicyWithholding
    - Role
    - Segments
    - Suspense
11. The Required element is not valid if the DataType is Identifier or Check.
12. Validation for required fields is performed when a screen is saved (or submitted). If a required field is blank or null it fails validation, and any validation specified by the Validation element is not performed.
13. If a user is prevented by the security configuration from entering data in a required field, the screen will not pass validation when submitted.

14. Suggestions for field names are listed below. This applies to MathVariable names as well.
15. Do not use spaces or underscores.
    Example: DateOfBirth not Date_of_Birth.
16. Do not abbreviate unless the resulting word is extremely long. In that case, abbreviate but retain the meaning of word.
    Examples: PolicyAnniversary not PolAnniv. NextMonRemainIssueChg for NextMonthaversaryRemainingIssueCharge.
17. Add abbreviated words to the Data Dictionary so that the meaning remains clear.
18. Field name and display value should be the same if possible.
    Example: `<Name>IssueDate</Name>, <Display>Issue Date</Display>` **not** `<Display>Date of Issue</Display>`
19. Filler fields should have a DataType value of Blank, and no Display element.
    Combo boxes and radio buttons:
    - Capitalize the first letter of each word.

    Example: DateOfDeath not DateofDeath.
    - Exception is the use of upper case in SQL reserved words.
    - Exception is the use of upper case for Validation error messages that appear to the user.
    - In a combo box, start with 00 for the value followed by 01, 02, etc.
    - Always use a two-digit format: 00, 01, 02, 03, etc.
    - The 00 value should always stand for a not applicable value or for a **Select an Option** value in a combo box.
    - It may not always make sense to start off a combo box with a 00 value. Please use your judgment or consult your lead B.A.
    - A combo box with a defaulted selection should not have a **00** selection.
    - Radio buttons should have a 00 value for the negative response and 01 for the affirmative response. This can apply to a combo box as well.
21. Fields of DataType TextArea are available for all screens supporting the <Fields> element. Note the following:
    - Fields of the TextArea data type are not available as criteria fields in search screens. TextArea fields configured on a search screen will be ignored.
    - The TextArea data type is not supported within the <Table> element regardless of the rule in which it is configured.
    - The field entry section of an InquiryScreen does not support the TextArea data type. However, the TextArea data type is supported in the results section if it displays fields but not a data grid.
    - There is no absolute maximum to the length of a TextArea field; the number of characters entered may be limited by specifying the <Length> element.
    - There is no minimum number of Unicode characters for the TextArea datatype.
22. Avoid having the same value for the <Name> tag for more than one <Field>. If multiple fields have the same name, the display, recording of the field value and action/events on the field will not be able to differentiate between the multiple instances.

Math

1. General Math best practices:
    - Within the `<Math>` section, any MathVariable constants should come first.
    - Do notdefine MathVariables that are notused elsewhere by the system configuration or the OIPA user.
    - A MathVariable should define its VARIABLENAME and TYPE attributes (in that order) prior to any other attribute.

    Example: `<MathVariable VARIABLENAME=”Example” TYPE=”SQL” DEFAULT =”01” DATATYPE=”TEXT”>`
    - Consolidate logic into a single MathVariable where appropriate.

Example: Where an intermediate calculation value is not needed, don't break the calculation into many steps.

- Only use ROUND when there is a reason.
- Use DATATYPE="DECIMAL" when dealing with integers larger than 2,147,483,648 in math configuration. In OIPA, the maximum value of an integer is 2^{31} or 2,147,483,648. If a user tries to use an integer value larger than the maximum, it will cause an error.
- When using logging (with the LOG="Yes" attribute), remember that the system performs all logging at the end of transaction processing. Therefore, only the ending value of a MathVariable is logged. Although the configuration allows the LOG attribute on any use of a variable in the configuration, the intermediate values are not logged, even if LOG="Yes" occurs multiple times for the same variable. In such a case, to avoid confusion it is best to specify the LOG attribute only for the last use of the variable. Another method is to create a "Log Section" at the bottom of the configuration that specifies all math variables to be logged."

2. When defining MathVariable names, the following practices are recommended:
    - Use only alphanumeric characters — letters, digits, and underscore ('_'). Do not use spaces.
    - The first character of a MathVariable name should be a letter. The underscore is allowed, but not recommended.
    - By convention, the dollar sign is not used and should be avoided.
    - After the first character, any combination of letters, digits, and underscores is allowed.
    - The maximum length of a MathVariable name is database-dependent. A safe length is 25 characters, but maximum length of up to 36 characters may be allowed in certain installations.
    - Use descriptive variable names, with full words if possible instead of abbreviations that may not be meaningful to others.
    - MathVariable names are case sensitive, and using CamelCase-type names helps to make the names more readable and understandable. Underscores may also be used to separate words.

    Example: `<MathVariable VARIABLENAME="ActualEndOfMonthDateMV" ...`
3. When defining the TYPE for a MathVariable use:
    - `VALUE` if setting a MathVariable to a constant, whether it is a numeric or non-numeric constant.

    When defining the constant of Checked or UnChecked, the values should always be either `CHECKED` or `UNCHECKED` in all capital letters, respectively.

    Example: `<MathVariable TYPE="VALUE">CHECKED</MathVariable>`
    - `FIELD` if defining the MathVariable to a field from the `<Fields>` section.

    Note: You must prefix the field name with "Activity:" **Example**: `<MathVariable TYPE="FIELD">Activity:Field1</MathVariable>`. Use Valuation:Policy to retrieve values from activity valuation when possible. **Example**: When trying to retrieve CashValue from PolicyValues use TYPE="FIELD" Valuation:Policy:CashValue.
    - `POLICYFIELD` if setting a MathVariable to a value in the policy's <`Fields`> section.
    - `PLANFIELD` if setting a MathVariable to a value in the plan's <`Fields`> section.

    Use PlanFields any time a particular fact, amount, threshold value, etc., is tied to a product (plan), is static, and is used in a constant fashion across multiple transactions and/or calculations. It should be defined in PlanFields rather than defined repeatedly across transaction or business rule XML. If the information should change, then only one change must be made (to the PlanScreen business rule).

    Use the PlanScreen business rule to define your plan fields.
    - `EXPRESSION` is used in the following situations:

    Use EXPRESSION when performing arithmetic using pre-defined MathVariables, literals or activity fields or if assigning a value from another MathVariable.

    Example: `<MathVariable TYPE="EXPRESSION">Activity:Field1 + Activity:Field2</MathVariable>`

Note: When doing arithmetic with activity fields you must use the prefix `Activity:`.
Use `EXPRESSION` when assigning a new MathVariable to have the value of an existing MathVariable or Activity field, or if assigning value from another MathVariable.

- `FUNCTION` if using one of the system's pre-defined operations. These features aim to provide users with functionality that is frequently used without the need for manually re-configuring the logic.

Do not use double quotes around the parameters. A list of available functions is provided in the Functions topic.

- `MATHIF` if you want to incorporate if/else logic into your math configuration.

When evaluating an integer in a MATHIF, neither single nor double quotes are needed for the IF attribute.
Example: `<MathIF IF="PolicyDuration=1"`

- `IIF` as another way to incorporate if/else logic into your math configuration. This TYPE is different from MATHIF in that it always assigns one of two values to a single math variable, depending on whether the logical expression evaluates to true or false.

Suggestions for when to use MathIF versus TYPE="IIF":

- Use MathIF when one or more MathVariables need to be assigned values only if the logical condition is met.. That is; if the condition is not met, the MathVariables do not need to be defined.
- MathIF also provides a method for setting the values of multiple math variables based on the evaluation of a single expression. In other words, the expression only needs to be evaluated once.

  Example of MATHIF statement:

  ```
  <MathIF IF="A=B">
      <MathVariable VARIABLENAME="MV1"
       TYPE="EXPRESSION" DATATYPE="INTEGER">1+1</MathVariable>
      <MathVariable VARIABLENAME="MV2"
       TYPE="EXPRESSION" DATATYPE="INTEGER">2+2</MathVariable>
  </MathIF>
  <MathIF IF="A<>B">
      <MathVariable VARIABLENAME="MV3"
       TYPE="VALUE" DATATYPE="INTEGER">0</MathVariable>
      <MathVariable VARIABLENAME="MV4"
       TYPE="VALUE" DATATYPE="INTEGER">10</MathVariable>
  </MathIF>
  ```

  Pseudo-code:

  ```
  If A = B
      MV1 = 1 + 1
      MV2 = 2 + 2
  Else
      MV3 = 0
      MV4 = 10
  End
  ```

- Use IIF when only one math variable needs to be set to either of two values based on the evaluation of the expression.

  Example of IIF statement:

  ```
  <MathVariable VARIABLENAME="MV" TYPE="IIF"
   EXPRESSION="FlatCheck=CHECKED"
  DATATYPE="INTEGER">01,02</MathVariable>
  ```

Pseudo-code:

```
If FlatCheck = CHECKED
   MV = 01
Else
   MV = 02
End
```

- `SQL` if the value you are trying to obtain is only obtainable by executing a SQL statement.
- `COLLECTIONVALUE` if you are trying to retrieve a value from a COLLECTION.
- `INTEGERARRAY, NUMERICARRAY, STRINGARRAY, DATEARRAY` if you need a list of values. An array should contain the same data types. Only one-dimensional arrays can be created.

Note: Be certain that you define the array with the appropriate DATATYPE. Do not configure a NUMERICARRAY to store text values or dates.

Some examples of using arrays for list values include (but are not limited to):

  - Rate information
  - Fund information
  - Dates
  - Segment information

- There are various ways to create/populate/manipulate arrays, according to the array type. The OPERATIONS include:
  - `FILLBY-LIST`

    The values inserted into a FILLBY-LIST array can be values contained in math variables and activity fields and/or they can be hard-coded.

    When using FILLBY-LIST to insert date or string values into an array, the values must be surrounded by double quotes.

  - `FILLBY-SQL`

    Used when you want to fill an array with the results from a SQL statement.

    The type of array you are creating and the type of data you are retrieving from your SQL statement must match.

    The SQL statement must only return one column from a table.

    FILLBY-SQL arrays are commonly used to retrieve point-in-time values from logged math.

  - `FILLBY-FUND`

    Used when you want to fill an array with a list of FundGUIDs for a given policy that has cash value.

    The plan (product) must have funds configured in order to use this array/operation type.

    Once you have your FundGUID, you are able to retrieve additional information about the fund such as AsValuation data, AsFundField data, etc.

  - `FILLBY-DEPOSIT`

    Used when you want to fill an array with deposit ValuationGUIDs related to a specific FundGUID.

    When using FILLBY-DEPOSIT to create an array, your transaction/rule must be configured to perform valuation and assignment.

Commonly used in conjunction with FILLBY-FUND however all that is really needed is a FundGUID (which can sometimes be obtained by other means)

Some examples of usage include:

- Obtaining Cash Value per deposit.
- Obtaining number of units purchased per deposit.
- Obtaining gain/loss per deposit.

- `CREATE`

  Used to initialize a MathVariable as an array to later be populated with data.

  Be certain that you define the array with the appropriate DATATYPE.

  Most common example is defining an array prior to performing a loop and then inserting data into the array during each iteration of a loop.

- `REPLACE`

  Used when you want to replace all elements from an existing array for all the elements in an entirely different and separate array.

  Commonly used with the CREATE method where CREATE is an empty array and then is replaced with an array with elements via REPLACE method.

- `COPY`

  Used when you want to produce an exact replica of an existing array.

  The new copy must not have the same MathVariable name as the original.

- `TRANSFORM`

  Used when you want to perform calculations on an array. **Note**: The same rules that apply to TYPE= ”EXPRESSION” apply to TRANSFORM as well.

  When performing calculations using two or more arrays, the arrays must be the same length.

  You may also use literals.

  There is a feature that can be used with NUMERICARRAY, STRINGARRAY and DATEARRAY via METHOD= ”r;” called EXPAND

  - Use this method when you are trying to repeat the first value and/or all the middle values and/or the last value of an array.
  - You must have at least three elements in an array to use the EXPAND method.
  - Some examples of usage include:
    - Expand an array that does not contain enough elements to perform a calculation with another array.

      Example: Monthly rate stored as a single value (in PLANFIELD) but you need to calculate the value for all 12 months. You would use EXPAND to take the monthly rate and expand it 12 times.

    - Use for varying interest rates into Commutation functions.

4. When using `TYPE=”FUNCTION”`, do not use double quotes around the parameters.
5. The use of `Blank` is not needed in a conditional expression where blank is a previously defined field. Instead use two single quotes (i.e., '').
6. Use Data Dictionary defined field names for consistency across products.
7. Use Valuation:Policy to retrieve values from valuation when possible.

When trying to retrieve CashValue from PolicyValues use `TYPE=”FIELD”` Valuation:Policy:CashValue.

8. Capitalize the words `AND` and `OR` when writing SQL. Use sentence case `And` and `Or` when configuring anything other than SQL statements.
9. When you need to obtain the XML data for the current activity, use the following configuration to help performance:

```
<MathVariable VARIABLENAME="ThisPendingActivityXML" TYPE="SQL">
 SELECT XMLData FROM AsActivity WHERE ActivityGUID =
'[Activity:ActivityGUID]'
 </MathVariable>
```

   - Always remove the data contained in a math variable as described above before leaving Math as demonstrated by the following configuration.

```
<MathVariable VARIABLENAME="ThisPendingActivityXML"
TYPE="VALUE"></MathVariable>
```

   This configuration will avoid writing too much data to the database. In the extreme circumstance that you do need to load a large amount of XML into a variable, this will clear out the variable so it does not write as much data to AsActivity XMLData
10. If a data value is already defined as a field, don’t use a SQL to retrieve the value again from the database. Since this value is already a <Field>, change to TYPE=”FIELD” and reference the data by the name of the field.
11. If a piece of information is already defined as a field, do not use a SQL to retrieve this information again from the database. Since this value is already a <Field>, change to TYPE=”FIELD” and reference the field’s name.

Spawns

1. Pass spawn fields in the same sequence as they are configured in the transaction that is being spawned.
2. Any GUID fields passed as spawn fields should be disabled in the spawned transaction.
3. Spawned transactions should use `SPAWNCODE=”01”` if the spawned transaction is effective on the same day as the spawning transaction.
4. See AsCode for other spawn code values.
5. The DataTypes in spawn fields must match the corresponding data types in the receiving fields.

Overview of Screen Rules

Screen rules are used to control the display of screens in OIPA. Screen rules exist as global rules but can be overridden to meet the needs of individual plans. Global screen rules are located in the **Global Explorer** tab in the **Business Rules** folder. Any overrides (except company level overrides) created will be stored in the **Main Explorer** tab under the **Company | Plan | Business Rules | Screen** node.

Most screen rules are configured using the XML Source pane. Visual editing is not available for these rules at this time. An explanation of each rule is provided below. There are two exceptions: RoleScreen and CompanyScreen. These two rules can be configured using visual editing tools.

Please see the **XML Configuration Guide** topic in this help system for a complete list of all elements, attributes and values needed for Screen rule configuration. View **Business Rules | Screen Rules**.

Screen Rules

ActivityResultsScreen: This rule defines the configuration of the Activity Result screen in OIPA. The screen displays when the Activity Detail icon to the left of a processed activity is clicked. Configuration allows the user to control whether parent and/or child funds display and whether the original allocation amount displays. The rule should be overridden at the transaction level if either one of these display enhancements is required. If one or both of these display enhancements is needed, the rule configuration should be updated to reflect the display required and should be attached to a specific transaction. A transaction override of the rule is not included in the TransactionBusinessRulePacket.

The Verification Screen provides a different view, completely configurable, with a section for allocation. The VerificationScreen makes the optional SHOWORIGINAL attribute available.

AddressScreen: This business rule allows the user to configure the non-fixed fields and validations for the address roles on the Address screen. Fixed field values can also be controlled through configuration of this rule. Mailing addresses can be set to expire based on date criteria and a contingent value established to introduce an active or inactive status. Configuration supports foreign addresses and dates. The AddressType is controlled by the AsCode table's AsCodeAddressType column

An address change letter can be automatically generated when a change is made to an existing address. The Spawn IF logic will determine when and if a letter is generated. Configuration has the option to spawn a client level activity and create messages as needed through events/actions. To make use of this functionality the following configuration requirements must be met.

- The activity must be a client level activity.
- Spawning is only available from the Address screen.
- The only spawn code supported is 03.

Refer to the Address Change Letter Prototype for configuration steps to accomplish this task.

The <Agreement> element has several important attributes.

- The TYPECODE attribute defines the agreement type from the AsCodeAgreementType table.
- The ROLETYPE attribute defines the roles that can be assigned to the agreement from the AsCodeAgreementRoleType table.
- The PARENT attribute identifies the agreements that are children of another agreement.

    - PARENT attribute with a value indicates that the agreement is a child of the agreement types listed. **Ex:** PARENT="01,02". The **Add Child** right-click option will not be available on the agreement.
    - Absence of a PARENT attribute indicates that the agreement is a parent. The **Add Child** right-click option will be available on the agreement.
- The ROOT attribute identifies agreements that can appear in the Agreement Type drop down box when a new agreement is added to a Group Customer.
    - Absence of a ROOT attribute identifies that the agreement type can appear in the Agreement Type drop down box when a new agreement is added.
    - ROOT="Yes" indicates that the agreement type can appear in the Agreement Type drop down box when a new agreement is added to a Group Customer.
    - ROOT="No" indicates that the agreement type cannot appear in the Agreement Type drop down box when a new agreement is added to a Group Customer.

AllocationScreen: This business rule defines the allocations that are assigned for the plan and policy. This rule does not have visual editing support and must be configured directly in the XML Source pane. It is used when configuring allocations using the default method.

CaseScreen: This business rule allows a user to create and edit case records, part of the New Business Underwriting process. In OIPA, the Case screen is accessed by selecting **Case** from the Main Menu and clicking on **New Case**, or by selecting **Search Case** from the Main Menu, then searching for and selecting an existing case.

Information displayed on this screen includes the case name, case number, case status, creation date and date last updated, all of which correspond to columns within the AsCase database table. Additionally, dynamic fields can be configured to display other information on the screen. A case's status is represented by a two digit role code, and these role codes are defined in the AsCodeCaseStatus code name.

The screen has three sections:

- The Case General Info section displays the screen's fixed fields, such as case name and number.
- The Policy/Application Table section displays the table defined in the <Table> tags within the <Policies> tags of the CaseScreen rule.
- The Case Detail section displays the dynamic, configurable fields defined in the CaseScreen rule's configuration. These fields' values are stored in the AsCaseField database table.

Masking and field-level security is supported on the Case screen.

CaseSearchScreen: This business rule is used to configure the CaseSearchScreen, which allows an underwriter or CSR to search for cases or applications as part of the New Business Underwriting process. The CaseSearchScreen functions similarly to the Policy Search screen. The CaseSearchScreen's configuration defines the fields that are used to store the results of a case or application search. In OIPA, the Case Search screen is accessed by selecting **Case** from the Main Menu and clicking on **Search Case**.

The screen has two sections:

- The Case Search Criteria section contains the fields into which case search criteria can be entered.
- The Case Search Results section displays the results of a case search.

ChartOfAccountsScreen: This rule determines the set-up of the dynamic fields for the ChartOfAccounts screen (only for the criteria section) and determines how validation can be performed.

ChildFundScreen: This rule has been deprecated as of the 9.5.0.0 release. Refer to the Child Fund page for information on setting up child funds.

This screen rule has an upper section that allows for the display of Fixed Fields for class type, class name, description, and number of members. In addition, there are configurable dynamic fields to further define the class. The screen layout contains a subsection that can be expanded and collapsed by clicking on the section header for Class Detail and the expanded subsection tab will be highlighted in red. Tabs are available for Class Details, Class Plans, Class Rules and Class Members.

ClientScreen: This business rule allows the user to configure fixed and dynamic fields on the Client screen. A separate section is configured for each Client Type, which is identified by its typecode. This rule is also used to control the display of policy roles, individual fields, address table, the TaxID field on Client screen, and the process button for future activities on the Client Activity screen.

Configuration supports the display of foreign calendars and dates as well as numerous formats of the client name. The LegalResidenceCode field and the TypeCode are the only required fields and they determine the display of all information on the screen.

If this screen is called as a result of the use of a client field in an activity, then it will display in a popup window. Security for fields and buttons displayed in the popup window tabs will be determined by the security established on the current Client screen.

The Spawn element can be used within Actions and Events to trigger the spawning of activities when specific fields are updated on the Client screen.

> There are some **configuration considerations** to keep in mind when working with the Client screen. In version 9 of OIPA, the ClientGUID is not an inherent GUID available to the fields section of the Client screen. This means that when a new client is created, a ClientGUID does not yet exist until after the Save button is clicked. If there are fields that require a ClientGUID, they can cause the screen to crash (stack trace) since the ClientGUID cannot be found. To avoid this situation, move any SQL in the field section that needs to resolve the ClientGUID into the Action/Event section for OnChange or OnSubmit events. The will insure proper screen results for a new client. This is not an issue for saved clients in the database since the ClientGUID already exists.

ClientSearchScreen: This business rule defines the configuration for search criteria and fields for the Client Search screen. The screen allows the user to search on various client types such as Individual, Corporate or Producer. When the user enters this screen, the default search criterion is Individual.

External clients can be searched from this screen in OIPA if the supporting configuration is added to this rule. The attribute EXTERNAL=”Yes” can be added to the <Client> section to indicate that the client type is external. The element, <ExternalClientSearchRetriever>, can also be added to the <Client> section of the ClientSearchScreen. This element defines how the client specific information is to be populated in the Client Search results when the user hits the **Find** button. This element contains a class name that implements an interface. The data retrieved is defined in the ExternalClientDetailScreen business rule

If this screen is called as a result of the use of a client field in an activity, then it will display in a popup window. Security for fields and buttons displayed in the popup window tabs will be determined by the security established on the current Client Search screen.

This rule will also allow use of policy and segment context variables providing POLICYGUID and SEGMENTGUID values for any search parameters. The PolicyGUID and SegmentGUID context variables will return appropriate values when there is a Policy and Segment context. In case there is no Policy context , the values will be returned as "Null" without any system error.

ClientSearchScreen BusinessRule works in three modes:

- Client Search under Client Context
- Find Client in RoleScreen (Policy and Segment Roles)
- Find Client in any Activities

In the RoleScreen FindClient context, the ClientSearchScreen will be able to access the POLICYGUID (both Policy and Segment RoleScreen)and SEGMENTGUID (only in Segment RoleScreen) fields using which other policy and screen field values can be calculated for search parameters. In other contexts, these two values will return “Null” in the ClientSearchScreen BusinessRule.

CommentsScreen: This business rule is used to configure OIPA's various Comment screens, which can be implemented for policies, segments, clients, activities and suspense records. Comments on these screens can use preset comment templates, which are configured with the **Comment Templates** node in the Admin Explorer's Administration folder, or can be completely user-entered. Comment templates can be implemented at the global, company, plan group or plan level.

CommentsSearchScreen: This business rule is used to define the search criteria for comments and to configure the display of comment search results. It can be configured at the global, policy, segment, activity, and client levels, as well as for suspense records. In addition to containing fixed fields, the Comments Search screen can use dynamic fields to filter out specific types or categories of comments.

CompanyScreen: The CompanyScreen business rule must be configured before accessing the Company Data node in the Main Explorer. This business rule defines the fields that hold constant values related to a company. It should only be overridden at the company level. The XML Source pane can be used to configure the screen using XML.   A field defined as <DataType>**Money**</DataType> in the CompanyScreen business rule will display a currency field for entry in the Rules Palette CompanyData node.

DisbursementApprovalScreen: This business rule allows for the configuration of dynamic fields on this screen. These fields will be used to search for specific disbursements. The table section defines how the results are displayed to the user. The Disbursement Amount column can be totaled using optional configuration. Currencies must be of the same type. Mixed currencies will not total.

DisbursementScreen: This business rule contains the fields that display when the Disbursement link is selected in the Activity Results window.

DisbursementSearchScreen: This business rule is used to configure the dynamic fields in the DisbursementSearchScreen to allow the user to search for disbursement records that match the specified criteria. If the DisbursementSearchScreen business rule is not used the fixed fields will be displayed by default and used for searches. For example Company, Plan, Start Date and End Date will be displayed.

The DisbursementScreen and DisbursementSearchScreen business rules together constitute the Disbursement screen. Configuration for the Disbursement Search section is done in the DisbursementSearchScreen business rule; whereas the configuration for the Disbursement details section is done in the DisbursementScreen business rule.

ExternalClientDetailScreen: This business rule holds the associated configurable fields for an external client. The fields defined in this business rule will be accessible through SQL. Specific external client information stored in the external database will not be accessible in the OIPA database.

Client details for a role may be configured in this rule. Role fields are configurable fields on the Role Screen business rule based on the role code. The Role Code for the External Client will be labeled as ‘External’. The field data is stored in OIPA’s AsRole and AsRoleField tables.

FundScreen: The FundScreen business rule can be configured to provide additional information for fund records. This rule defines whether there will be child funds and/or benefit funds. Parent and child funds are used when the same fund may be offered but there are different classes of the fund (versions, bands, groups, etc.). Extra fields can be stored at the parent and child level of the fund. Additionally this is where funds applicable to benefit split are determined as well.

This rule is overridden at the Primary Company Level and copybooks are not supported in this rule.

This rule must be configured in the XML Source pane.

If child funds or benefit funds are needed, the following attributes must be present in the <ChildFunds> element.

- ALLOWED: this is a required attribute that will accept a literal **Yes** or **No** with the default value being No. This attribute will indicate if child funds should be created from parent funds.
- BENEFITFUNDS: this is an optional attribute that will accept a literal **Yes** or **No** with the default value being No. This attribute will indicate if benefit funds should be created from child funds.

The optional <**MaximumDuplicate**> attribute specifies the number of times a relationship can be duplicated for a client. A validation will prevent the user from adding more than the maximum number designated and presents the user with the following error message, “Maximum count exceeded for this relationship.” If the <**Maximum**> attribute is not configured, then an infinite number of same client/same primary/same secondary relationships may exist for the Group Customer.

A complete explanation of the elements available for this rule is included in the XML Configuration Guide. An overview of the major elements is provided below.

- The TYPECODE attribute in the <PrimaryRelationship> element defines the primary relationship type from AsCodePrimaryRelationshipType.
- The secondary relationships that can be associated to the primary relationship are defined in the <SecondaryRelationships> section.
  - The <TypeCode> element has a VALUE attribute, which identifies the secondary relationship type from AsCodeSecondaryRelationshipType.
  - The <ClientTypeIdentifier> element is used in conjunction with the main level <ClientTypeIdentifier> element at the bottom of the rule. This element identifies the ID that should be referenced when determining the client types that can be assigned a secondary relationship.
- The <ClientTypeIdentifier> element at the bottom or the rule has an ID attribute, that corresponds to the <ClientTypeIdentifier> element above in the <SecondaryRelationship> secton. Match the value from the above <ClientTypeIdentifier> element to the ID to find the client type.
  - The <ClientType> element in the <ClientTypeIdentifiers> section holds the code that comes from either AsCodeOrganizationType or AsCodePersonType. Only client types defined in this element will be available to assign a secondary relationship.

PlanActivityScreen: This business rule controls the Plan-Level Activity screen. The configuration will determine the number of activities that will be shown on the Plan-Level Activity screen, set the date from which to display activities and provide warnings when using activity icons. This rule may be defined at the Global level or as a Plan level override.

PolicyOverviewScreen: This business rule is used to configure the PolicyOverviewScreen. This screen provides a read only summary of all policy details. It is the first option on the Left Navigation menu and is the default screen view when a policy is loaded in OIPA. Both fixed and dynamic fields from the PolicyScreen can be configured on this screen, as well as new fields, and CopyBooks are supported. All Data Types supported by the Field section of OIPA screen rules are supported in the PolicyOverviewScreen, with the exception of Client and Identifier types. Overrides of this screen are supported at the Global, Subsidiary Company, Plan Group and Plan levels. Screen warning can be configured using Actions, Events and ScreenMath. On Load events for fixed and dynamic fields are also supported. Security is applied at the Plan Page level in the Admin Explorer.

The screen is divided into sections, the order of which is shown below and is set in base code. If a section is not present in configuration, then it will not display on the Policy Overview screen. If a user does not have access to the original page that corresponds to the section, then that section will also not display.

- **Policy Details**: This is the first section of the rule. If this element is present in configuration, but no fields are defined, then the section will appear, but will be blank. Fields seeking data outside of Policy (for example, Client, Segment, Address, etc.) will require a query to populate. For all SQL access, the PolicyGUID must be a known value on the screen. If you are configuring a policy field for display, no query is needed. The field configuration will carry the same field name and data type and will pull the value from the Policy screen. If the field value changes on the Policy screen, the same field value will be reflected on the Policy Overview screen. Field level security and masking are defined in the PolicyOverviewScreen rule and are not inherited from the PolicyScreen rule. Field names determine whether fixed and dynamic fields belong to the Policy screen. Values are initiated from the corresponding Policy fixed or dynamic fields.
- **Policy Roles**: The SHOW="Yes" attribute tells OIPA to display the Policy Roles section. All active roles will display. The <Message> element allows a configured message to be presented to the user on the Policy Roles section in OIPA.
- **Segments**: The SHOW="Yes" attribute tells OIPA to display the Segments section. All segment fields will be displayed. The <Message> element allows a configured message to be presented to the user on the Segments section in OIPA. If Segments configuration is not present, Segment Roles will not be available for display.
- **Segment Roles**: The SHOW="Yes" attribute tells OIPA to display the Segment Roles section. This section displays all information for segment roles that is configured in the SegmentRoleScreen rule. The <Message> element allows a configured message to be presented to the user on the Segment Roles section in OIPA.
- **Values**: The SHOW="Yes" attribute tells OIPA to display the Values section. The <Message> element allows a configured message to be presented to the user on the Values section in OIPA.

PolicyRequirementScreen: This business rule is used to configure OIPA's requirement summary table, which is accessed by clicking the **Requirements** link in the menu on the left side of the screen when an application or policy is open. If this rule is not configured, a default table will be used to display the requirement summary.

PolicySearchScreen: This business rule is used to configure the PolicySearchScreen. It defines the fields that are used to store the results of a search.

RequirementResultSearchScreen: This business rule is used to configure the Requirement Result Search screen, which is used to search for requirement results and, if needed, match them to existing requirements.

RequirementScreen: The RequirementScreen business rule must be configured in order for OIPA to handle requirements properly. This rule will need to be configured as a screen rule. The global rule should only have an empty opening and closing tag. The actual rule is configured as a company level override.

SegmentRoleScreen: This business rule defines the dynamic fields that can be displayed and updated on the specified Role Detail(s) windows. The segment selected during the policy entry process dictates which role options are visible and available on the Segment Role screen. This rule exists at Global and Plan levels. Configuration should only create company level overrides of this rule at the primary company level.

SuspenseScreen: This business rule is used to create and control suspense records. Suspense records are used to track money. This business rule identifies where the money came from and allows for the money to be used as payment to various polices. A suspense record is used as a holding account until the money is applied or refunded. A unique suspense number is generated with the suspense record for identification purposes.

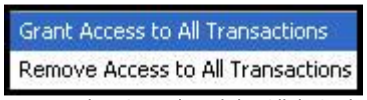
SuspenseSearchScreen: This business rule is used to configure the Suspense Search Criteria section and Results section of the Suspense Search screen. Fields from AsSuspense and AsSuspenseField tables can be used as the suspense search criteria, based on specific suspense records in the database. The Results section can be configured as is the case with other search screens, to present on the UI as a grid using standard table definition syntax.

WithholdingScreen: This business rule defines the layout of the Withholding screen, which signifies the amount or percentage of federal and state taxes to be withheld from taxable disbursements defined by the Policy Owner.

Transaction Security

After Company and Plan security have been defined, the transactions associated with the company and plans are displayed under the Transaction Security folder. Plan folders are listed inside the associated company folder.

Security can be added to all transactions in a plan by right-clicking on the plan name. Security can also be assigned to individual transactions by opening the Plan folder and selecting a specific transaction.


Grant Access to All Transactions
Remove Access to All Transactions
Transaction Security Right-Click Options

Security is applied from the top down. Once Primary Company security is defined in the Company file and the Company pages, then the Plan Security folder will populate with available plans. Once Plan security is defined, then the Transaction Security folder will populate with available transactions.

Explanation of Transaction Security

To open a Transaction Security editor, right-click on the transaction and select **Check-out**. There are four sections that display in the Configuration Area.

If the underlying rule has a context that allows state overrides, the Rules Palette will prompt the user to select a state value from a Context pop-up window.This state value is used to resolve potential copybooks for their field information.

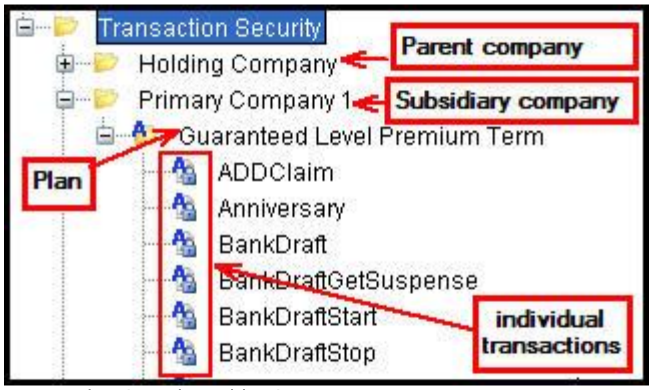
1. **Transaction level security**: grants access to all buttons and fields associated with the transaction by clicking the checkbox at the top of the Configuration Area to the right of the transaction's name.
2. **Button security**: grant access to individual buttons and actions for the transaction by clicking the checkbox to the right of a button. Buttons that are unchecked will not be visible to the user in OIPA. Override buttons receive security from this section. These override buttons are related to the TransactionTimes business rule .

   - **ActivityAddOverride**: controls the display of the Override option on the Add Activity screen when an activity error occurs due to the TransactionTimes business rule.
   - **ActivityDelete**: When an activity is pending this controls the display of the trash can icon to the right of the activity. This icon allows a user to delete an activity.
   - **ActivityDeleteOverride**: controls the ability to override and delete an activity that could not process due to the attached TransactionTimes business rule.
   - **ActivityDetail**; controls the display of activity detail inside the Add Activity window when a new activity is added.
   - **ActivityError**: controls the display of the error icon in the Action column on the Activity screen. This icon displays when an activity cannot process due to errors.
   - **ActivityErrorOkOverride**: controls the display of the OK button on the Error Override window when the activity is halted due to the TransactionTimes business rule. A user cannot override the error without this privilege.
   - **ActivityPostAssignmentError**:controls the display of the PostAssignment error icon in the Action column on the Activity screen. This icon displays when an activity cannot process due to a post assignment validation error post assignment validation error.

- **ActivityPostAssignmentErrorOkOverride**: controls the display of the OK button on the Post Assignment Error Override window. A user cannot override an error without this privilege.
- **ActivityProcess**: controls the display of the lightning bolt icon in the Action column on the Activity screen. This icon allows a user to process an activity.
- **ActivityProcessNUVPending**: controls the display of the lightning bolt icon to process an activity that is in NUV Pending status.
- **ActivityProcessNUVPendingOverride**: controls the display of the override option when an activity in NUVPending fails to process.
- **ActivityProcessOverride**: controls the display of the override option when normal activity processing is interrupted.
- **ActivityRecycle**: controls the display of the recycle icon to the right of the activity. It allows a user to recycle a processed activity.
- **ActivityRequirement**: controls the display of the requirement icon in the Action column on the Activity screen. This icon only applies to activities that have requirements that must be satisfied before the activity can process.
- **ActivityRequirementDeleteOverride**: controls the display of the Delete option in the Activity Requirements window, when TransactionTimes halts activity processing.
- **ActivityRequirementOkOverride**: controls the display of the Override option in the Requirement window. This checkbox is accessed by clicking the requirement icon in the Action column on the Activity screen.
- **ActivityResult**: controls the display of the Activity Detail icon to the left of a processed activity. When clicked it shows the results of the processed activity.
- **ActivityReverse**: controls the display of the trash can icon for activities that have already processed.
- **ActivityReverseOverride**: controls the display of the recycle icon when TransactionTimes halts the processing of an activity.
- **ActivityUpdateOverride**: controls the display of an update option for an activity in pending status that has not been able to process due to the TransactionTimes business rule.
- **Ok**: controls the display of the OK button when the Add Activity window is open.
- **Quote**: controls the display of the Quote button in the Add Activity window when an activity is initially added. This button is only available for Client Financial and Policy Financial transactions.
- **RequirementDelete**: controls the display of the Delete option when a user right-clicks on a requirement from the Requirement window. Click the requirement icon in the Action column on the Activity screen to reveal a list of requirements with right-click menus.
- **RequirementDetail**: controls the display of the Requirement.Detail option when a user right-clicks on a requirement from the Requirement window. Click the requirement icon in the Action column on the Activity screen to reveal a list of requirements with right-click menus.
- **RequirementOK**: controls the display of the OK button when the Requirement window is open.
- **Verify**: controls the display of the Verify button in the Add Activity window when a new activity is initially added. This button is only available for Client Financial and Policy Financial transactions.

3. **Error Overridability Security**: Specifies whether users belonging to the security group have the ability to override individual errors displayed during activity processing. The exact errors that can have their overridability configured on this pane are specified in the ValidateExpressions or PostAssignmentValidateExpressions business rule attached to the transaction. Each error on this pane has a Security drop-down box used for designating whether the error should be overridable by users belonging to the security group.The options available for selection in the Security drop-down box are as follows:

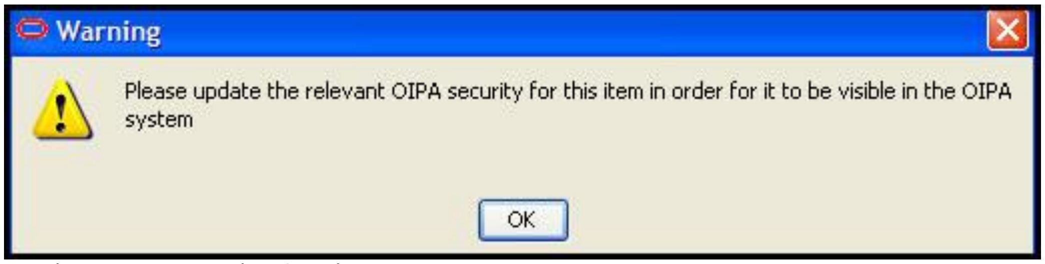
   - **Override Allowed**: Users belonging to the security group will be able to override the error. The corresponding security record will be removed from AsAuthTransactionError, the database table that holds the security data for each error number. By default, all errors are set to Override Allowed for all security groups.
   - **Override Not Allowed**: Users belonging to the security group will not be able to override the error. A security record will be added to the AsAuthTransactionError.
4. **Field security**: grant access to individual fields. There are three options for field security:
   - Visible and Editable: the field is both enabled and the value held in the field is visible. This is the default setting and when selected no database entry will be made.
   - Hide field value: the field is disabled, but the value held in the field is hidden.
   - Disable field value: the field is disabled and any values are visible.
5. **Requirement Fields**: this section will be enabled for transactions that were configured with requirements. Specific levels of access to requirement fields in OIPA are set here.
6. **Masking**: Add security to the mask applied to a text field. Security levels are defined in AsCodeMaskSecurityLevel and AsMaskDetail.



Transaction Security Folder Structure

Security for New Companies, Plans and Transactions

When a new transaction, plan or company is created, it must have security applied to it or it will not be visible in OIPA. Configurors will be able to see the new item in the Rules Palette and can edit it if necessary. OIPA users will not be able to view the new item until the Security Manager assigns security to it.


Warning

Please update the relevant OIPA security for this item in order for it to be visible in the OIPA system

Warning Message to Update Security

The Security Manager must be notified each time a new item is created so that security can be added.

Steps to Assign Security to a New Item

1. Open the Admin Explorer tab in the Rules Palette.
2. Open **Security | Application Security | Security Groups**.
3. Right-click on Security Groups node and select **Refresh**. This will update the folders with the new item that was added. Make sure any security changes previously made are saved before performing refresh.
4. Open the folder for the security group that should receive access to the new item.
5. Open the folder that corresponds to the type of item added. If it is a new company, open the **Company Security** folder. If it is a new plan, open the **Plan Security** folder. If it is a new transaction, open the **Transaction Security** folder.
6. Locate the new item in the folder structure. Double-click on the name to open it in the Configuration Area.
7. Click the check box at the top to grant security to all pages, buttons and fields or expand the individual sections to add security to individual buttons or fields.
8. If assigning security to a transaction, and if masking was added to a text field in the transaction, expand the Masks section and select a security level for the mask.
9. If assigning security to a transaction, and if overridable errors are configured in a ValidateExpressions or PostAssignmentValidateExpressions business rule attached to the transaction, set the overridability of each error by selecting **Override Allowed** or **Override Not Allowed** from each Security drop-down box. By default, all errors will be set to Override Allowed for all security groups.
10. Repeat steps 3 through 9 for all security groups that need access to the new item.
11. Check in all security group files to make sure the changes are saved to the database.

Overridable Error Authorization Prototype

OIPA's ValidateExpressions and PostAssignmentValidateExpressions business rules now support the ability to configure the security roles that are able to override specific errors.

Prototype Explanation

The following transactions and business rules were configured in the Model Prototype Plan, located within the Prototype Company, to demonstrate this new functionality.

Transactions

- **SecurityOverrideForPAVEErrors**: This transaction is configured to return errors based on the value entered in the "Amount" field. To view the prototype configuration, navigate in the Main Explorer to **Companies | Prototype Company | Plans | Functional Prototype Plan | Transactions | SecurityOverrideForPAVEErrors**. The key configuration for this transaction is explained below.
    - An "Amount" field is configured to accept an integer value.
- **SecurityOverrideForVEErrors**: This transaction is configured to return errors based on the value entered in the "Amount" field. To view the prototype configuration, navigate in the Main Explorer to **Companies | Prototype Company | Plans | Functional Prototype Plan | Transactions | SecurityOverrideForPAVEErrors**. The key configuration for this transaction is explained below.
    - An "Amount" field is configured to accept an integer value.

Business Rules

- **ValidateExpressions**: This business rule is attached to the SecurityOverrideForVEErrors transaction. It contains configuration that specifies whether errors returned from the transaction to which it is attached are able to be overridden, as well as the specific errors that are able to be returned from the transaction. To view the prototype configuration, navigate in the Main Explorer to **Companies | Prototype Company | Plans | Model Prototype Plan | Business Rules | Attached Rules | ValidateExpressions**. The key configuration for this business rule is explained below.
    - The <Expressions> element has its OVERRIDABLE attribute set to "Yes," which designates that errors returned from the transaction are able to be overridden.
    - The <Expressions> element's ERRORNUMBER attribute contains two error numbers. These error numbers will appear in the Overridable Errors section of the security group's Transaction Security page.
- **PostAssignmentValidateExpressions**: This business rule is attached to the SecurityOverrideForPAVEErrors transaction. It contains configuration that specifies whether errors returned from the transaction to which it is attached are able to be overridden, as well as the specific errors that are able to be returned from the transaction. To view the prototype configuration, navigate in the Main Explorer to **Companies | Prototype Company | Plans | Model Prototype Plan | Business Rules | Attached Rules | PostAssignmentValidateExpressions**. The key configuration for this business rule is explained below.
    - The <Expressions> element has its OVERRIDABLE attribute set to "Yes," which designates that errors returned from the transaction are able to be overridden.
    - The <Expressions> element's ERRORNUMBER attribute contains two error numbers. These error numbers will appear in the Overridable Errors section of the security group's Transaction Security page.

Security Groups

- **Prototype Super:** This Security Group is configured to be able to override ORY001, but not ORY002 (see the Error Numbers section for an explanation of each error number).
- **Prototype Tester:** This Security Group is configured to be able to override both errors (see the Error Numbers section for an explanation of each error number).

Error Numbers

- **ORY001**: This error will be returned if the user enters a value less than 1000 in the "Amount" field.
- **ORY002**: This error will be returned if the user enters a value less than 500 in the "Amount" field.

View Prototype in OIPA

1. Log in to OIPA using credentials for a user belonging to the Prototype Super security group.
2. Open a policy belonging to the Model Prototype Plan.
3. Add the SecurityOverrideForVEErrors activity.
4. Enter a value under 500 in the Amount field. The system will display error number ORY002 without the ability to override.
5. Enter a value under 1000 in the Amount field. The system will display error number ORY001 with the ability to override.
6. Log out of OIPA and log back in using credentials for a user belonging to the Prototype Tester security group.
7. Repeat steps 2 through 5 above. This time, both errors will be overridable.

Called Event and Call External Event Prototype

OIPA now allows Events in a transaction to generate an Action that results in triggering an event configured in a BR associated with the transaction. This feature is provided to cater to a situation where the action to be performed in the target BR is to be triggered on an event defined in the transaction but the action to be performed is also dependent on the values in one or more fields in the target BR.

This functionality will be supported by defining an Action in the target BR (MultiField BR is used to demonstrate the capability in this case) as a CALLEDEVENT with an ID attribute. Such a called event defined in the target BR can be triggered from the transaction through an Action of type CALLEXTERNALEVENT which is defined in the transaction with the same value in the ID attribute.

Currently this feature is supported in MultiFields BR and TransactionAllocationScreen BR. This may be extended to other associated BRs in future.

Scenario

In a specific transaction, a specific event to assign value or validate values in a Multifield in one or more instances of the multifield is required to be triggered on the basis of a specific event inside the transaction. The event is dependent on the values in both the transaction field and the multifield values.

Prototype Configuration

- The enhancement is added to an existing transaction ‘MultifieldEvents’ in the Functional Prototype Plan under Prototype Company. The events of type ‘CALLEXTERNALEVENT’ are configured in the transaction which in turn invokes the events of type ‘CALLEDEVENT’ present in the ‘ Multifield-MultifieldEvents’.
    - Transaction Field: "TestField" is created as a Transaction field to demonstrate this functionality. Based on the value of TestField and the event type, the corresponding events are triggered in the Multifield.
    - Transaction Events: The following events are configured to demonstrate the functionality.
        - OnLoad: In the transaction, on load event, the Action of type ‘CALLEXTERNALEVENT’ is configured which invokes the event ‘MFEventOnLoad’ present in the MultiField –MultiFieldEvents business rule.
        - OnChange: In the transaction, the Action of type ‘CALLEXTERNALEVENT’ is configured which invokes the event ‘MFTestTextEventOnChange’ and ‘MFTestComboEventOnChange’ present in the MultiField –MultiFieldEvents business rule.
        - OnSubmit: In the transaction, the Action of type ‘CALLEXTERNALEVENT’ is configured which invokes the event ‘MFEventOnSubmit’ present in the MultiField – MultiFieldEvents business rule.
    - Multifield Events:This is an existing multifield business rule, where four multifields-TextTestMF, ComboTestMF, RadioTestMF and CheckTestMF are configured.The events invoked by transaction are configured in the Multifield-MultiFieldEvents multifield. The following events are configured in the same:
        - MFEventOnLoad: This event is associated with the multifield CheckTestMF. It invokes actionset MFActionOnLoad which displays a warning message and marks the field CheckBoxTest1 as ‘CHECKED’ on load.
        - MFTestTextEventOnChange: This event is associated with multifield TextTestMF. Based on the following values provided to the field TestText2 of index 0, the corresponding action is configured for the field TestText1 of index 0:
            - Hide
            - Show

    - Disable
    - Enable
    - Assign
    - ReadOnly
    - Test - Display and error message.
  - MFTestComboEventOnChange: This event is associated with multifield ComboTestMF. It invokes queryset “TestComboOptionsActionOnChange” for field “ComboTest5”.
  - MFEventOnSubmit: This event is configured for multifield RadioTestMF. It invokes actionset MFActionOnSubmit which displays a warning message.

View Prototype in OIPA

1. Log in OIPA using the Prototype Company user ID and password.
2. Click **Policy | New** from the Main menu.
3. Add a shell policy to test the prototype.
4. Click **Add Activity** on the Secondary menu.
5. Select the MultifieldEvents activity from the Activity drop down box.
6. Make changes to the various fields view the action event triggered as explained above to view this functionality in action.

Multifield Value in Math and ScreenMath Prototype

The MultifieldEvents prototype demonstrates how OIPA allows picking up the values of Multifield values from specific instances in math variables in the ScreenMath and Math sections in transactions.

Scenario

In a specific transaction, the multifield value entered in the second instance of the Multifield is required to be used for validation in screen math and display in the verification screen. The value will need to be picked up in math variables in ScreenMath and Math to perform the same.

Prototype Configuration

- The existing transaction ‘MultifieldEvents’ in the Functional Prototype Plan is used for demonstrating the functionality associated with this feature. The following features are configured to demonstrate the same functionality.
    - A math variable “TextTestMV” of type MultiField is added in the Math section of the transaction which fetches the value from the associated Multifield rule “Multifield-MultifieldEvents” from Multifield “TextTestMF” at index 1. This can be viewed in the Math Variables section after processing the activity.
    - A VerificationScreen BR is attached to this transaction where the above created math variable “TextTestMV” is accessed as a field. This can be viewed during Verify.
    - A ValidateExpression BR is also attached to this transaction which will perform validation based on the math variable “TestTextMV” and display errors as appropriate while processing the transaction.

View Prototype in OIPA

1. Log in OIPA using the Prototype Company user ID and password.
2. Click **Policy | New** from the Main menu.
3. Add a shell policy to test the prototype.
4. Click **Add Activity** on the Secondary menu.
5. Select the MultifieldEvents activity from the Activity drop down box.
6. Make changes to the various fields and press verify. The verification screen will display the value entered in the multifield TextTestMF at index 1. Enter any text other than 'New Text' in TextTestMF at index 1 and an error message will also be displayed on the verification screen.
7. Enter value 'New Text' in TextTestMF at index 1 and process the transaction. Click on the activity results and go to the Math tab to see the multifield value from TextTestMF at index 1 being available in the math variable TextTestMV.

Scheduled Valuation Configuration

The purpose of Scheduled Valuation is to perform a policy valuation at a specific time for a group of policies and store that valuation for subsequent use. The time intervals for running the valuation are set by the user. The frequency at which policies are subject to valuation is also set by the user. Typical time intervals are quarterly, semi or annually. Multiple currencies are supported in Scheduled Valuation and policy information such as value and deposit information can be stored in the database.

A plan level transaction is configured to perform scheduled valuation. There are two business rules that should be attached to the transaction: **ScheduledValuation** and **CopyToScheduledValuationFields**.

Scenario

Use a plan level transaction to perform scheduled valuation on all policies within the plan.

> Cycle is involved in scheduled valuation. A cycle agent has to be deployed; either in a web container like Weblogic or Websphere, or as a standalone application, and must be running in order for scheduled valuation to work correctly. Refer to the Cycle document in the Documentation Library on OTN for additional information on cycle.

Configuration Requirements

The following components must be configured in order to perform scheduled valuation:

- a plan must be created and the PlanScreen business rule must be configured and segments must be defined.
- funds must be created and associated with the plan.
- the InterestRateCalculation business rule must be overridden for each fixed fund.
- the PolicyScreen business rule must be overridden at the plan level.
- a plan-financial transaction with the <Asynchronous> element must be configured. The fields and math should contain data that all scheduled valuation records will receive via the configuration in the CopyToScheduledValuationFields business rule.
- the ScheduledValuation business rule with the PlanGUID for the plan where scheduled valuation should occur. This should be attached to the transaction.
- the CopyToScheduledValuationFields business rule, which should be attached to the transaction.
- the PolicyValues business rule overridden at the plan level. This is where individual values for each policy executing through ScheduledValuation are calculated. These may be accessed by CopyToSechededuledValuationFields.
- Cycle agent must be deployed and running.

Optional Requirements

If Point-in-Time valuation is needed, then the following additional business rules and tables need to be configured:

- the PointinTimeValuation business rule should be overridden at the plan level
- the WriteValuationElements business rule should be overridden at the plan level
- AsPlan table column named PointInTimeValuation must be set to a value of **Y**(or **T** if the plan is transitioning from Traditional valuation)
- DepositLevelTracking table column named AsFund must be set to a value of **N**

Database Tables

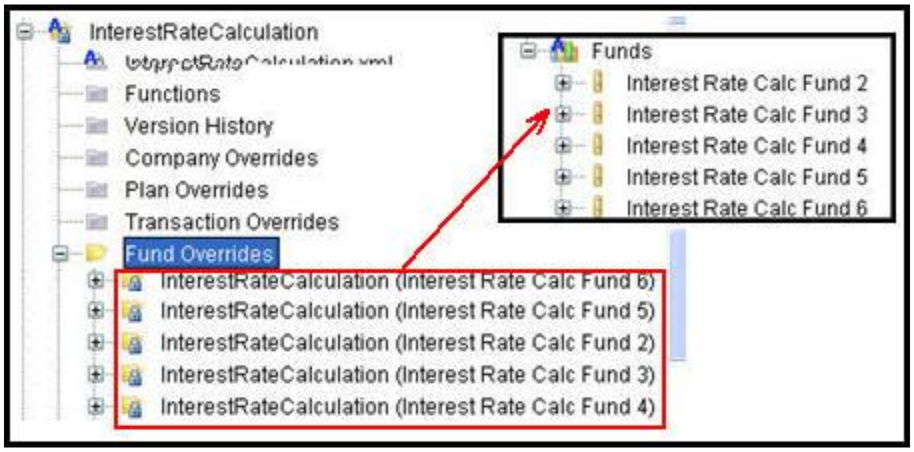
The following database tables were created to support scheduled valuation:

- AsScheduledValuation: Contains details of the scheduled valuation computed by the system.
- AsScheduledValuationDeposit: Contains details of deposits associated with a scheduled valuation.
- AsScheduledValuationField: Contains details of dynamic fields associated with a scheduled valuation.
- AsScheduledValuationFund: Contains details of funds associated with a scheduled valuation.
- AsValuesRequest: Contains records used for tracking the execution of batch valuation of policies, such as a valuation's start time and end time.

> **Note**: A valuation will not have its end time recorded if the valuation does not complete successfully.

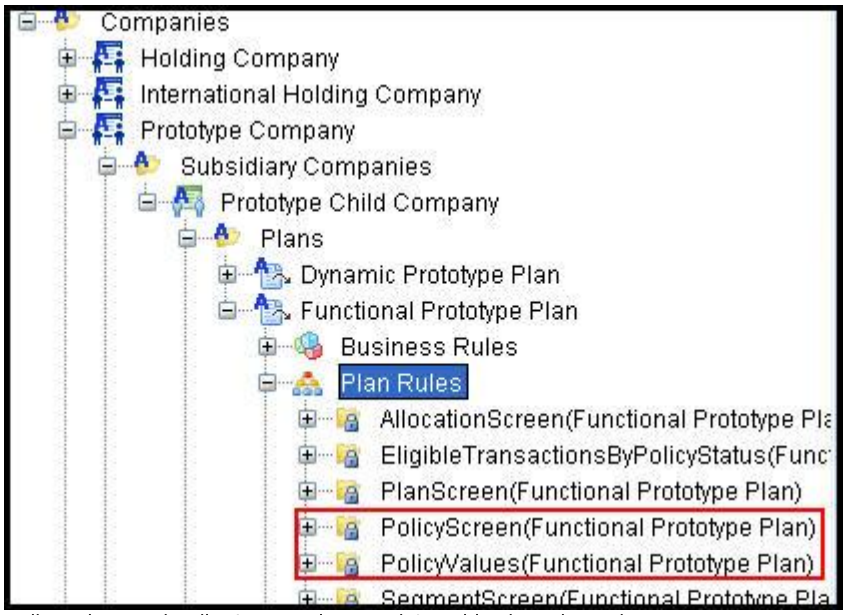
Prototype Samples

Funds need to be created, along with an InterestRateCalculation business rule override for each fund. The Funds are displayed in the Main Explorer in **Companies | Prototype Company | Subsidiary Companies | Prototype Child Company | Plans | Functional Prototype Plan | Funds.** The rule overrides are displayed in the Global Rules Explorer in **Business Rules | System | InterestRateCalculation | Fund Overrides.**


InterestRateCalculation
InterestRateCalculation.xml
Functions
Version History
Company Overrides
Plan Overrides
Transaction Overrides
Fund Overrides
InterestRateCalculation (Interest Rate Calc Fund 6)
InterestRateCalculation (Interest Rate Calc Fund 5)
InterestRateCalculation (Interest Rate Calc Fund 2)
InterestRateCalculation (Interest Rate Calc Fund 3)
InterestRateCalculation (Interest Rate Calc Fund 4)
Funds
Interest Rate Calc Fund 2
Interest Rate Calc Fund 3
Interest Rate Calc Fund 4
Interest Rate Calc Fund 5
Interest Rate Calc Fund 6
Funds from Main Explorer and InterestRateCalculation Overrrides from Global Rules Explorer

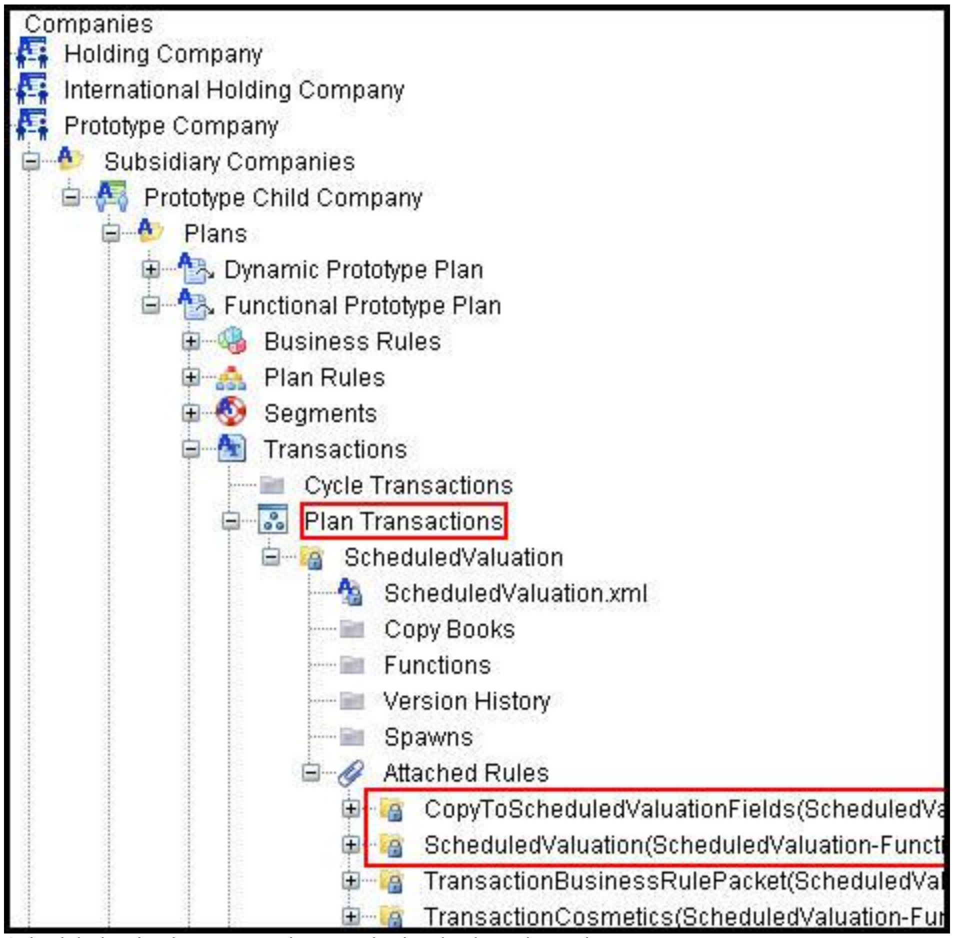
There are several business rules used to demonstrate this configuration:

- PolicyScreen business rule: this rule must be overridden at the plan level. Two new fields are needed: one for InterestBonusQual and one for Effective date. Navigate in the Main Explorer to **Companies | Prototype Company | Subsidiary Companies | Prototype Child Company | Plans | Functional Prototype Plan | Plan Rules.**
- PolicyValues business rule: this rule should be overridden at the plan level. Navigate in the Main Explorer to **Companies | Prototype Company | Subsidiary Companies | Prototype Child Company | Plans | Functional Prototype Plan | Plan Rules.**


Companies
Holding Company
International Holding Company
Prototype Company
Subsidiary Companies
Prototype Child Company
Plans
Dynamic Prototype Plan
Functional Prototype Plan
Business Rules
Plan Rules
AllocationScreen(Functional Prototype Pla
EligibleTransactionsByPolicyStatus(Func
PlanScreen(Functional Prototype Plan)
PolicyScreen(Functional Prototype Plan)
PolicyValues(Functional Prototype Plan)
SegmentScreen(Functional Prototype Pla

PolicyValues and PolicyScreen Plan Level Overrides in Main Explorer

- ScheduledValuation business rule: this rule should be overridden at the transaction level and attached to the transaction. Navigate in the Main Explorer to **Companies | Prototype Company | Subsidiary Companies | Prototype Child Company | Plans | Functional Prototype Plan | Transactions | Plan Transactions | ScheduledValuation | Attached Rules.**
- CopyToScheduledValuationFields business rule: this rule should be overridden at the transaction level and attached to the transaction. Navigate in the Main Explorer to **Companies | Prototype Company | Subsidiary Companies | Prototype Child Company | Plans | Functional Prototype Plan | Transactions | Plan Transactions | ScheduledValuation | Attached Rules.**

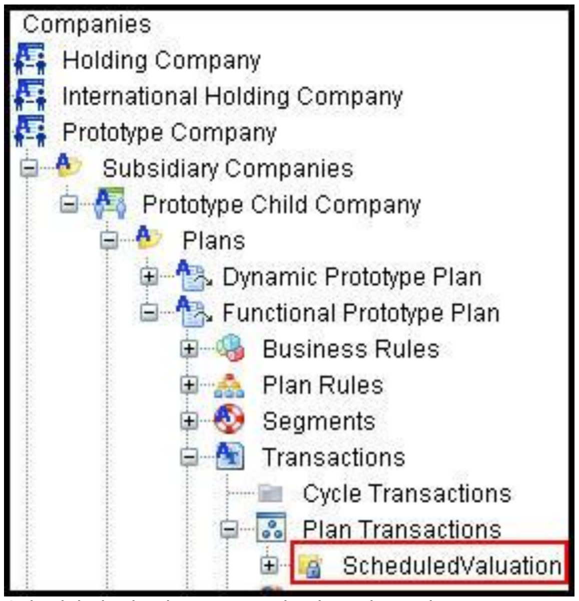

Companies
Holding Company
International Holding Company
Prototype Company
Subsidiary Companies
Prototype Child Company
Plans
Dynamic Prototype Plan
Functional Prototype Plan
Business Rules
Plan Rules
Segments
Transactions
Cycle Transactions
Plan Transactions
ScheduledValuation
ScheduledValuation.xml
Copy Books
Functions
Version History
Spawns
Attached Rules
CopyToScheduledValuationFields(ScheduledVa
ScheduledValuation(ScheduledValuation-Functi
TransactionBusinessRulePacket(ScheduledVal
TransactionCosmetics(ScheduledValuation-Fun

Scheduled Valuation Transaction Attached Rules in Main Explorer

- InterestRateCalculation rule: this rule was discussed in the fund section and should be overridden at the fund level for each fund.

There is one plan level transaction configured to demonstrate scheduled valuation:

- ScheduledValuation: this transaction must include the <AsynchronousActivity> element. Navigate in the Main Explorer to **Companies | Prototype Company | Subsidiary Companies | Prototype Child Company | Plans | Functional Prototype Plan | Transaction | Plan Transactions** to view the configuration.



ScheduledValuation Transaction in Main Explorer

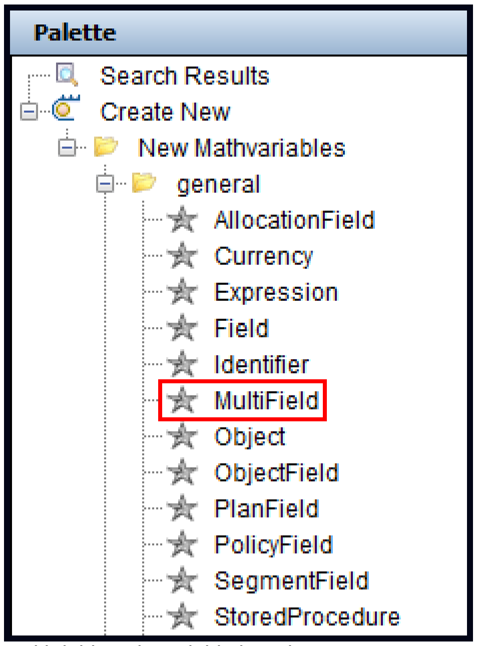
MultiField

The MULTIFIELD math variable is used to access multifield values (using INDEX as an attribute) in Screen Math and Math sections.

The INDEX attribute can be any integer value ranging from 0-99. When the INDEX attribute is equal to 0, the first instance of a multifield will be retrieved. Each additional instance of a multifield will be accessed by incrementing the INDEX value in a subsequent math variable. A math variable that is configured with an INDEX greater than the number of instances of the multifield selected on the activity will result in the value of the math variable being equal to null.

For math variables that are data type STRING or DATE, null is a valid value and no error will be returned. For math variables that are data type DECIMAL or INTEGER null is not a valid value and a system error will be returned, unless a default value is specified using configuration.

A math variable that is configured with the TYPE equal to MULTIFIELD, but has no INDEX attribute specified will result in a code-generated system error.



MultiField Math Variable in Palette

Note: For a comprehensive explanation of the characteristics of this math variable, refer to the XML Configuration Guide in the **Help** menu of the Rules Palette.

Define the Field Math Variable

Drag and drop the MultiField Math variable from the Palette window onto the Math Pane. Enter the following information for the math variable:

- Type a **Name** in the Name field. This is required.
- The data types are Activity, BigText, Boolean, Date, Decimal, Integer, Map, Object, Text and XML. This field is required.
- Make sure variable type is defined as MULTIFIELD. This is required.
- Specify an index. This specifies the particular instance of the multifield to access.
- Select Yes or No from the **Log** field. The default is No. This is an optional field.

- **Round** offers the options of **Yes** or **No**. Round is only enabled if you select the Decimal in the data type property. If you would like to round the Decimal, select **Yes**. Enter the number of decimal places to round the result value. **Note**: When the data type of Decimal is selected, the default value for Round is set to No. You must change the selection of the radio button to Yes if you wish to round.
- **Default** is used for a Decimal or Integer value so the system will not return a null value. For example, the default for an Integer data type could be -999999999.
- **MultiField** is the name of the MultiField you wish to retrieve. Type in the Name of the field.

Check the transaction in to save the information to the database.

Suspense Overview

Suspense and accounting functionality (Chart of Accounts in Admin Explorer) can work in tandem or separately. Best practice is to implement both and have them work together to provide information on financial activities related to a company.

The Prototype Company provides an example of suspense processing that supports multiple currencies. View the prototype example for additional information.

How Suspense Works

Money can be posted directly to a policy or entered into a suspense record. Part of the suspense record or the entire record can be attached to a policy.

Accounting processes based on criteria indicators set on the Chart of Accounts and the optional ChartOfAccountsSpecifications rule. The ChartOfAccountsSpecifications rule allows the use of indicator values on the suspense record that control which suspense account(s) the money is apply to.

If money cannot be applied directly to a policy, then it is usually entered as a suspense record at the company level through the Suspense screen, which may also invoke account processing.

When the entire suspense amount is disbursed, the suspense item is closed. If only a portion of the suspense item is disbursed, then the suspense item remains open. Any disbursed amount is added to the suspense record's attached amount.

If the disbursement activity is reversed, then the amount of the disbursement is returned to suspense and the suspense is be reopened if it had been closed. The amount returned is subtracted from the suspense record's attached amount.

Suspense Accounts

A suspense account is an account that is used to temporarily store money until a decision is made about where the money will be allocated. It identifies where the money came from and allows the money to be used as payment to one or more policies. Suspense accounts are set-up in the Rules Palette from the Admin Explorer tab in the Chart of Accounts folder.

Suspense Records

Suspense records can be created from the Suspense screen in OIPA, or they can be generated by the GenerateSuspense business rule when it is attached to a transaction that is processed in OIPA. If a suspense record is created on the Suspense screen, it can then be selected on the Suspense Search tab of a transaction's Activity Detail screen for use by that transaction, provided the transaction is configured to display the Suspense Search tab.

A transaction may be both configured to display the Suspense Search tab of the Activity Detail screen and have the GenerateSuspense business rule attached. In this case, if a suspense record is selected from the Suspense Search tab, that record will be attached to the transaction. If a suspense record is not selected from the Suspense Search tab, then the GenerateSuspense business rule will be used to generate a new suspense record, which will then be attached upon the processing of the transaction.

A unique suspense number is generated with each suspense record for identification purposes. Suspense records are written to the suspense account(s) for a company's general ledger. These records temporarily hold money until an activity generates a process to disperse or apply the money. Suspense records are created in OIPA and are associated with existing suspense accounts.

High Level View of Suspense Configuration

The following list provides an overview of the major steps involved in setting up suspense in OIPA. Follow the links to pages for more information on each specific step in the process.

1. Set up suspense accounts in the Admin Explorer using Chart of Accounts.
2. Configure Suspense Screens.
   - Configure the Suspense Screen and/or Suspense Screen Overrides.
   - Configure the Suspense Search Screen.
3. Configure Suspense Refund
   - Configure the transaction that will activate the refund. Typically client level activities are used for suspense refunds.
   - Configure the suspense refund number in the transaction.
   - Configure the suspense section in the transaction.
   - Configure the disbursement section in the transaction.
4. Configure Suspense Accounting.
   - Configure the suspense or multisuspense element in a transaction that is configured to apply money to policies from a suspense account. OIPA uses the suspense element to capture the suspense number and amount from an activity and apply it to the associated suspense record. The suspense record's Attached Amount is updated by adding the amount entered in the activity to the Attached Amount. The Attached Amount starts at zero, and once the amount equals the suspense amount, the record is closed.
   - Configure the GenerateSuspense business rule if suspense should automatically be generated for activities that move money into a policy. This rule creates accounting detail for a supplied field amount in order to establish a relationship between the amount and a suspense record.
   - Configure the MaintainSuspense business rule if the user needs to be able to change suspense field values and generate accounting through a collection of multiple suspense tickets. This rule should be attached to a transaction. Refer to the XML Configuration guide for a complete explanation of the elements and attributes for this rule.

UPDATES TO THE XML CONFIGURATION GUIDE

This section contains pages from the XML Configuration Guide that were updated for the 9.6.1.0 release.

CopyToClientFields (For Transactions)

Description

This business rule is attached to a transaction to allow one or more MathVariables to be copied from an activity to one or more client fields when the activity to which this rule is attached is processed.
In addition to field values, CopyToClientFields will automatically update the OptionText of combo box and radio button fields.

Note: The CopyToClientFields rule cannot be used to update external client information.

CopyToClientFields Element/Attribute Table

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
<CopyToClientFields>	Required opening and closing tag of the CopyToClientFields rule. **Note**: CLIENTGUID attribute is optional.	CLIENTGUID	**Optional attribute: MathVariable** This attribute is used to specify the ClientGUID of the client whose records are to be updated. **Note**: Transaction to which this business rule is attached should contain the MathVariables that capture the ClientGUIDs.
<Fields>	**Required element:** Used to specify the client fields that will be updated by values from MathVariables.		
<Field>	**Repeatable element:** The opening and closing tag that encompasses <From> and <To>.		If a <From> element is present, then a <To> element should be present. Similarly, a <FromCollection> element must have a <To> element.
<From>	This element is used to specify the MathVariable or activity field where the data should be copied from.		**Required element value: MathVariable/ActivityField** Name of MathVariable or field from the transaction. Not required if using FromCollection and no CLIENTGUID attribute is specified on the opening element.
<To>	**Required element:** This element is used to specify the fields in the Client screen where the data will be copied to.		**Required element value: Field** Name of the field in the Client screen .

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
<FromCollection>	**Require element:** This element is required when no CLIENTGUID attribute has been specified on the opening element. The collection consists of Client GUIDs as the key and data value for the indicated <To> field.		**Required element value:** MathVariable of variable type Collection.
<Client>	**Optional, repeatable element:** Used to copy data to a particular client or clients.	CLIENTGUID	When the CLIENTGUID attribute is used with the <Client> element, CLIENTGUID and POLICYROLES attributes MUST not be used in the <CopyToClientFields> element.
<Fields>	See <Fields> element above.		

XML Example

```
<CopyToClientFields CLIENTGUID="ClientGUIDMV">
      <Fields>
            <Field>
                  <From>NewFirstNameMV</From>
                  <To>NewFirstNameField</To>
            </Field>
            <Field>
                  <From>NewLastNameMV</From>
                  <To>NewLastNameField</To>
            </Field>
      </Fields>
</CopyToClientFields>
```

XML Schema

```
<CopyToClientFields CLIENTGUID="[String]">
      <Fields>
            <Field>
                  <From\>
                  <To\>
            </Field>
            <Field>
                  <FromCollection\>
                  <To\>
            </Field>
      </Fields>
</CopyToClientFields>
```

CopyToAddressFields

Description

This business rule allows one or more MathVariables to be copied from an activity to one or more Address fields.
In addition to field values, CopyToAddressFields will automatically update the OptionText of combo box and radio button fields.

CopyToAddressFields Element/Attribute Table

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
<CopyToAddressFields>	The opening and closing tags of this rule.	ADDRESSGUID	**String:**
<Fields>	Allows configuration of dynamic fields.		
<Field>	**Repeatable element**; The opening and closing tag that encompasses <From> and <To>		
<From>	Name of the Math Variable or activity field data is being copied from.		**Required element value**; **MathVariable/ActivityField** Name of the MathVariable or field from the transaction. Not required if using FromCollection and no CLIENTGUID attribute is specified on the opening element.
<FromCollection>	Name the collection from which data is copied.		**Required element value:** MathVariable of variable type Collection.
<To>	Name of Address field data is being copied to.		**Required element value Field**; Name of the field in Client screen .

XML Example

```
<CopyToAddressFields ADDRESSGUID="AddressGUID">
      <Fields>
```

```
            <Field>
                  <From>AddressLineOne</From>
                  <To>AddressLine1</To>
            </Field>
            <Field>
                  <From>AddressEffectiveDate</From>
                  <To>EffectiveDate</To>
            </Field>
            <Field>
                  <From>MVNationCode</From>
                  <To>NationCode</To>
            </Field>
            <Field>
                  <From>MVReturnMailIndicator</From>
                  <To>ReturnMailIndicator</To>
            </Field>
      </Fields>
</CopyToAddressFields>
```

CopyToPolicyFields

Description

This business rule is attached to a transaction to allow one or more MathVariables to be copied from the activity or requirement to one or more policy fields. If the fields are displayed on the Policy screen, the values will be viewable.

In addition to field values, CopyToPolicyFields will automatically update the OptionText of combo box and radio button fields.

CopyToPolicyFields Element and Attribute Table

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
<CopyToPolicyFields>	The required opening and closing elements of this business rule.		
<Fields>	**Required element**: Identifies the Field section.		
<Field>	**Repeatable element:** The <Field> tag is used to update a field in AsPolicyField table by passing the required information from the transaction or requirement.		
<From>	This element is used to specify the MathVariable or field from which the data should be copied.		The name of the activity field or MathVariable that data is being copied from.
<To>	This element is used to specify the field on the Policy screen to which the data should be copied.		The name of the PolicyField that data is being copied to. The value of the <To> tag will be saved in the AsPolicy or AsPolicyField database table.

XML Example

```
<CopyToPolicyFields>
      <Fields>
            <Field>
                  <From>Activity:ReinsuranceIndicator</From>
                  <To>ReinsuranceIndicator</To>
            </Field>
      </Fields>
</CopyToPolicyFields>
```

XML Schema

```
<CopyToPolicyFields TYPE="IFEMPTY">
      <Fields>
            <Field>
                  <From>[String]</From>
                  <To>[String]</To>
            </Field>
      </Fields>
</CopyToPolicyFields>
```

ORACLE INSURANCE

CopyToRoleFields

Description

This business rule allows one or more MathVariables or Activity fields to be copied to one or more specified RoleFields upon processing the activity to which the CopyToRoleFields business rule is attached. Configuration has the option to update multiple roles from a single attribute, multiple roles by a singular role code, and one or more roles using collections. This rule may be used to update policy role fields, segment role fields, or both.

In addition to field values, CopyToRoleFields will automatically update the OptionText of combo box and radio button fields.

This rule must be listed in TransactionBusinessRulePacket.

CopyToRoleFields Element/Attribute Table

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
<CopyToRoleFields>	The opening and closing tag of the CopyToRoleFields business rule.		
<PolicyRoles>	A section identifying the policy roles and the fields that are to be updated by the rule. PolicyRoles configuration is not required.		The rule can be configured for Segment role update only, Policy role update only, or update of both types of roles.
<PolicyRole>	**Required and Repeatable:** A section to identify roles that are to be updated.		Either ROLECODE or ALLROLES must be provided.
	Optional: RoleCode that is used to target a role for update. Cannot coexist with ALLROLES attribute or FromCollection element.	ROLECODE	A singular Role Code; expected population via MathVariable.
	Optional: Cannot exist with ROLECODE attribute. Cannot coexist with <FromCollection> element.	ALLROLES	**Yes|No** If Yes, all Policy roles (excluding CSR) meeting the status criteria will be updated.
<Tests>	**Optional:** Allows for additional filtering of the roles that may be updated, or creates additional MathVariable or activity field		

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
	triggers that allow role updates by creating conditions outside of role status. All conditions in this section must evaluate to true before the policy's role may be updated.		
<Test>	**Repeatable element**.		A conditional statement that tests a MathVariable or Field against another MathVariable or Field or other literal value.
<RoleStatus>	**Optional:** Opening tag to list role status information. Status codes in the list further filter the roles that may be updated. Without this information, the filter is not applied and the status is disregarded.		
<Status>	**Required, Repeatable:** A role status that can accept the role field update.		A valid Role status code.
<Fields>	**Required/Repeatable:** Defines a section to contain the fields that will be updated.		
<Field>	**Repeatable:** Identifies the field that is updated and the updated value. <From> and <FromCollection> elements are mutually exclusive.		
<From>	**Required:** Identifies the math variable or activity field as the source of the update. An activity field must be prefixed with Activity:[field name].		A MathVariable or activity field from the transaction to which the rule is attached.
<FromCollection>	**Required:** Identifies a MathVariable of type COLLECTION as the source of the update.		A MathVariable of type COLLECTION where the keys are role GUIDs and the values are the source data.
<To>	**Required:** Identifies the field that is updated.		A literal name for a field.

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
<SegmentRoles>	**Optional:** A section identifying the segment roles and the fields that are to be updated by the rule.		Segment roles configuration is not required. This rule can be configured for Policy role update only, Segment role update only, or update of both role types.
<SegmentRole>	**Optional and Repeatable:** A section to identify roles that are to be updated.		Either ALLROLES or ROLECODE attribute must be provided.
	Optional: If Yes, all Segment roles meeting the status criteria will be updated. The CSR role is excluded from ALLROLES update.	ALLROLES	**Yes | No**
	Optional: This is used to target a singular RoleCode for update. Cannot coexist with ALLROLES attribute or <FromCollection>.	ROLECODE	One Role Code
	Required: Used with ROLECODE or ALLROLES. Cannot coexist with <FromCollection>.	SEGMENTGUID	One SegmentGUID
<Tests>	**Optional:** Allows for additional filtering of the roles that may be updated, or creates additional MathVariable or activity field triggers that allow role updates by creating conditions outside of role status. All conditions in this section must evaluate to true before the segment's role may be updated.		
<Test>	**Repeatable**		A conditional statement that tests a MathVariable or Field against another MathVariable, Field, or literal value.
<RoleStatus>	**Optional**; Opening tag to list role status		

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
	information. Status codes in the list further filter the roles that may be updated. Without this information, the filter is not applied and the status is disregarded.		
<Status>	**Required and Repeatable:** A role status that can accept the role field update.		A valid Role status code
<Fields>	**Required/Repeatable**; Defines a section to contain the fields that will be updated.		
<Field>	**Repeatable**; Identifies the field that is updated and the updated value. <From> and <FromCollection> elements are mutually exclusive.		
<From>	**Required**; Identifies the MathVariable or Activity field as the source of the update. An activity field must be prefixed with Activity:[field name].		A MathVariable or Activity field from the transaction to which the rule is attached.
<FromCollection>	**Required:** Identifies a MathVariable of type COLLECTION as the source of the update.		A MathVariable of type COLLECTION where the keys are role GUIDs and the values are the source data.
<To>	**Required:** Identifies the field that is updated.		A literal name for a field.

XML Sample

```
<CopyToRoleFields>
      <PolicyRoles>
            <PolicyRole ALLROLES="Yes">
                  <Tests>
                        <Test>IsPolicyTerm=true</Test>
                  </Tests>
                  <RoleStatus>
                        <Status>01</Status>
                  </RoleStatus>
                  <Fields>
```

```
                        <Field>
                              <From>Activity:RoleRelationship</From>
                              <To>RoleRelationship</To>
                        </Field>
                        <Field>
                              <From>SystemDateMV</From>
                              <To>RoleTestDate</To>
                        </Field>
                  </Fields>
            </PolicyRole>
      </PolicyRoles>
      <SegmentRoles>
            <SegmentRole ALLROLES="Yes" SEGMENTGUID="Activity:WhichSegment">
                  <Tests>
                        <Test>IsSegmentTypeNot72=true</Test>
                  </Tests>
                  <RoleStatus>
                        <Status>01</Status>
                  </RoleStatus>
                  <Fields>
                        <Field>
                              <From>Activity:RoleRelationship</From>
                              <To>RoleRelationship</To>
                        </Field>
                        <Field>
                              <From>SystemDateMV</From>
                              <To>RoleTestDate</To>
                        </Field>
                  </Fields>
            </SegmentRole>
      </SegmentRoles>
</CopyToRoleFields>
```

XML Schema

CopyToRoleFields – PolicyRole Schema

```
<CopyToRoleFields>
      <PolicyRoles>
            <PolicyRole ALLROLES="Yes|No" ROLECODE="MathVariable">
               <Tests>
                  <Test>conditional statement</Test>
               </Tests>
               <RoleStatus>
                  <Status>[role status]</Status>
               </RoleStatus>
               <Fields>
```

```
                  <Field>
                        <FromCollection>[activity field|math
                        variable]</FromCollection>
                        <To>[column name|field name]</To>
                  </Field>
                  <Field>
                        <From>[activity field|math variable]</From>
                     <To>[column name|field name]</To>
                  </Field>
               </Fields>
            </PolicyRole>
      </PolicyRoles>
</CopyToRoleFields>
```

XML Schema – CopyToRoleFields – Segment Role Schema

```
<CopyToRoleFields>
      <SegmentRoles>
            <SegmentRole ALLROLES="Yes|No" ROLECODE="MathVariable"
            SEGMENTGUID=”MathVariable”>
                  <Tests>
                        <Test>conditional statement</Test>
                  </Tests>
                  <RoleStatus>
                        <Status>[role status]</Status>
                  </RoleStatus>
                  <Fields>
                        <Field>
                              <FromCollection>[activity field|math
                              variable]</FromCollection>
                              <To>[column name|field name]</To>
                        </Field>
                        <Field>
                              <From>[activity field|math variable]</From>
                              <To>[column name|field name]</To>
                        </Field>
                  </Fields>
            </SegmentRole>
      </SegmentRoles>
</CopyToRoleFields>
```

XML Schema Policy Roles – ALLROLES =Yes

```
<CopyToRoleFields>
      <PolicyRoles>
            <PolicyRole ALLROLES="Yes">
                  <Tests>
                        <Test>conditional statement</Test>
                  </Tests>
                  <RoleStatus>
                        <Status>[role status]</Status>
                  </RoleStatus>
                  <Fields>
                        <Field>
                              <From>[activity field|math variable]</From>
                              <To>[column name|field name]</To>
                        </Field>
                  </Fields>
            </PolicyRole>
      </PolicyRoles>
</CopyToRoleFields>
```

XML Schema Segment Roles – ALLROLES =Yes

```
<CopyToRoleFields>
      <SegmentRoles>
            <SegmentRole ALLROLES="Yes" SEGMENTGUID=”MathVariable”>
                  <Tests>
                        <Test>conditional statement</Test>
                  </Tests>
                  <RoleStatus>
                        <Status>[role status]</Status>
                  </RoleStatus>
                  <Fields>
                        <Field>
                              <From>[activity field|math variable]</From>
                              <To>[column name|field name]</To>
                        </Field>
                  </Fields>
            </SegmentRole>
      </SegmentRoles>
</CopyToRoleFields>
```

XML Schema Policy Roles – ROLECODE Attribute Used

```
<CopyToRoleFields>
      <PolicyRoles>
            <PolicyRole ROLECODE="MathVariable">
                  <Tests>
                        <Test>conditional statement</Test>
                  </Tests>
                  <RoleStatus>
                        <Status>[role status]</Status>
                  </RoleStatus>
                  <Fields>
                        <Field>
```

```
                              <From>[activity field|math variable]</From>
                              <To>[column name|field name]</To>
                        </Field>
                  </Fields>
            </PolicyRole>
      </PolicyRoles>
</CopyToRoleFields>
```

XML Schema Segment Roles – ROLECODE Attribute Used

```
<CopyToRoleFields>
      <SegmentRoles>
            <SegmentRole ROLECODE="MathVariable" SEGMENTGUID=”MathVariable”>
                  <Tests>
                        <Test>conditional statement</Test>
                  </Tests>
                  <RoleStatus>
                        <Status>[role status]</Status>
                  </RoleStatus>
                  <Fields>
                        <Field>
                              <From>[activity field|math variable]</From>
                              <To>[column name|field name]</To>
                        </Field>
                  </Fields>
            </SegmentRole>
      </SegmentRoles>
</CopyToRoleFields>
```

XML Schema – Policy Roles ALLROLES, ROLECODE, and SEGMENTGUID Attributes Omitted; <FromCollection> Used

```
<CopyToRoleFields>
      <PolicyRoles>
            <PolicyRole>
                  <Tests>
                        <Test>conditional statement</Test>
                  </Tests>
                  <RoleStatus>
                        <Status>[role status]</Status>
                  </RoleStatus>
                  <Fields>
                        <Field>
                              <FromCollection>[math
                              variable]</FromCollection>
                              <To>[column name|field name]</To>
                        </Field>
                  </Fields>
            </PolicyRole>
      </PolicyRoles>
</CopyToRoleFields>
```

XML Schema Segment Roles – ALLROLES, ROLECODE, and SEGMENTGUID Attributes Omitted; <FromCollection> Used

```
<CopyToRoleFields>
      <SegmentRoles>
            <SegmentRole>
                  <Tests>
                        <Test>conditional statement</Test>
                  </Tests>
                  <RoleStatus>
                        <Status>[role status]</Status>
                  </RoleStatus>
                  <Fields>
                        <Field>
                              <FromCollection>[math
                              variable]</FromCollection>
                              <To>[column name|field name]</To>
                        </Field>
                  </Fields>
            </SegmentRole>
      </SegmentRoles>
</CopyToRoleFields>
```

CopyToSegmentFields

Description

This business rule is used to copy one or more activity values to a segment field. A MathVariable or a field name can be used to place a single value into a segment field and a collection can be used to place multiple values onto multiple segments.

In addition to field values, CopyToSegmentFields will automatically update the OptionText of combo box and radio button fields.

This rule must be listed in the TransactionBusinessRulePacket.

CopyToSegmentFields Element/Attribute Table

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
<CopyToSegmentFields>	The required opening and closing elements of this business rule.		Enter actual GUID for SEGMENTGUID value.
		SEGMENTGUID	A MathVariable that contains the value of a segment GUID. Identifies the segment by its GUID that will be updated. This attribute cannot be used in combination with the <FromCollection> element.
<Fields>	**Required / Repeatable Element**: Defines a section to contain the fields that will be updated.		
<Field>	**Required / Repeatable Element**: Identifies a field that is updated and its data source. <From> and <FromCollection> elements are mutually exclusive.		
<From>	**Required**: Identifies the MathVariable or activity field as the source value.		A MathVariable or activity field from the transaction to which the rule is attached.
<FromCollection>	**Required**: This element is used to update multiple segments, each with their own individual value. This		A MathVariable of variable type Collection. The keys are segment GUIDs and the values are the updated values.

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
	element cannot be used in combination with the SEGMENTGUID attribute.		
<To>	**Required**: Identifies the field that is updated.		A literal name for a segment field.

XML Example

```
<CopyToSegmentFields SEGMENTGUID="DeferredAnnuityGUID">
      <Fields>
            <Field>
                  <From>Yes</From>
                  <To>DollarCostAveragingProgram</To>
            </Field>
            <Field>
                  <From>Activity:Amount</From>
                  <To>DCAAmount</To>
            </Field>
            <Field>
                  <From>Activity:Frequency</From>
                  <To>DCAFrequency</To>
            </Field>
            <Field>
                  <From>ValidStartDate</From>
                  <To>DCAStartDate</To>
            </Field>
            <Field>
                  <From>ValidStartDate</From>
                  <To>DCANextTransferDate</To>
            </Field>
            <Field>
                  <From>DCATransafersRemaining</From>
                  <To>DCATransfersRemaining</To>
            </Field>
      </Fields>
</CopyToSegmentFields>

<CopyToSegmentFields>
      <Fields>
            <Field>
                  <FromCollection>SegmentModalPremiumAmt</FromCollection>
                  <To>SegmentModalPremium</To>
            </Field>
            <Field>
            <FromCollection>SegmentNextYearsModalPremiumAmt</FromCollec
        tion>
            <To>SegmentNextYearsModalPremium</To>
        </Field>
        <Field>
            <FromCollection>SegmentAnnPremiumAmt</FromCollection>
            <To>SegmentAnnualPremium</To>
        </Field>
        <Field>
            <FromCollection>SegmentNextYearsAnnPremiumAmt</FromCollecti
        on>
            <To>SegmentNextYearsAnnualPremium</To>
        </Field>
    </Fields>
</CopyToSegmentFields>
```

CopyToWithholdingFields

Description

CopyToWithholdingFields loads the withholding fields from the database and updates them based on the values specified in the business rule. The <From> element identifies the math variable or field where a value is being obtained. The <To> element identifies the field that is being updated.
The optional <Test> elements allow conditional logic to be configured, to determine whether the Withholding fields of a pending activity should be updated. If multiple test conditions are configured, they are viewed as AND statements; if all conditions are not met, the update will not be made.
Only one set of withholdings can be updated by a rule.
In addition to field values, CopyToWithholdingFields will automatically update the OptionText of combo box and radio button fields.

CopyToWithholdingFields Element/Attribute Table

Element/Tag	Definition	Attribute	Element/Attribu Descrip
<CopyToWithholdingFields>	Required opening and closing tag.		
		POLICYGUID	**Required attribut level update: PolicyGUID** Identifies the polic withholding will be
		CLIENTGUID	**Required attribut level update: ClientGUID** Identifies the clien withholding will be
<Tests>	**Optional element:** Allows for further definition of the activities that may be updated.		
<Test>	**Required element if <Test> is present.**		A conditional state tests a MathVarial against another M Field, or literal val
<Fields>	**Required element:** Opening tag of fields configuration.		
<Field>	**Repeatable element:** The opening and closing tag that encompasses <From> and <To>.		
<From>	This element is used to specify the math variable or withholding field where the data should be copied from.		**Required elemen MathVariable/Wit** Name of MathVar from the transacti
<To>	**Required element:** This element is used to specify the fields in the Withholding screen where the data will be copied to.		**Required elemen Field** Name of the field screen .

XML Example

```
<CopyToWithholdingFields POLICYGUID="Activity:PolicyGUID">
      <Tests>
            <Test TYPE="Expression">1 = 1</Test>
      </Tests>
      <Fields>
            <Field>
                  <From>Activity:FederalAmount</From>
                  <To>FederalAmount</To>
            </Field>
            <Field>
                  <From>Activity:FederalPercent</From>
                  <To>FederalPercent</To>
            </Field>
            <Field>
                  <From>Activity:StateAmount</From>
                  <To>StateAmount</To>
            </Field>
            <Field>
                  <From>Activity:StatePercent</From>
                  <To>StatePercent</To>
            </Field>
      </Fields>
</CopyToWithholdingFields>

<CopyToWithholdingFields POLICYGUID="Activity:PolicyGUID">
      <Tests>
            <Test TYPE="Expression">1 = 1</Test>
      </Tests>
      <Fields>
            <Field>
                  <From>Activity:FederalAmount</From>
                  <To>FederalAmount</To>
            </Field>
            <Field>
                  <From>Activity:FederalPercent</From>
                  <To>FederalPercent</To>
            </Field>
            <Field>
                  <From>Activity:StateAmount</From>
                  <To>StateAmount</To>
            </Field>
            <Field>
                  <From>Activity:StatePercent</From>
                  <To>StatePercent</To>
            </Field>
      </Fields>
```

```
</CopyToWithholdingFields>
```

CopyToRequirementFields (For Transactions)

Description

This business rule is attached to a transaction to copy activity field values to requirement fields.
In addition to field values, CopyToRequirementFields will automatically update the OptionText of combo box and radio button fields.

CopyToRequirementFields Element/Attribute Table

Element	Definition	Attribute	Element/Attribute Value and Description
<CopyToRequirementFields>	Opening and closing tags of the business rule.		
		REQUIREMENTGUIDS	A MathVariable containing a list of RequirementGUIDs for the requirements that will be changed by this rule.
<Fields>	See Fields section for additional details.		
<Field>			
<FromCollection>	Defines a collection MathVariable that is the source of the data with which to update.		Key is a RequirementGUID that identifies the specific requirement that is updated. Value is the value to which the field will be updated.
<From>	Specifies the requirement MathVariable from which the data should be copied.		The name of the MathVariable from which the data should be copied.
<To>	Defines the target field for the update.		The name of the field on the Requirement screen to which the value should be copied.

XML Sample

```
<CopyToRequirementFields REQUIREMENTGUIDS="RequirementsArray">
      <Fields>
            <Field>
                  <FromCollection>CloseDateCollection</FromCollection>
                  <To>CloseDate</To>
            </Field>
      <Fields>
</CopyToRequirentFields>
```

CopyToProgramFields

Description

The CopyToProgramFields business rule is used to update program fields. The update capability of the rule is restricted so that fixed program fields and program status may not be updated. The rule may be attached only to a program transaction. Updates are limited to dynamic disabled program fields.
In addition to field values, CopyToProgramFields will automatically update the OptionText of combo box and radio button fields.

> As a best practice, the ProgramGUID should be referenced in the configuration so that the GUID can be used as an identifier.

CopyToProgramFields Element/Attribute Table

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
<CopyToProgramFields>		The opening and closing tags of this rule.	
<Tests>		**Optional element** Allows for further definition of the activities that may be updated.	
<Test>		**Required if <Tests> is present.** A conditional statement that tests a MathVariable or Field against another MathVariable, Field, or literal value.	**String:**
<Fields>		**Required element** Common field definition. See Fields Element.	
<Field>		**Repeatable element**; The opening and closing tag that encompasses <From> and <To>	
<From>		**Required element value** Name of the Program source field data is being copied from.	Program: field name is available.
<To>		**Required element value** Name of Program field data is being copied to.	

XML Example

```
<CopyToProgramFields>
      <Tests>
            <Test>IsProgram = 'true'</Test>
      </Tests>
      <Fields>
            <Field>
                  <From>NextBusinessDay</From>
                  <To>NextScheduledDate</To>
            </Field>
      </Fields>
</CopyToProgramFields>
```

XML Schema

```
<CopyToProgramFields>
      <Tests>
            <Test> conditional statement </Test>
      </Tests>
      <Fields>
            <Field>
                  <From></From>
                  <To></To>
            </Field>
      </Fields>
</CopyToProgramFields>
```

AddRoles

Description

The purpose of the AddRoles business rule is to add existing clients in the database to new roles on an existing policy or segment. This rule can only be attached to a policy-level transaction and can only be used to add roles to the policy on which the activity is being processed. For each role in AddRoles, at least ROLECODE and CLIENTGUID/CLIENTGUIDCOLLECTION are required. All other fields, if not specified, will be set to null. This rule must be listed in TransactionBusinessRulePacket.
Multiple roles of the same role code may be created by the rule when the configuration uses the CLIENTGUIDCOLLECTION attribute and <FromCollection> element.
Note: Reversing off an activity that uses AddRoles changes the role code of the added role to a Deleted role code. The role record should not be deleted completely.
AddRoles will automatically set the OptionText of combo box or radio button fields.

AddRoles Element/Attribute Table

<AddRoles>		The opening and closing tag of the AddRoles business rule.
<Role>		**Required/Repeatable Element:** This element is used to define the criteria for the specified roles that are added to the policy. Criteria are defined through various attributes.
	CLIENTGUID, CLIENTGUIDCOLLECTION	**Required attribute:** These attributes are mutually exclusive. If a single Client should be added to a particular type of role, use CLIENTGUID. If multiple Clients should be added to the same role, use CLIENTGUIDCOLLECTION.
	ROLECODE	**Required attribute:** The type of role the client is added to depends on the value of the ROLECODE attribute. The MathVariable must contain a valid RoleCode from AsCode=>AsCodeRole table.
	COMPANYGUID	**Optional attribute:** This attribute is used to specify the CompanyGUID that the newly added role should be saved with in the AsRole table.
	STATECODE	**Optional attribute:** This attribute is used to specify the StateCode from AsCode=>AsCodeState table that the newly added role should be saved with in the AsRole table.
	PERCENTDOLLARCODE	Attribute PERCENTDOLLARCODE will no longer be supported. While it exists in AsRole, we will copy a NULL value for that column to AsRole.

	ROLEPERCENT	**Optional attribute:** Used to specify the percentage of allocation for the newly added role through this business rule. Saves the specified Role Percent in the AsRole table.
	ROLEAMOUNT	**Optional attribute:** Used to specify the amount of allocation for the newly added role through this business rule. Saves the specified Role Amount in the AsRole table.
<Tests>		**Optional Element:** Allows configuration of Test(s) to see if a section of rule should be invoked. All conditions in this section must evaluate to true before the role(s) is/are added.
<Test>		**Required/Repeatable Element:** (Expression) Condition to add a role to the policy.
	TYPE	**Optional attribute** (="Expression") To indicate the type of the condition. Example: <Test TYPE="Expression">SomeMathVariable=27</Test>
<Fields>		**Optional Element:** Used to create a record in AsRoleField table by passing the required information from the transaction.
<Field>		**Required/Repeatable Element:** This tag is used to specify the values with which the newly added roles should be updated in AsRoleField table.
<From> <FromCollection>		**Required Element value:** These elements are mutually exclusive. When a single client is being added as a role, use <From>. The element accepts a field or MathVariable. When there is a potential to add multiple clients as a role, use <FromCollection>. The element accepts a MathVariable of a collection type. The keys are ClientGUIDs.
<To>		**Required Element value:** (RoleScreenFieldName) Name of the RoleScreen field.

XML Sample

```
<AddRoles>
      <Role CLIENTGUIDCOLLECTION="AddClientGUIDCollection"
      ROLECODE="AddRoleCodeCollection" ROLEPERCENT="RolePercentMV"
      ROLEAMOUNT="RoleAmountMV">
```

```
            <Tests>
                  <Test TYPE="Expression">ClientCount=4</Test>
            </Tests>
            <Fields>
                  <Field>
                        <FromCollection>TaxIDCollection</FromCollection>
                        <To>TaxIDField</To>
                  </Field>
            </Fields>
      </Role>
      <Role CLIENTGUID="AddClientGUID" ROLECODE="AddRoleCode"
      PERCENTDOLLARCODE="" ROLEPERCENT="RolePercentMV2"
      COMPANYGUID="CompanyGUIDMV" STATECODE="StateCodeMV">
            <Fields>
                  <Field>
                        <From>TaxIDMV</From>
                        <To>TaxIDField</To>
                  </Field>
            </Fields>
      </Role>
</AddRoles>
```

Calculate

Description

This business rule calculates various segment and policy values. When a segment is added to a policy, the math configured in this business rule takes into account various aspects of the policy and arrives at the values, which may be stored in the database tables. The standard naming convention for the rule is to attach a suffix to the rule name with a common name for the segment to which it is associated, such as CalculateGeneral-BaseCoverage. Calculate business rules are always identified by their <Calculate> parent element.

Note: You can create a plan override of a Calculate business rule for each segment. In the SegmentScreen business rule, associate the calculate button with a specific CalculateGeneral rule that is tailored to a specific segment.

When the Calculate rule is executed, it first processes the <Input> element section. The <Input> element provides for the inclusion of <MathVariables>. Each <MathVariable> must be included inside this element in order for it to be used in any part of the Calculate rule.

After completing the <Input> section, the <Validation> element section is processed. This section allows for data validation. It also processes substitution of the value of math variables or segment fields within validation messages at the time any message is generated.

The final section processed is the <Output> element section. This section provides a method of moving values to various database tables. The <Mappings> sub-element allows <MathVariables> calculated in the <Input> to be copied. In order for each <MathVariable> to be mapped, it must be included in this section regardless of its inclusion in the <Input> section. The Calculate rule will automatically update OptionText for combo box and radio button fields.

By default, each <MathVariable> that is mapped as a field is stored as a segment field in the AsSegmentField database table. In order to store a <MathVariable> as a segment field it must be included as a field in the specific segment's segment name business rule. If the value should not be displayed for the end user in OIPA, then the Hidden attribute of the <Field> element must be used in the segment business rule to hide the field.

If a <MathVariable> needs to be stored in a table other than Segment, such as a policy field in the AsPolicyField table, then it must be specified. The GROUP attribute of <Mapping> allows for the specification of the table the fields should be stored in. In this secenario, GROUP would be set to Policy.

Specific values from the database can be retrieved without writing SQL statements, by using the available defined prefixes and fields for configuration. Please see the Available Prefixes and Fields for Configuration for a listing.

CalculateGeneral Attribute/Element Table

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
<Calculate>		The opening and closing tag of the Calculate	

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
		business rule.	
<Input>		The opening and closing tag that contains the MathVariables for the rule.	
<MathVariables>		Please see MathVariable Elements.	
<MathVariable>			
<MathIF>		Please see MathIF Elements.	
<MathLoop>		Please see MathLoop Elements.	
<CopyBook>		Please see CopyBook Rule.	
<Validation>			
<Expression>		An expression using MathVariables and operators. Allows validation of the values contained in MathVariables. See Validation for full details.	**String**
	TYPE		**ErrorOnTrue** **ErrorOnFalse**
	MESSAGE	The error message to be displayed to the user. **Note:** The value of a Math Variable or a Segment Field can be substituted in the validation message surrounded by $$$. See General Structure and	**String**

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
		Best Practices	
<Output>		Identifies the values that were calculated and will be written to the database.	
<Mappings>		Starting tag that identifies which MathVariable will be mapped.	
<Mapping>		Identifies the MathVariable with the value that will be copied to the database table. The value identifies the Math Variable.	
	OUTPUTNAME	Defines the name of the field that will be updated.	**String**
	TYPE	Always specify a value of "FIELD".	**String**
	GROUP	Identifies the table to which the data will be saved.	**Segment**: The data will be saved to the AsSegment database table. This is the default behavior. **Policy**: The data will be saved to the AsPolicy database table.
	ROLEGUID		
	ROLEGUIDARRAY		
	DEFAULT		
<BenefitSplit>		**Optional element** Defines Benefit Split parameters for calculations. **Note**: The precision of the resulting BenefitSplit units will be set by the Plan's AllocationScreen	

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
		rule using the Policy level settings.	
<Allocations>		**Required element** Selects the allocation from which the benefit split will be built.	
	TYPECODE	Holds a literal or variable indicating the Allocation Type Code to merge.	
	LEVEL	Holds a literal or a Policy/Plan variable indicating the level of the type code to merge.	
<MergeAllocations>		**Optional element** Controls choosing the fixed fund allocations outside of the Allocation Screen.	
	MERGE	Holds a literal or variable (Yes/No) indicating if merge logic is required.	**Yes | No** No is the default value.
<AddAllocations>		Required if MERGE is 'Yes'.	
	TYPECODE	Holds a literal or variable indicating the Allocation Type Code to merge.	
	LEVEL	Holds a literal or a Policy/Plan variable indicating the level of the type code to merge.	
	PERCENT	Holds a literal or variable	

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
		percentage of the allocation to be applied. The precision of the resulting merged allocations will be set by the Plan's AllocationScreen rule using the Policy level settings	
<FinalAllocations>		Controls writing new AsAllocation records with the result of the merged allocations. Final Allocations are saved at the Segment level.	
	TYPECODE	Holds a literal or variable indicating the Allocation Type Code of the FinalAllocation AsAllocation record.	
<Relation>		**Required element** Contains relation keys to link benefit funds to the parent/child allocations.	
<Criteria>		**Required, repeatable element** Identifies the fund field with which to match the Input Math value. The value of the Criteria is a Math Variable from the CalculateGeneral input math, or a literal value, for	

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
		matching the criteria name.	
	NAME	Identifies the Fund Field.	
	DATATYPE	Specifies the data type of the criteria value and Fund Field.	
<FixedBenefitFund>		Required if the segment supports fixed benefit payouts. The element holds a variable containing a fund type code 03 FundGUID to which all fund type code 01 fixed allocation(s) will merge.	
<EffectiveDate>		**Required element** Holds a Math Variable date value used to look up the unit values of the benefit funds.	
<CreateDeferredSplit>		**Optional element** Holds a Math Variable or literal value of Yes or No indicating if type 51 benefit split records are to be created.	**Yes | No** No is the default value.
<VariableBenefit>		**Required element** Specifies a currency or decimal Math Variable representing the calculated variable benefit amount. The currency is	

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
		assumed to be the Plan default. **Note:** Currently only used when 'Solve for Benefit' is selected.	
<FixedBenefit>		**Required element** Specifies a currency or decimal Math Variable representing the calculated fixed benefit amount. The currency is assumed to be the Plan default. **Note:** Currently only used when 'Solve for Benefit' is selected.	
<GeneratePendingRequirements>		**Optional element**	**Yes/No**

XML Examples

Calculate General Example

```
<Calculate>
      <Input>
            <MathVariables>
            <!-- SegmentFields -->
                  <MathVariable VARIABLENAME="True" TYPE="VALUE"
                  DATATYPE="INTEGER">1</MathVariable>
                  <MathVariable VARIABLENAME="False" TYPE="VALUE"
                  DATATYPE="INTEGER">0</MathVariable>
                  <MathVariable VARIABLENAME="GMDBEffectiveDate"
                  TYPE="SEGMENTFIELD"
                  DATATYPE="DATE">StartDate</MathVariable>
                  <!-- Clients On Policy DOB Collection -->
                  <MathVariable VARIABLENAME="AnnuitantExists" TYPE="SQL"
                  DATATYPE="INTEGER" DEFAULT="-999999999">
                        SELECT (CASE WHEN COUNT(*) &gt; 0 THEN 1 ELSE 0 END)
                        FROM AsCode JOIN AsRole ON AsRole.PolicyGUID =
                        '[Policy:PolicyGUID]'
                        AND AsRole.RoleCode = AsCode.CodeValue WHERE
                        AsCode.CodeName = 'AsCodeRole'
                        AND AsRole.RoleCode='27'
                  </MathVariable>
                  <MathVariable VARIABLENAME="AnnuitantDOB" TYPE="SQL"
                  DATATYPE="DATE">
                        SELECT AsClient.DateOfBirth FROM AsCode LEFT JOIN
                        AsRole
                        ON AsRole.RoleCode = AsCode.CodeValue AND
                        AsRole.PolicyGUID = '[Policy:PolicyGUID]'
                        LEFT JOIN AsClient ON AsClient.ClientGUID =
                        AsRole.ClientGUID
                        WHERE AsCode.CodeName = 'AsCodeRole' AND
                        AsCode.CodeValue = '27'</MathVariable>
                  <MathVariable VARIABLENAME="AnnuitantDOBExist"
                  TYPE="FUNCTION" DATATYPE="BOOLEAN">IsEmpty(AnnuitantDOB)
                  </MathVariable>
                  <MathIF IF="AnnuitantDOBExist=false">
                        <MathVariable VARIABLENAME="AnnuitantAgeMV"
                        TYPE="FUNCTION"
                        DATATYPE="INTEGER">ANBAgeOf(AnnuitantDOB,GMDBEffectiv
                        eDate</MathVariable>
                  </MathIF>
                  <MathVariable VARIABLENAME="GMDBMaximumAgeMV"
                  TYPE="PLANFIELD"
                  DATATYPE="INTEGER">GMDBMaximumAge</MathVariable>
            </MathVariables>
      </Input>
      <Validation>
            <Expression TYPE="ErrorOnTrue" MESSAGE="Annuitant Date of Birth
            is blank.">
                  IsEmpty(AnnuitantDOB) And AnnuitantExists &gt; 0
            </Expression>
            <Expression TYPE="ErrorOnTrue" MESSAGE="Annuitant must be of age
            70 or less.">
                  AnnuitantAgeMV &gt; GMDBMaximumAgeMV
            </Expression>
      </Validation>
      <Output>
            <Mappings>
                  <Mapping OUTPUTNAME="AnnuitantAge" TYPE="FIELD"
                  GROUP="Segment">AnnuitantAgeMV</Mapping>
            </Mappings>
      </Output>
      <GeneratePendingRequirements>No</GeneratePendingRequirements>
</Calculate>
```

Calculate Benefit Split Example

```
<Calculate>
```

```
<Input>
      <MathVariables>
            <MathVariable VARIABLENAME="PolicyBandVariable"
            TYPE="POLICYFIELD"
            DATATYPE="TEXT">BandVariable</MathVariable>
            <MathVariable VARIABLENAME="BenefitSplitBand"
            TYPE="SEGMENTFIELD"
            DATATYPE="TEXT">BenefitSplitBand</MathVariable>
            <MathVariable VARIABLENAME="UnitValueDate"
            TYPE="SEGMENTFIELD"
            DATATYPE="DATE">BenefitValuationDate</MathVariable>
            <MathVariable VARIABLENAME="DeferredSplit"
            TYPE="SEGMENTFIELD"
            DATATYPE="TEXT">DeferredSplit</MathVariable>
            <MathVariable VARIABLENAME="VariableBenefitAmount"
            TYPE="SEGMENTFIELD"
            DATATYPE="DECIMAL">VariableBenefitAmount</MathVariable>
            <MathVariable VARIABLENAME="FixedBenefitAmount"
            TYPE="SEGMENTFIELD"
            DATATYPE="DECIMAL">FixedBenefitAmount</MathVariable>
            <MathVariable VARIABLENAME="TotalBenefitAmount"
            TYPE="EXPRESSION" DATATYPE="DECIMAL">VariableBenefitAmount
            + FixedBenefitAmount</MathVariable>
            <MathVariable VARIABLENAME="TotalBenefitAmountCurrency"
            TYPE="FUNCTION"
            DATATYPE="CURRENCY">ToCurrency(TotalBenefitAmount,'USD')</M
            athVariable>
            <MathVariable VARIABLENAME="FixedBenefitFundGUID"
            TYPE="SQL" DATATYPE="TEXT">
                  SELECT AsFund.FundGUID FROM AsPlanFund
                  JOIN AsFund ON AsFund.FundGUID = AsPlanFund.FundGUID
                  AND AsFund.TypeCode = '03' AND AsFund.FundName =
                  'Dynamic Fixed'
                  WHERE PlanGUID = '[Policy:PlanGUID]'
            </MathVariable>
            <MathVariable VARIABLENAME="MergeAllocations"
            TYPE="SEGMENTFIELD"
            DATATYPE="TEXT">MergeAllocations</MathVariable>
            <MathVariable VARIABLENAME="MergePercent"
            TYPE="SEGMENTFIELD"
            DATATYPE="DECIMAL">MergePercent</MathVariable>
            <MathVariable VARIABLENAME="AllocationTypeMV"
            TYPE="SEGMENTFIELD"
            DATATYPE="TEXT">AllocationType</MathVariable>
      </MathVariables>
</Input>
<Output>
      <Mappings>
            <Mapping OUTPUTNAME="SegmentAmount"
            TYPE="FIELD">TotalBenefitAmountCurrency</Mapping>
      </Mappings>
      </Output>
      <BenefitSplit>
            <Allocations TYPECODE="AllocationTypeMV" LEVEL="Policy">
                  <MergeAllocations MERGE="MergeAllocations">
                        <AddAllocations TYPECODE="01" LEVEL="Plan"
                        PERCENT="MergePercent">
                              <FinalAllocations TYPECODE="05"/>
                        </AddAllocations>
                  </MergeAllocations>
            </Allocations>
            <Relation>
                  <Criteria NAME="BandVariable"
                  DATATYPE="TEXT">PolicyBandVariable</Criteria>
                  <Criteria NAME="InvestmentRate"
                  DATATYPE="TEXT">BenefitSplitBand</Criteria>
                  <FixedBenefitFund>FixedBenefitFundGUID</FixedBenefitFund>
            </Relation>
            <EffectiveDate>UnitValueDate</EffectiveDate>
            <CreateDeferredSplit>DeferredSplit</CreateDeferredSplit>
            <VariableBenefit>VariableBenefitAmount</VariableBenefit>
            <FixedBenefit>FixedBenefitAmount</FixedBenefit>
      </BenefitSplit>
</Calculate>
```

XML Schema

```
<Calculate>
      <Input>
            <MathVariables />
      </Input>
      <Output>
            <Mappings />
      </Output>
      <BenefitSplit>
            <Allocations TYPECODE="[CodeValue]" LEVEL="[Policy, Plan]">
                  <MergeAllocations MERGE="">
                        <AddAllocations TYPECODE="[CodeValue]"
                        LEVEL="[Policy, Plan]" PERCENT="">
                              <FinalAllocations TYPECODE="[CodeValue]"/>
                        </AddAllocations>
                  </MergeAllocations>
            </Allocations>
            <Relation>
                  <Criteria NAME="" DATATYPE="" />
                  <FixedBenefitFund />
            </Relation>
            <EffectiveDate />
            <CreateDeferredSplit />
            <VariableBenefit />
            <FixedBenefit />
      </BenefitSplit>
      <GeneratePendingRequirements>[Yes|No]</GeneratePendingRequirements>
</Calculate>
```

MathVariable Element

Within the `<Math>` element, calculations can be evaluated using MathVariables. MathVariables are variables that are used in the Math section of business rule configuration and use the `<MathVariable>` tag. A MathVariable name is assigned as the value of the attribute `VARIABLENAME`. After you name the MathVariable, you will identify what `type` of MathVariable you want to create. There are various types of MathVariables available to configure processing. MathVariables can be used to retrieve a field value, create arrays, structure if-then structures, perform calculations with operators or call functions. Each of the math types has its own set of attributes, elements and possible values, which are listed below. All MathVariables **must** have a data type declared using the `DATATYPE` attribute.

The `<MathVariable>` tag can use a `LOG="Yes"` attribute, which will log results to AsActivityMath. When using logging, remember that the system performs all logging at the end of transaction processing. Therefore, only the ending value of a MathVariable is logged. Although the configuration allows the LOG attribute on any use of a variable in the configuration, the intermediate values are not logged, even if LOG="Yes" occurs multiple times for the same variable. In such a case, to avoid confusion it is best to specify the LOG attribute only for the last use of the variable. Another method is to create a "Log Section" at the bottom of the configuration that specifies all math variables to be logged.

There are also `<MathLoop>` and `<MathIF>` elements, which are available to perform loops and if conditions, respectively. Both must have a data type declared as well.

Available DATATYPE Options

- `ACTIVITY`
- `BIGTEXT`
- `BOOLEAN`
- `CURRENCY`
- `DATE`
- `DECIMAL`
- `INTEGER`
- `TEXT`
- `MAP`
- `OBJECT`
- `XML (Works with TYPE="VALUE" and "FIELD")`

`DATATYPE` of either decimal or integer may not be blank, and a `DEFAULT` attribute must be defined.

If you will be dealing with integers that are larger than 2147483648 in math configuration then the DECIMAL data type must be used.

MathVariables may be used to pass values into spawn configuration. Math statements are processed sequentially from top to bottom in the configuration.

XML Example

```
<MathVariables>
      <!-- Calculate Days from Last Bank Draft -->
      <MathVariable VARIABLENAME="LastActiveBankDraftEffectiveDate"
            TYPE="VALUE" DATATYPE="DATE"></MathVariable>
```

```
<MathVariable VARIABLENAME="BankDraftEndDate" TYPE="FIELD"
      DATATYPE="DATE">Activity:EffectiveDate</MathVariable>
<MathVariable VARIABLENAME="LastActiveBankDraftEffectiveDate"
TYPE="SQL" DATATYPE="DATE">
      <SqlServer>SELECT TOP 1 AsActivity.EffectiveDate FROM AsActivity
            JOIN AsTransaction ON AsTransaction.TransactionGUID =
            AsActivity.TransactionGUID
            AND AsTransaction.TransactionName = 'BankDraft'
            WHERE AsActivity.PolicyGUID = '[Policy:PolicyGUID]'
            AND AsActivity.StatusCode IN ('01','13','14')
            AND AsActivity.TypeCode IN ('01','04')
            ORDER BY AsActivity.EffectiveDate DESC
      </SqlServer>
      <Oracle>SELECT EffectiveDate FROM (SELECT
            AsActivity.EffectiveDate FROM AsActivity
            JOIN AsTransaction ON AsTransaction.TransactionGUID =
            AsActivity.TransactionGUID
            AND AsTransaction.TransactionName = 'BankDraft'
            WHERE AsActivity.PolicyGUID = '[Policy:PolicyGUID]'
            AND AsActivity.StatusCode IN ('01','13','14')
            AND AsActivity.TypeCode IN ('01','04')
            ORDER BY AsActivity.EffectiveDate DESC) WHERE ROWNUM = 1
      </Oracle>
      <DB2>SELECT AsActivity.EffectiveDate FROM AsActivity
            JOIN AsTransaction ON AsTransaction.TransactionGUID =
            AsActivity.TransactionGUID
            AND AsTransaction.TransactionName = 'BankDraft'
            WHERE AsActivity.PolicyGUID = '[Policy:PolicyGUID]'
            AND AsActivity.StatusCode IN ( '01' , '13' , '14' )
            AND AsActivity.TypeCode IN ( '01' , '04' )
            ORDER BY AsActivity.EffectiveDate DESC FETCH FIRST 1 ROWS
            ONLY
      </DB2>
</MathVariable>
<MathVariable VARIABLENAME="TestLastActiveBankDraft" TYPE="FUNCTION"
      DATATYPE="BOOLEAN">IsEmpty(LastActiveBankDraftEffectiveDate)
</MathVariable>
<MathIF IF="TestLastActiveBankDraft = false">
      <MathVariable VARIABLENAME="DaysFromLastBankDraft"
      TYPE="FUNCTION"
      DATATYPE="INTEGER">DaysDiffOf(LastActiveBankDraftEffectiveDate,Ba
      nkDraftEndDate)</MathVariable>
</MathIF>
<MathVariable VARIABLENAME="CommentToSpawn" TYPE="FIELD"
DATATYPE="BIGTEXT">Activity:AnnualStatementComment</MathVariable>
<MathVariable VARIABLENAME="XmlValueMV" TYPE="VALUE" DATATYPE="XML"/>
<MathVariable VARIABLENAME="XmlField1MV" TYPE="FIELD"
DATATYPE="XML">Activity:XMLData</MathVariable>
```

```
</MathVariables>
```

MULTIFIELD

Description

The MULTIFIELD math variable is used to access multifield values (using INDEX as an attribute) in Screen Math and Math sections.

The INDEX attribute can be any integer value ranging from 0-99. When the INDEX attribute is equal to 0, the first instance of a multifield will be retrieved. Each additional instance of a multifield will be accessed by incrementing the INDEX value in a subsequent math variable. A math variable that is configured with an INDEX greater than the number of instances of the multifield selected on the activity will result in the value of the math variable being equal to null.

For math variables that are data type STRING or DATE, null is a valid value and no error will be returned. For math variables that are data type DECIMAL or INTEGER null is not a valid value and a system error will be returned, unless a default value is specified using configuration.

A math variable that is configured with the TYPE equal to MULTIFIELD, but has no INDEX attribute specified will result in a code-generated system error.

Note: This math variable is currently only supported in transaction configuration.

MULTIFIELD Element/Attribute Table

TYPE=MULTIFIELD				
Element	**Attributes**	**Attribute Value**	**Element Value**	**Definition**
<MathVariable>	TYPE	MULTIFIELD		**Required attribute:** The multifield value is used to access multifield values in math. The MathVariable element value is the name of the MultiField Field being retrieved.
	INDEX	Integer value	The integer reference to the instance of multifield to access.	**Required attribute:** This attribute is used to specify the

TYPE=MULTIFIELD				
Element	**Attributes**	**Attribute Value**	**Element Value**	**Definition**
				particular instance of the multifield to access.

XML Example

```
<MathVariable VARIABLENAME="CorrectionDateMV" TYPE="MULTIFIELD" INDEX="1"
DATATYPE="DATE">CorrectionDate</MathVariable>
```

MultiFields Element

Description

A set of fields that can be repeated is called MultiField. MultiFields enable and allow Screen and Transaction configuration with multiple sets of dynamic field values. To use MultiField in a business rule or transaction, it has to be first defined using a MultiField business rule. It can then be used by screen displaying business rules with the <MultiField> element. For detailed information about configuring MultiFields, refer to "MultiField Configuration" in the Rules Palette Help.
Note: In any one Screen or Transaction, only one MultiField BR can be used through a MultiFields tag. In case multiple MultiFields are required, the same can be defined inside a single MultiField BR. If more than one MultiFields tag is used, only the first one will take effect.

MultiFields Element Table

Element/Tag	Attribute	Definition	Element/Attribute Value
<MultiFields>		**Required:** The opening and closing tag of the MultiFields element. This element is used to turn on/off the multifields section. The statement can occur in any part of the Transaction/Screen XML.	**Required element value: Yes**
	RULE	A literal value referencing the exact rule name of a MultiFields business rule (AsBusinessRules.RuleName).	

XML Sample

```
<MultiFields RULE="IdentifierFields">Yes</MultiFields>
```

XML Schema

```
<MultiFields RULE="[business rule name]">Yes</MultiFields>
```

MultiFields Business Rule

Description

The MultiFields business rule defines a multifield rule configuration. This configuration defines a set of one or more fields and a number range. The number range allows the user to select the number of fields to display on the screen. Each configured multifields business rule can be referenced by its rule name in MultiFields elements in screen and transaction configurations that use multifields.
For class, segment name, and client relationship only, the<Actions> and <Events> elements are valid within the MultiFields rule. See Actions/Events.
The SQL statements in the <Start> and <End> elements of the <MultiField> element may refer to the value of fields in a transaction, with the field name enclosed in brackets [ ], as in the example:
<Start>SELECT MIN(FieldIndex)+1 FROM AsAgreementMultiValueField
Where AgreementGUID = '[SelectedPerformanceContact]'</Start>
The <Name> value should be unique across a given MultiField BR. In case more than one <Field> inside the same <MultiField> section or across multiple <MultiField> sections is defined with the same <Name>, the display and behavior of the system under Action/Events will not be able to differentiate between those Fields.

MultiFields Business Rule Element/Attribute Table

Element	Definition	Attribute	Element/Attribute Values
<MultiFields>	**Required:** The opening and closing tag of the MultiFields rule.		
<MultiField>	**Required:**		
<Name>	Used to define the name of the multifield for Events. This name can be different than Title but must be unique.		
<Title>	**Optional element:** Defines the title of the MultiFields pop-up display window.		If this element is omitted in the business rule, the MultiFields window title will be "MultiFields".
<ComboDisplay>	**Optional element:** Defines the display name of the Number of Fields combo box where the user can select the number of multifields to display.		
<Start>	**Required:** This specifies the starting value of the Number of Fields combo box that		**Integer**

Element	Definition	Attribute	Element/Attribute Values
	allows the user to select how many multifields to display.		
<End>	**Required:** This specifies the ending value of the Number of Fields combo box.		**Integer**
<Fields>	**Optional element:** If no fields are specified, the Number of Fields combo box in the Multifield section will be empty. See Fields Element.		
<Events>			
<Event>	Optional conditional logic. All existing Elements and Attributes for Actions/Events are available for use within MultiField tags.		**MultiValueFieldIndex=**(starting from 0) This is a built in variable available when the Event processing is in multifields scope.
<Actions>	All existing Elements and Attributes are available for use within MultiField tags. See Actions/Events.		
<ActionSet>	All existing Elements and Attributes are available for use within MultiField tags		
<Condition>	Optional conditional logic. All existing Elements and Attributes are available for use within MultiField tags	IF	Any expression resulting in a true or false value. **MultiValueFieldIndex=**(starting from 0) This is a built in variable available when the Event processing is in multifields scope.
<Action>	All existing Elements and Attributes are available for use within MultiField tags	ACTIONTYPE	String **$$$MultiValueFieldIndex$$$** This can be used within string. It is a built in variable available when the Event processing is in multifields scope.

XML Sample

```
<MultiFields>
   <MultiField>
      <Name>TextTestMF</Name>
      <Title>Text Test</Title>
      <ComboDisplay>Text Test</ComboDisplay>
      <Start>3</Start>
      <End>5</End>
      <Fields>
         <Field>
            <Name>TextTest1</Name>
            <Display>Text Test 1</Display>
            <DataType>Text</DataType>
         </Field>
         <Field>
            <Name>TextTest2</Name>
            <Display>Text Test 2</Display>
            <DataType>Text</DataType>
         </Field>
         <Field>
            <Name>TextTest3</Name>
            <Display>Text Test 3</Display>
            <DataType>Text</DataType>
         </Field>
         <Field>
            <Name>TextTest4</Name>
            <Display>Text Test 4</Display>
            <DataType>Text</DataType>
            <Default>Test4</Default>
         </Field>
      </Fields>
      <Events>
         <Event TYPE="ONLOAD">
            <ActionSet ID="TestTextAction"/>
         </Event>
         <Event TYPE="ONCHANGE" FIELD="TextTest1">
            <ActionSet ID="TestTextAction"/>
         </Event>
         <Event TYPE="ONSUBMIT">
            <ActionSet ID="TestTextAction"/>
         </Event>
      </Events>
      <Actions>
         <ActionSet ID="TestTextAction">
            <Condition IF="IsEmpty( TextTest1 ) And multiValueFieldIndex=1">
               <Action ACTIONTYPE="ERROR">TextTest1
               $$$multiValueFieldIndex$$$ cannot be empty.
```

```
                </Action>
            </Condition>
            <Condition IF="Not IsEmpty(TextTest1) And TextTest1 = 'Test'">
                <Action ACTIONTYPE="ERROR">TextTest1
                $$$multiValueFieldIndex$$$ cannot be Test.
                </Action>
            </Condition>
                <Action ACTIONTYPE="ASSIGN"
                FIELD="TextTest4">'Text4'</Action>
                <Action ACTIONTYPE="HIDE" FIELD="TextTest3"/>
                <Action ACTIONTYPE="DISABLE" FIELD="TextTest4"/>
         </ActionSet>
      </Actions>
   </MultiField>
</MultiFields>
```

ActionEvents

Description

ActionEvents provide the ability to perform actions on a field as an event occurs on the screen. Configuration establishes the important events and links them to action sets via ID attributes. When the system detects one of these events, it automatically executes the action set assigned to the event. ActionEvents can be used to:

- create custom error and warning messages that appear to the end user after the system validates what was entered.
- change fields from enabled to disabled (and vice versa) when an event occurs.
- change fields from hidden to displayed (and vice versa) when an event occurs.
- change a field's data according to the value in a trigger field.

ActionEvents Elements/Attributes Table

Element/Tag	Definition	Attributes	Element/Attribute Value and Description
<Events>	The opening and closing tag for the element.		
<Event>	**Repeatable:** Identifies the event that is being defined.	TYPE	ONLOAD ONCHANGE ONSUBMIT CALLEDEVENT: this will be used to call an EVENT which is configured outside the transaction in a business rule which is associated with the transaction. Currently this feature is available in the TransactionAllocationScreen BR and MultiFields BR. Please refer to the prototype configuration example available in Rules Palette guide for details of how and when this can be used.
		FIELD	Any field name in the <Fields> section. Used with ONCHANGE to identify the trigger field, which when it changes, causes the actions to process.
		ID	The unique ID for the event being invoked by the transaction. The ID value should match the ID in the CALLEXTERNALEVENT definition in the transaction XML.
<ScreenMath>	Opening and closing tags for Event Math section.		
<Math>	Call the <Math> that is defined in the	ID	Any ID name given in the Math section under <ScreenMath>.

Element/Tag	Definition	Attributes	Element/Attribute Value and Description
	<ScreenMath> section to be run whose name matches an ID.		
<MathVariables>			
<MathVariable>		TYPE	SUSPENSEFIELD VALUE
<Actions>			
<QuerySet>	Defines QuerySet.	ID	Any name (no spaces).
<Condition>	Optional conditional logic.	IF	Any expression resulting in a true or false value. When the expression results in a true value, the system passes execution to the first statement contained by the opening and closing tags of the parent element. Statement execution continues serially to the next statements until the closing parent tag, an Else element or an ElseIf element is reached. When the expression results in a false value, the system passes execution control to an Else or ElseIf element or closing parent tag, whichever is closest. Functions and direct date comparisons are available for use in the expression. Screen math variables and fields are available for use in the expression.
		VALUE	String.
<Action>	Defines whether to fill combo box with SQL query, or use fixed options.		**String:** SQL query string.
		ACTIONTYPE	SQL QUERY The following action types are the only ones available to Events in multifields: HIDE SHOW DISABLE DISABLEALL: disables all fields, both dynamic and fixed, including multifields within a configuration section. This can only be used with the ONLOAD event type. If used with ONSUBMIT or ONCHANGE it will be ignored. ENABLE ASSIGN CALLEXTERNALEVENT: this will be used exclusively to invoke an event

Element/Tag	Definition	Attributes	Element/Attribute Value and Description
			which is configured outside the transaction in a business rule which is associated with the transaction. Currently this feature is available in the TransactionAllocationScreen BR and MultiFields BR. Please refer to the prototype configuration example available in Rules Palette guide for details of how and when this can be used.
		FIELD	The field name where the action is to occur. When used with MULTIFIELD this indicates which field within the multifield will be impacted. If INDEX is not indicated all instances of that field within the multifield will be updated.
		INDEX	Used with MULTIFIELD. Indicates the position/row of the field the Action is to occur on (starts at 0). If INDEX is used only the field for that index will be impacted.
		MULTIFIELD	The multifield name where the action is to occur. If only the MultiField is referenced then the event occurs on the entire section.
		ID	To be used in conjunction with CALLEXTERNALEVENT. The ID value should match the ID in the CALLEDEVENT definition in the MultiFields BR/TransactionAllocation BR.
<ElseIf>	ElseIf is executed when prior conditions in IF and ElseIf elements are not true. Use this element when there are further conditions to express in the condition structure. IF attribute must be included. Multiple ElseIf elements are possible within a condition structure.	IF	Any expression resulting in a true or false value. When the expression results in a true value, the system passes execution to the first statement contained by the opening and closing tags of the parent element. Statement execution continues serially to the next statements until the closing parent tag, an Else element or an ElseIf element is reached. When the expression results in a false value, the system passes execution control to an Else or ElseIf element or closing parent tag, whichever is closest. Functions and direct date comparisons are available for use in the expression. Screen math variables and fields are available for use in the expression.

Element/Tag	Definition	Attributes	Element/Attribute Value and Description
		VALUE	String.
<Else>	Else is executed when prior conditions in IF and ElseIf elements are not true. Use this element when there are no further conditions to express in the condition structure. Only one Else is possible within a condition structure.		Execution is passed to the first Action statement contained within the opening and closing <Else> element. Each statement is serially executed until the closing tag is reached.
<ActionSet>	Defines ActionSet	ID	Any name (no spaces).
<Condition>	Same as defined above.		
<Action>	Defines the type of action that is to occur.		**Optional element value: String:** Message to be displayed if ACTIONTYPE is ERROR or WARNING, and the IF condition is met.
		FIELD	The field name where the action is to occur. This attribute is required when action type is equal to assign. It is also available when action type is equal to error or warning.
		ACTIONTYPE	ERROR Provides a message to the user. The user cannot proceed until the all errors are fixed. All errors will display in one section of the screen (if more than one occurs). When FIELD is used, the error will appear in the Validation section at the top of the screen. **Note:** The value of a Math Variable or a Segment Field can be substituted in the error message surrounded by $$$. See General Structure and Best Practices
			WARNING Provides a warning message to the user. This is a warning to the user and the user may proceed even with the error messages. All errors will display in one section of the screen (if more than one occurs). When FIELD is used, the warning will

Element/Tag	Definition	Attributes	Element/Attribute Value and Description
			appear in the Validation section at the top of the screen. **Note:** The value of a Math Variable or a Segment Field can be substituted in the warning message surrounded by $$$. See General Structure and Best Practices
			SHOW Will make hidden field visible.
			HIDE Will hide a visible field.
			ENABLE Allows data entry in an editable field.
			DISABLE Prohibits data entry in an editable field, and changes the background color.
			DISABLEALL Prohibits data entry in all editable fields and multifields within a configuration section. **Note:** The DISABLEALL option is available for the following screens: • ClientScreen • AddressScreen
			READONLY Prohibits data entry in a field and does not change background color.
			ASSIGN Allows a value to be set to the field defined in the FIELD attribute. This action does not trigger an OnChange event on the receiving field.
<ElseIf>	Same as defined above.		
<Else>	Same as defined above.		

XML Example Without Multifield Events

```
<Events>
   <Event TYPE="ONLOAD">
         <QuerySet ID="OnLoadQualType" FIELD="FundingMethod"></QuerySet>
   </Event>
   <Event TYPE="ONCHANGE" FIELD="FundingMethod">
      <ActionSet ID="OnChangeFundingMethod"></ActionSet>
   </Event>
   <Event TYPE="ONSUBMIT">
      <ActionSet ID="FinalValidation"></ActionSet>
```

```
    </Event>
</Events>
<ScreenMath>
    <Math ID="GlobalScreenMath" GLOBAL="Yes">
        <MathVariables>
            <MathVariable VARIABLENAME="NA" TYPE="VALUE"
            DATATYPE="TEXT">00</MathVariable>
            <MathVariable VARIABLENAME="NonQualified" TYPE="VALUE"
            DATATYPE="TEXT">13</MathVariable>
            <MathVariable VARIABLENAME="Replacement" TYPE="VALUE"
            DATATYPE="TEXT">98</MathVariable>
        </MathVariables>
    </Math>
</ScreenMath>
<Actions>
    <QuerySet ID="OnLoadQualType">
        <Condition IF="QualType=GlobalScreenMath:NonQualified">
            <Action ACTIONTYPE="SQLQUERY">SELECT AsCode.CodeValue,
            AsCode.ShortDescription FROM AsCode WHERE AsCode.CodeName =
            'AsCodeMoneyType' AND AsCode.CodeValue IN ('00','01','50') ORDER BY
            AsCode.CodeValue</Action>
            <Else>
            <Action ACTIONTYPE="SQLQUERY">SELECT AsCode.CodeValue,
            AsCode.ShortDescription FROM AsCode WHERE AsCode.CodeName =
            'AsCodeMoneyType' AND AsCode.CodeValue IN ('00','01','98') ORDER BY
            AsCode.CodeValue</Action>
            </Else>
        </Condition>
    </QuerySet>
    <ActionSet ID="OnChangeFundingMethod">
        <Condition IF="FundingMethod=GlobalScreenMath:Replacement">
            <Action ACTIONTYPE="ENABLE" FIELD="ReplacementType"></Action>
            <Else>
            <Action ACTIONTYPE="DISABLE" FIELD="ReplacementType"></Action>
            <Action ACTIONTYPE="ASSIGN"
            FIELD="ReplacementType">GlobalScreenMath:NA</Action>
            </Else>
        </Condition>
    </ActionSet>
    <ActionSet ID="FinalValidation">
        <Condition IF="IsEmpty(GrossAmount)">
            <Action ACTIONTYPE="ERROR">Gross Amount is a required
            field.</Action>
            <ElseIf IF="GrossAmount &lt;= 0">
            <Action ACTIONTYPE="ERROR">Gross Amount must be greater than
            $0.00.</Action>
            </ElseIf>
        </Condition>
        <Condition IF="IsEmpty(EffectiveDate)">
```

```
          <Action ACTIONTYPE="ERROR">Effective Date is a required
          field.</Action>
          <ElseIf IF="EffectiveDate &gt; SystemDate">
          <Action ACTIONTYPE="ERROR">Effective Date cannot be in the
          future.</Action>
          </ElseIf>
       </Condition>
       <Condition IF="ReplacementType=GlobalScreenMath:NA And
       FundingMethod=GlobalScreenMath:Replacement">
          <Action ACTIONTYPE="ERROR">A Replacement Type of
          $$$ReplacementType$$$ must be chosen.</Action>
       </Condition>
    </ActionSet>
</Actions>
```

XML Example With MultiField Events

```
<Transaction>
    <EffectiveDate STATUS="Enabled" TYPE="SYSTEM"></EffectiveDate>
    <MultiFields RULE="MultiField-SGMultiFieldPrototype">Yes</MultiFields>
       <Fields>
          <Field>
             <Name>TestField</Name>
             <Display>Non-MultiValue Field Test</Display>
             <DataType>Text</DataType>
          </Field>
          <Field>
             <Name>TransactionField</Name>
             <Display>TransactionField</Display>
             <DataType>Text</DataType>
          </Field>
       </Fields>
       <ScreenMath>
          <Math ID="GlobalScreenMath" GLOBAL="Yes">
             <MathVariables>
                <MathVariable VARIABLENAME="ValueFromScreenMath" TYPE="VALUE"
                DATATYPE="TEXT">ScreenMath</MathVariable>
             </MathVariables>
          </Math>
       </ScreenMath>
       <Events>
          <Event TYPE="ONCHANGE" FIELD="TestField">
             <ActionSet ID="TestChangeExternalAction"></ActionSet>
          </Event>
          <Event TYPE="ONCHANGE" FIELD="TestField">
             <ActionSet ID="TestChangeInternalAction"></ActionSet>
          </Event>
          <Event TYPE="ONLOAD">
```

```
            <ActionSet ID="OnLoadExternalAction"></ActionSet>
         </Event>
         <Event TYPE="ONCHANGE" FIELD="EffectiveDate">
             <ActionSet ID="ActionOnChange"/>
          </Event>

      </Events>
      <Actions>
         <ActionSet ID="OnLoadExternalAction">
            <Action ACTIONTYPE="ASSIGN"
            FIELD="TransactionField">'OnLoadValue'</Action>
            <Action ACTIONTYPE="ASSIGN" MULTIFIELD="TextTestMF"
            FIELD="TextTest2" INDEX="1">'ValueFromTransaction'</Action>
         </ActionSet>
         <ActionSet ID="TestChangeExternalAction">
            <Condition IF="Not IsEmpty(TestField) And
            TestField='HideCombo'">
               <Action ACTIONTYPE="HIDE" MULTIFIELD="ComboTestMF"></Action>
            </Condition>
            <Condition IF="Not IsEmpty(TestField) And
            TestField='ShowCombo'">
               <Action ACTIONTYPE="SHOW" MULTIFIELD="ComboTestMF"></Action>
            </Condition>
         </ActionSet>
         <ActionSet ID="TestChangeInternalAction">
            <Condition IF="Not IsEmpty(TestField) And
            TestField='HideTransactionField'">
               <Action ACTIONTYPE="HIDE" FIELD="TransactionField"></Action>
            </Condition>
         </ActionSet>
         <ActionSet ID="ActionOnChange">
              <!--The MultiFields BR - MultiField-SGMultiFieldPrototype -
         will contain an Event of type CALLEDEVENT with ID = "MFEvent" -->
             <Action ACTIONTYPE="CALLEXTERNALEVENT" ID="MFEvent"></Action>
          </ActionSet>

      </Actions>
      <Math>
         <MathVariables>
            <MathVariable VARIABLENAME="One" TYPE="VALUE"
            DATATYPE="TEXT">One</MathVariable>
         </MathVariables>
      </Math>
</Transaction>
```

XML Example of a MultiFields BR to Explain CALLEDEVENT Configuration

```
<MultiFields>
         <MultiField>
                 <Name>AddressMultifield</Name>
                 <Title>Address Information</Title>
                 <Start>1</Start>
                 <End>3</End>
                 <Fields>
                         <Field>
                                 <Name>AddressGUID</Name>
                                 <Display>AddressGUID</Display>
                                 <DataType>Text</DataType>
                         </Field>
                 </Fields>
                 <Events>
                         <Event TYPE="CALLEDEVENT" ID="MFEvent">
                                 <ScreenMath ID="ScreenMath"/>
                                 <ActionSet ID="MFActionOnChange"/>
                         </Event>
                 </Events>
                 <ScreenMath>
                         <Math ID="ScreenMath" GLOBAL="No">
                                 <MathVariables>
                                         <MathVariable
VARIABLENAME="RoleGUIDMV" TYPE="MULTIFIELD" INDEX="MultiValueFieldIndex"
DATATYPE="TEXT">RoleGUID</MathVariable>
                                         <MathVariable
VARIABLENAME="AddressGUIDMV" TYPE="SQL" DATATYPE="TEXT">SELECT * FROM
TABLE(PKG_INQUIRY_SCREEN.fncEftRoles('[RoleGUIDMV]',
'[EffectiveDate]'))</MathVariable>
                                 </MathVariables>
                         </Math>
                 </ScreenMath>
                 <Actions>
                         <ActionSet ID="MFActionOnChange">
                                 <Action ACTIONTYPE="ASSIGN"
MULTIFIELD="AddressGUID">AddressGUIDMV</Action>
                         </ActionSet>
                 </Actions>
         </MultiField>
 </MultiFields>
```

ValidateExpressions

Description

The Validation Expressions section in a screen rule or a transaction supports the ability to configure "hard" edits or pop-up error messages and to optionally suggest fixes.
Note: This business rule is overridden at the transaction level.

ValidateExpressions Element/Attribute Table

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
<ValidateExpressions>	The opening and closing tag for the business rule.		
<Expression>	**Required /Repeatable Element**; This tag specifies the error messages to be displayed on the screen when a specific Condition is evaluated to be True/ False.		**Required Element Value**: **Expression**: Specifies the condition that needs to be evaluated upon trying to process the transaction.
		TYPE	**Optional Attribute**: **ErrorOnTrue/ErrorOnFalse**: Tells the math engine to error if the expression is evaluated to True or False. **ErrorOnTrue**: If the Expression is true then displays the error message. **ErrorOnFalse**: If the Expression is false then displays the error message. Note: ErrorOnFalse is a default value if TYPE="ErrorOnTrue" is not specified.
		FIX	**Optional Attribute**: **ErrorFixTipMessage** To provide instructions to the users to guide them in correcting the issue that caused the error message. The actual error fix tips should be specified as the value. Example:

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
			== === ERRORNUMBER ErrorFixTip == === LH001 - Change The Members Age or Override U001 - (null) == === If the LH001 Error Number is specified in "ERRORNUMBER" attribute then in the "FIX" attribute the "ErrorFixTip" should be configured.
		OVERRIDABLE	**Optional Attribute**: **Yes**: Error messages will be able to be overridden by checking the **Override?** checkbox. **Auto**: Defines the ability to automatically override the error. **No**: The **Override?** checkbox will be disabled, preventing errors from being overridden. Note: If this attribute is omitted from the configuration, the **Override?** checkbox will be enabled, allowing errors to be overridden.
		ERRORNUMBER	**Optional Attribute**: **ErrorNumber**: Indicates the field value to define error message activity. Code values for error message from the AsErrorCatalog database table. Example: LH001, U001, V000, V001, V012, V020 Note: If the OVERRIDABLE attribute is set to "Yes," the error numbers entered as values of this attribute will display in the Overridable Errors section of the Transaction Security pane. The security group's ability to override each error can be configured on this pane.
		MESSAGE	**Required Attribute**: **ErrorMessage**: This attribute is used to specify the error message that should be displayed upon evaluating the condition. Note: The value of a Math Variable or a Segment Field can be substituted in the error message surrounded by $$$. See General Structure and Best Practices
		WARNING	**Optional Attribute**: **Yes:** The expression is defined as a warning. **No:** The expression is defined as an error. This is the default value. Note: If this attribute is present in the configuration, but the VerificationScreen rule has not been configured for the transaction, then the attribute will be ignored. If the attribute is absent from the configuration, then the expression will be defined as

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
			an error, by default.

XML Example

```
          <ValidateExpressions>
             <Expression TYPE="ErrorOnTrue" OVERRIDABLE="Yes, Super"
             MESSAGE="Complex
               policy change event must process within current modal
             period." ERRORNUMBER="UO01"
               FIX="To process on this date, at least one Premium
             transaction must
               be reversed"
             >DaysDifferenceBetweenPreviousPaidToDateAndEffectiveDate
             &lt; Zero
             </Expression>
             <Expression TYPE="ErrorOnTrue" OVERRIDABLE="Yes"
              MESSAGE="Owner’s Age of $$$OwnerAgeMV$$$ must be less than
             or equal to $$$MaxAgeNY$$$">
              OwnerAgeMV &gt; MaxAgeNY And IssueStateCodeMV = '32'
             </Expression>
<Expression TYPE="ErrorOnTrue" WARNING="Yes" ERRORNUMBER="W035"
FIX="Change amount to be equal to or greater than minimum"
MESSAGE="Amount is less than minimum premium">
 MoneyIn &lt; MinimumPremiumAmount
</Expression>
          </ValidateExpressions>
```

PolicySearchScreen

Definition

This business rule is used to configure the PolicySearchScreen. It defines the fields that are used to store the results of a search.

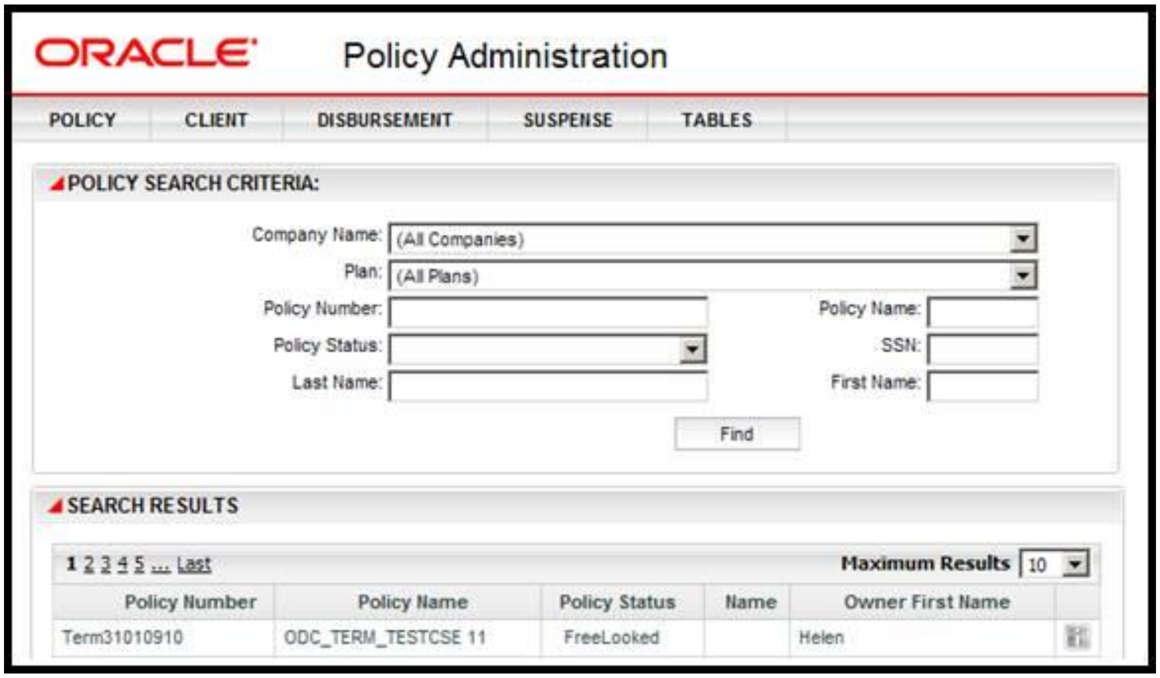
Both the search criteria (Fields & FixedFields) and results (Columns) will accept mask attributes in the DataType elements. Refer to the Fields and Tables sections to see mask attribute configuration.

PolicySearchScreen Element/Attribute Table

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
<PolicySearchScreen>		The opening and closing tags of the PolicySearchScreen business rule.	
<AutoSelect>		Allows configuration of automatic navigation.	**Yes**: If a search for a policy in OIPA returns a single result, then the Policy screen—or the Policy Overview screen, if configured—for that policy will automatically load. **No**: Even if a search for a policy returns a single result, the Policy screen/Policy Overview screen will not automatically load, and the user will have to manually select the policy from the policy search results.
<Search>		**Required and Repeatable Element**: Indicates and defines search criteria and the fields to be included as part of the Policy Search screen activity.	
<FixedFields>		Changes the labels on the "above the line" fixed fields.	
<Fields>		Dynamically changes labels on the "below the line" fields. See Fields Elements.	
<Results>		**Required:** Defines the results.	This element provides the return of the search and the data in the Results override the

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
			fixed fields.
	INITIALRESULTS		This attribute specifies the initial results that should display on the screen when it initially loads. **User**: Indicates Policy search results tied to the client login should be displayed. **None**: Indicates no Policy search results should be displayed. This is the default behavior.
<Table>		**Required:** The element that defines the screen as a table format and controls the display of results, formats results in a table. See Table Element.	

PolicySearchScreen Image



Company Name: (All Companies)
Plan: (All Plans)
Policy Number:
Policy Name:
Policy Status:
SSN:
Last Name:
First Name:

Maximum Results 10

Policy Number	Policy Name	Policy Status	Name	Owner First Name
Term31010910	ODC_TERM_TESTCSE 11	FreeLooked		Helen

XMLExample

```
<PolicySearchScreen>
      <AutoSelect>Yes</AutoSelect>
```

```
<Search>
      <FixedFields>
            <Field>
                  <Name>Company</Name>
                  <Display>Company</Display>
            </Field>
            <Field>
                  <Name>Plan</Name>
                  <Display>Plan</Display>
            </Field>
      </FixedFields>
      <Fields>
            <Field>
                  <Name>PolicyNumber</Name>
                  <Display>Policy Number</Display>
                  <DataType>Text</DataType>
                  <Group>Policy</Group>
                  <InputFocus>Yes</InputFocus>
            </Field>
            <Field>
                  <Name>LastName</Name>
                  <Display>Annuitant Last Name</Display>
                  <DataType>Text</DataType>
                  <Group ROLECODE="27">Client</Group>
            </Field>
            <Field>
                  <Name>FirstName</Name>
                  <Display>Annuitant First Name</Display>
                  <Group ROLECODE="27">Client</Group>
            </Field>
            <Field>
                  <Name>AppSignDate</Name>
                  <Display>Signed Date</Display>
                  <DataType>Date</DataType>
                  <Group>PolicyField</Group>
            </Field>
      </Fields>
</Search>
<Results INITIALRESULTS=”User”>
      <Table>
            <Column ALIGN="LEFT">
                  <Display>PolicyNumber</Display>
                  <Name>Policy Number</Name>
                  <Group>Policy</Group>
            </Column>
            <Column ALIGN="LEFT">
                  <Display>Policy Name</Display>
```

```
                        <Name>PolicyName</Name>
                        <Group>Policy</Group>
                  </Column>
                  <Column ALIGN="LEFT">
                        <Display>Status</Display>
                        <Name>StatusCode</Name>
                        <Group>Policy</Group>
                  </Column>
                  <Column ALIGN="LEFT">
                        <Display>Insured</Display>
                        <Name>Name</Name>
                        <Group ROLECODE="01">Client</Group>
                  </Column>
                  <Column ALIGN="LEFT">
                        <Display>Tax ID</Display>
                        <Name>TaxID</Name>
                        <Group ROLECODE="01">Client</Group>
                  </Column>
                  <Column ALIGN="LEFT">
                        <Display>Issue State</Display>
                        <Name>StateCode</Name>
                        <Group>Policy</Group>
                  </Column>
                  <Column ALIGN="LEFT">
                        <Display>PlanDate</Display>
                        <Name>PlanDate</Name>
                        <Group>Policy</Group>
                  </Column>
            </Table>
      </Results>
</PolicySearchScreen>
```

XML Schema

```
<PolicySearchScreen>
      <AutoSelect>[Yes|No]</AutoSelect>
      <Search>
            <FixedFields>
                  <Fields>
                        <Field>
                              <Name></Name>
                              <Group ROLECODE=""></Group>
                              <Display MASK=""></Display>
                              <DataType></DataType>
                              <Encrypt ERASE="[Yes|No]">[Yes|No]</Encrypt>
                              <Value></Value>
                              <Disabled>[ReadOnly|Exists|...]</Disabled>
                              <Hidden></Hidden>
                              <Length></Length>
                        </Field>
                  </Fields>
            </FixedFields>
            <Fields>
                  <Field>
                        <Name></Name>
                        <Group ROLECODE=""></Group>
                        <Display MASK=""></Display>
                        <DataType></DataType>
                        <Encrypt ERASE="[Yes|No]">[Yes|No]</Encrypt>
                        <Value></Value>
                        <Disabled>[ReadOnly|Exists|...]</Disabled>
                        <Hidden></Hidden>
                        <Length></Length>
                        <DefaultValue></DefaultValue>
                        <Query TYPE="[SQL|FIXED|RADIO]"></Query>
                        <Calculated TYPE="" METHOD="" PLAN=""></Calculated>
                  </Field>
            </Fields>
      </Search>
      <Results INITIALRESULTS="[User|None]">
            <Table NAME="">
                  <Column ALIGN="" FORMAT="" EDITABLE="" RECONCILE="[Yes|No]"
                  FIELD="" TOTAL="[Yes|No]">
                        <Display></Display>
                        <Name></Name>
                        <Group ROLECODE="">[Role|Client|Address]</Group>
                        <DataType></DataType>
                        <Query TYPE="[SQL|FIXED|RADIO]">
                              <OptionValue></OptionValue>
                              <OptionText></OptionText>
                        </Query>
                        <Calculated TYPE="" METHOD="" PLAN=""></Calculated>
                  </Column>
            </Table>
      </Results>
</PolicySearchScreen>
```

RoleScreen

Description

This business rule allows for the configuration of the Role screen and defines the dynamic fields that can be displayed and updated on the specified Role Detail(s) screens. The plan selected during the policy entry process dictates which role options are visible and available on the Role screen.

RoleScreen Element/Attribute Table

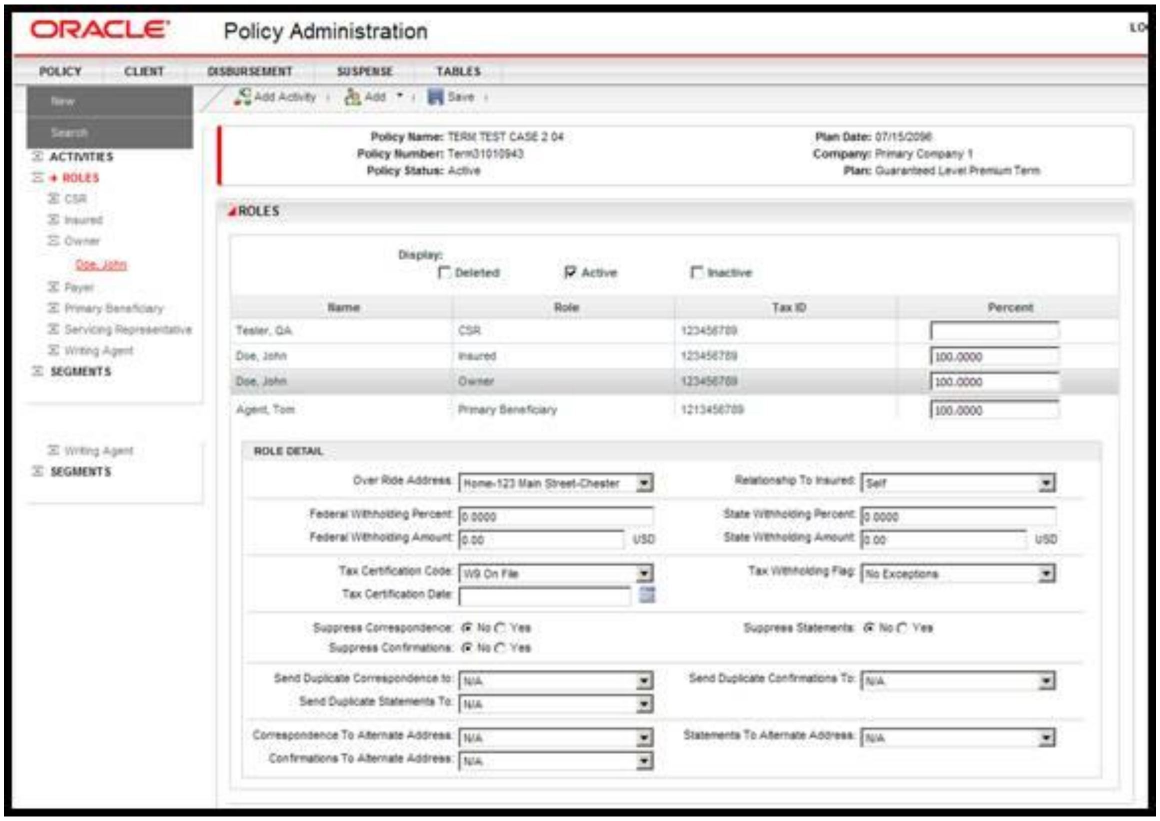
Element/Tag	Attribute	Definition	Element/Attribute Value and Description
<RoleScreen>		The opening and closing tag for the business rule.	
<DisplayRoleFields>		Will control whether role codes will be displayed as drop down list or checkboxes.	**Yes**: the role codes will be listed in a drop down box. The role details for the role selected will be displayed above the 'Add' button. **No**: this is the default behavior. The role codes will be displayed as checkboxes. The role details will not be displayed for the role selected. Note: In order for the Role screen to validate data, this element must have a value of Yes.
<Role>		**Required/Repeatable Element:** Parent element that has sub- elements to hold required information or have available information for a specific role that is defined separately by this tag.	
	ROLECODE	**Required Attribute:** The role code (as defined in AsCodeRole) that is to receive the definition described within the	**RoleCode**

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
		parent element.	
	NAME	Provides a user-friendly description of the role affected by this definition.	
<SuccessorOnDelete>		**Optional element:** This element allows the system to replace one role with another upon deletion of the original role. This is the role with which the original role should be replaced, upon its deletion. **Note:** The original role's RoleCode is specified in the <Role> tag. If the new role does not exist, then the user is warned with the same.	**Required element value:** **RoleCode**
	DELETEOLDROLE		**Yes:** Delete old role. **No:** Keep old role.
	IGNOREONSTATUS	**Optional Attribute:** Indicates the case statuses where the role will not be replaced.	**StatusCodes** The status codes are defined by the AsCodeStatus table.
<Ordinal>		**Optional element:** This element allows roles to be displayed in a specific order, according to the ordinal value. **Note:** Each role must have a unique ordinal value, and all ordinal values must be non-negative integers.	**Integer** Roles without an ordinal value are listed alphabetically by role description, with a secondary order that is alphabetical by client name.
<Fields>		Allows configuration of dynamic fields. See Fields Elements.	
<Events>		Allows configuration of events on the Role screen. Although the	

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
		Events element is widely used in configuration, the configuration explained here is specific to the RoleScreen rule. See the Action/Events page for an explanation of the elements and attributes generally available for Event configuration.	
	TYPE	Defines the type of event that should occur.	**ONSUBMIT**: Creates a Save button on the Role screen for existing roles. This button is used when updating role information. **ONCLICK**: The event will take place upon the user clicking the button. This value is used with BUTTON="Add" and BUTTON="Delete".
	BUTTON	Defines the type of button that will display on the Role screen.	**Add**: Creates an Add button, which will add a new role to the policy. **Delete**: Creates a Delete button, which will delete an existing role from the policy.
<ScreenMath>		See ScreenMath Element.	
<Actions>		See Action/Events.	
<RoleViews>		**Optional:** Opening tag for role view configuration.	The absence of RoleViews configuration results in the display of a default view (defined as Client Name, Role, Status, Tax ID and Percent Box). This default view applies to each role for which the optional RoleView has not been configured.
<DisplayStatus>		**Optional:** Configures the status filter for the views.	

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
		If <DisplayStatus> is absent from the configuration, only active roles are included.	
<Status>		**Required**if DisplayStatus is configured, **Repeatable:** Identifies the role status(es) that may be displayed for the view.	Single role status code (no comma-delimited lists). Role statuses allowed (AsCodeRoleStatus code): Active (01), Inactive (98), and Deleted (99).
<View>		**Optional, Repeatable:** Opening tag for view configuration.	
	NAME		**Required Attribute** (if View element is configured): The name of each View element.
<Roles>		**Required** (if Views are configured): Defines the roles that are included in the view.	
	ALLROLES	**Optional:** Specifies that a view contains all policy level roles.Do not specify <Role> element if ALLROLES is specified.	**Yes|No** The default value is No.
<Role>		**Required, Repeatable:** Defines the role(s) that are included in the view. Do not specify <Role> element if ALLROLES is specified.	Single role code (no comma-delimited lists).
<Table>		**Optional:** Common column definition to define the columns for the view. See Tables.	Available Groups are: Role, RoleField, Client, ClientField, Policy, and PolicyField.

RoleScreen Image (Default View)



Policy Name: TERM TEST CASE 2 04
Policy Number: Term31010943
Policy Status: Active
Plan Date: 07/15/2096
Company: Primary Company 1
Plan: Guaranteed Level Premium Term

ROLES

Display: ☐ Deleted ☑ Active ☐ Inactive

Name	Role	Tax ID	Percent
Tester, QA	CSR	123456789	
Doe, John	Insured	123456789	100.0000
Doe, John	Owner	123456789	100.0000
Agent, Tom	Primary Beneficiary	1213456789	100.0000

ROLE DETAIL

Over Ride Address: Home-123 Main Street-Chester
Relationship To Insured: Self
Federal Withholding Percent: 0.0000
Federal Withholding Amount: 0.00 USD
State Withholding Percent: 0.0000
State Withholding Amount: 0.00 USD
Tax Certification Code: W9 On File
Tax Certification Date:
Tax Withholding Flag: No Exceptions
Suppress Correspondence: [✓ No] Yes
Suppress Confirmations: [✓ No] Yes
Suppress Statements: [✓ No] Yes
Send Duplicate Correspondence to: N/A
Send Duplicate Statements To: N/A
Send Duplicate Confirmations To: N/A
Correspondence To Alternate Address: N/A
Confirmations To Alternate Address: N/A
Statements To Alternate Address: N/A

XML Example

```
                  <RoleScreen>
                      <DisplayRoleFields>Yes</DisplayRoleFields>
                      <Role ROLECODE=”13”>
                <Fields>
                    <Field>
                        <Name>OverRideAddress1</Name>
                        <Display>Override Address</Display>
                        <DataType>Combo</DataType>
                        <Query TYPE=”SQL”>SELECT &ldots;. </Query>
                    </Field>
                    <Field>
                        <Name>Line</Name>
                        <Display></Display>
                        <DataType>Line</DataType>
                    </Field>
                    <Field>
                        <Name>Line</Name>
                        <Display></Display>
                        <DataType>Line</DataType>
                    <Field>
```

```
            <Fields>
            <Fields>
               <Field>
                  <Name>AlternateCorrespondence</Name>
                  <Display>Correspondence to Alternate Address</Display>
                  <DataType>Text</DataType>
               </Field>
               <Field>
                  <Name>AlternateConfirmation</Name>
                  <Display>Confirmations to Alternate Address</Display>
                  <DataType>Text</DataType>
               </Field>
               <Field>
                  <Name>AlternateStatements</Name>
                  <Display>Statements to Alternate Address</Display>
                  <DataType>Text</DataType>
               </Field>
            </Fields>
                   <Ordinal>1</Ordinal>
                </Role>
                <RoleViews>
                   <DisplayStatus>
         <Status>98</Status>
                   </DisplayStatus>
                   <View NAME="Beneficiaries">
         <Roles ALLROLES="No">
            <Role>01</Role>
         </Roles>
                   </View>
                </RoleViews>
             </RoleScreen>
```

XML Schema

```
             <RoleScreen>

                <Role ROLECODE="[CodeString]">
                   <Ordinal>[integer]</Ordinal>
                   <Fields> </Fields>
                   <SuccessorOnDelete DELETETOLDROLE="[Yes|No]"
                   IGNOREONSTATUS="[StatusCode]">[RoleCode]</SuccessorOnDel
                   ete>

                </Role>

                <RoleViews>
                   <DisplayStatus>
         <Status>[role status code]</Status>
```

```
              </DisplayStatus>
              <View NAME="[name of the view]">
<Roles ALLROLES="Yes|No">
   <Role>[role code]</Role>
</Roles>
<Table>[common column definition]</Table>
              </View>
           </RoleViews>
        </RoleScreen>
```

PostAssignmentValidateExpressions

Description

The PostAssignmentValidateExpressions business rule is optionally associated with a transaction as a transaction override. It does not exist in the TransactionBusinessRulePacket. This allows the rule to process each time the activity processes while in an NUV pending state. The rule contains a math section that allows for standard math calculations, and a validation section that supports standard validation expressions.

Note: Named MathVariables in this rule must be unique and not replace any MathVariables in the transaction math section. However, MathVariables from the transaction are available to the math section of this rule. Valuation values modified by Assignment processing are also available.

PostAssignmentValidateExpressions Element/Attribute Table

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
<PostAssignmentValidateExpressions>	The opening and closing tag		Overridden at the transaction level.
<MathVariables>	Standard math section		
<MathVariable>	Standard math syntax		
<ValidateExpressions>	**Optional element:** Standard validation section		
<Expression>	**Required / Repeatable element**: This tag specifies a condition to be evaluated True/ False, and the error message to be displayed on the screen if the condition is met.		**Required element value**: **Expression** Specifies the condition that needs to be evaluated upon trying to process the transaction.
		TYPE	**Required Attribute**: **ErrorOnTrue** If the Expression is true then the error message displays. **ErrorOnFalse** If the Expression is false then the error message displays.
		OVERRIDABLE	**Optional Attribute:** **Yes**: the error messages will be displayed with a check box in the Error window. Clicking the checkbox will

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
			allow the user to override the error. **No**: Not overridable.
		ERRORNUMBER	**Optional Attribute**: **ErrorNumber**: Indicates the field value to define error message activity. Code values for error message from the AsErrorCatalog database table. Example: LH001, U001, V000, V001, V012, V020 Note: If the OVERRIDABLE attribute is set to "Yes," the error numbers entered as values of this attribute will display on the Overridable Errors section of the Transaction Security pane. The security group's ability to override each error can be configured on this pane.
		MESSAGE	**Required Attribute: ="ErrorMessage"** This attribute is used to specify the error message that is displayed upon evaluating the condition. **Note:** The value of a Math Variable or a Segment Field can be substituted in the error message surrounded by $$$. See General Structure and Best Practices
<GenerateAccounting>	**Optional element:** Indicates if accounting detail records per COA entries are written.		**No** **Yes** Literal or MathVariable. Default value is “Yes”.

XML Example

```
<PostAssignmentValidateExpressions>
   <MathVariables>
      <MathVariable VARIABLENAME="ActiveStatusCode"
      TYPE="VALUE" DATATYPE="TEXT">01</MathVariable>
      <MathVariable VARIABLENAME="PendingStatusCode"
      TYPE="VALUE" DATATYPE="TEXT">08</MathVariable>
```

```
                <MathVariable VARIABLENAME="True" TYPE="VALUE"
                DATATYPE="INTEGER">1</MathVariable>
                <MathVariable VARIABLENAME="False" TYPE="VALUE"
                DATATYPE="INTEGER">0</MathVariable>
                <MathVariable VARIABLENAME="WriteAccounting"
                TYPE="VALUE" DATATYPE="TEXT">No</MathVariable>
                <MathVariable VARIABLENAME="Condition1" TYPE="SQL"
                DATATYPE="INTEGER">SELECT...</MathVariable>
                <MathVariable VARIABLENAME="Condition2"
                TYPE="POLICYFIELD"
                DATATYPE="TEXT">StatusCode</MathVariable>
                <MathIF IF="Condition2 = ActiveStatusCode">
<MathVariable VARIABLENAME="Condition1" TYPE="EXPRESSION"
DATATYPE="INTEGER">Condition1 + 1</MathVariable>
<MathVariable VARIABLENAME="WriteAccounting" TYPE="VALUE"
DATATYPE="TEXT">Yes</MathVariable>
                </MathIF>
                <CopyBook>CopyBook-ActivityValuation</CopyBook>
            </MathVariables>
            <ValidateExpressions>
                <Expression TYPE="ErrorOnTrue" OVERRIDABLE="No"
                MESSAGE="Message Text 1">Condition1 = True</Expression>
                <Expression TYPE="ErrorOnTrue" OVERRIDABLE="Yes"
                 MESSAGE="Owner’s Age of $$$OwnerAgeMV$$$ must be less
                than or equal to $$$MaxAgeNY$$$">
                 OwnerAgeMV &gt; MaxAgeNY And IssueStateCodeMV = '32'
                </Expression>
                <Expression TYPE="ErrorOnTrue" OVERRIDABLE="No"
                MESSAGE="Message Text 3">Condition2 =
                PendingStatusCode</Expression>
            </ValidateExpressions>
            </GenerateAccounting>WriteAccounting</GenerateAccounting>
        </PostAssignmentValidateExpressions>
```

Suspense Elements

Description

The <Suspense> and <MultiSuspense> elements control the ability to apply suspense to an activity.
A transaction can have a <Suspense> element or a <MultiSuspense> element, but not both.
If a <Suspense> element is present in the transaction, a Suspense tab will be available on the Activity Details screen with a Suspense field, and a suspense record will be written in AsSuspense when the activity processes.
If a <MultiSuspense> element is present in the transaction, a Suspense tab will be available on the Activity Details screen with a Suspense section, and multiple suspense records will be written in AsSuspense when the activity processes.

Suspense Element/Attribute Table

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
<Suspense>	The start and end tags of the element. Cannot be used with <MultiSuspense> element.		Activity field containing the amount of the suspense that will be attached.
		AUTOENTRY	**Yes**: A suspense record is created for the amount of the element's reference field. This value is also automatically attached. **No**: A suspense record must exist and be attached by the user to the activity.
		OVERRIDABLE	**Yes**: Errors when received may be overridden by the user. **No**: Errors may not be overridden by the user. **Ignore**: Ignore errors
		VALUE	**Sufficient**: The value of the suspense record must be at least the value of the activity field. **Equals**: The value of the suspense record must equal the value of the activity field.
		REQUIRED	**Yes**: A suspense record must be selected from the Suspense tab of the Activity Detail screen. This is the default behavior that occurs if this attribute is omitted. **No**: No suspense record needs to be selected from the Suspense tab of the Activity Detail screen. If a suspense record is not selected, then no suspense record will be attached to the activity.

MultiSuspense Element/Attribute Table

Element/Tag	Definition	Attribute	Element/Attribute Value and Description
<MultiSuspense>	The start and end tags of the element. This element is used so multiple suspense items can be attached to the		Activity field containing the total amount of suspense that will be attached.

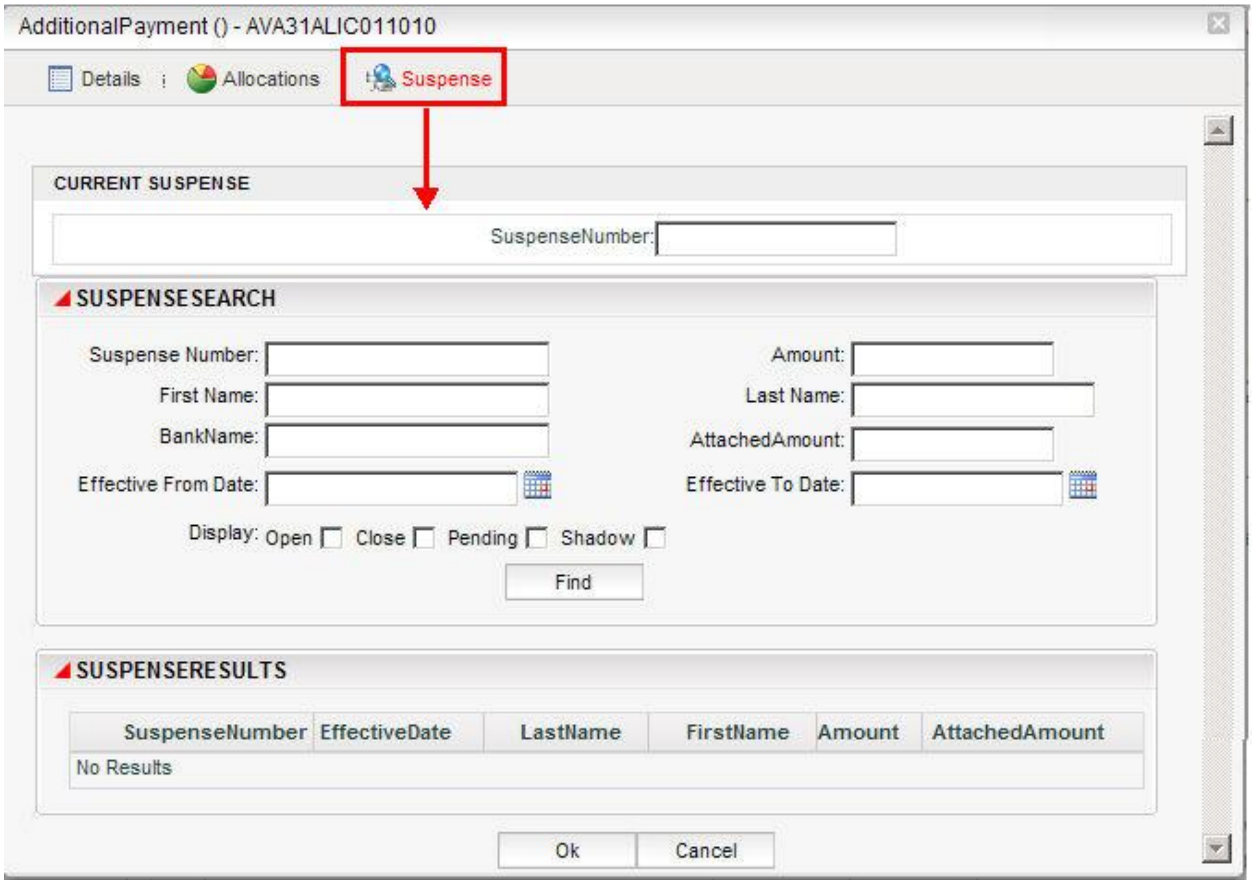
Element/Tag	Definition	Attribute	Element/Attribute Value and Description
	Activity. Cannot be used with <Suspense> element	FIELD	**String**
		START	**Integer** Defines the minimum number of suspense items.
		STOP	**Integer** Defines the maximum number of suspense items.

XML Example—Suspense

```
<Transaction>
…
   <Suspense REQUIRED=”No”
   VALUE=”Sufficient”>PremiumAmount</Suspense>
   <Fields>
      <Field>
<Name>PremiumAmount</Name>
<Display>Premium Amount</Display>
<DataType>Decimal</DataType>
      </Field>
   </Fields>
…
</Transaction>
```

XML Example—MultiSuspense

```
<MultiSuspense START="0" STOP="3">BonusAmount</MultiSuspense>
```



Example Suspense Activity Detail Screen

Fields Element

Description

The following elements allow for the configuration of fields. Fields dictate the entry information required for screens and activity processing.

Fields Element/Attribute Table

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
<Fields>		The opening and closing tag of the fields section.	
<Field>		The opening and closing tag for each field being configured.	
<Name>		**Required:** Exact name of the field. This is the name	

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
		that is found in the configuration whenever a reference is being made to this field.	
	BOLD ITALICS	**Optional:** Renders bold or italicized representation of a dynamic field's contents. If added to an inquiry's Input section <Name> element and set to Yes, then BOLD and ITALICS attributes will be ignored.	**Yes**: field contents will be bold and/or in italicized. **No**: contents will not have bold or italicized content.
<Group>		**Optional:** This is the name of the group (table) that should be used to obtain the value. This is used only where source data may come from multiple tables (i.e., search screens).	
	ROLECODE		RoleCode from ASRole table.
<Display>	BOLD ITALICS	**Optional:** Renders bold or italicized representation of a dynamic field's contents. If added to an inquiry's Input section <Name> element and set to Yes, then BOLD and ITALICS attributes will be ignored.	**Yes**: field contents will be bold and/or in italicized. **No**: contents will not have bold or italicized content.
<DataType>		**Required:** A classification identifying one of various types of data which provide general definition to the appropriate values, the operations that can be performed on the type and how the value is stored.	**Blank:** Used for Filler fields. **Check:** Displays check box on the screen. **Client:**Provides the user with a list of all available clients from the AsClient table. Allows the user to invoke the ClientSearchScreen from within ClientScreen, PolicyScreen, SegmentScreen, and ActivityDetailScreen.

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
			Combo: Combination/drop down field . **Date:** Date field with calendar icon. **Decimal:** Displays decimal point and calculator icon. **Identifier:** Generates a unique value by combining various values from other fields and a sequence number. **Integer:** Formats to a whole number. Displays calculator icon. **Label:** Displays a text label left-justified within the column of fields in which it is placed. **Line:** Displays line across frame for aesthetics. **Message:** Generates a static or dynamic message (using values from fields). **Money:** Displays a monetary value along with its currency code and Calculator icon. **Percent:** Displays percent sign. **Radio:** Mutually exclusive options. Selecting one automatically unselects the others. **Role:** **Text:** Free form entry. **TextArea:** Variable-length dynamic text field. **Note:** TextArea is not supported within the <Table> element. **Title:**Displays a bolded text label centered across both columns of fields.
	MASK	**Optional:** Used to specify the mask used to format or conceal the field data for display purposes.	**Text:** Any mask name that has been defined.
	CALENDAR	**Optional:** Used when DataType is set to “Date” to designate appropriate calendar format.	Gregorian, JP, JP_IMP
	FORMAT	**Optional:** Used when DataType	Translation Key values in AsTranslation table. Cannot be

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
		is set to "Date", and designates the appropriate date format.	Date.Format
<Parts>		**Optional:** Used when DataType is set to "Identifier".	
<Part>		See Parts element for details.	Specifies the parts to generate the identifier field.
<DefaultValue>		Default value of field. If set to SYSTEMDATE, the default value will be set to the system date.	**String**, **Code** or **Integer** (depending on DataType).
<Query>	TYPE	Used with the Combo datatype to retrieve options for the combo box.	**SQL:** SELECT Statement that returns the values to be displayed in the combo box. The Query must return a Key and a Value. The Key is a unique pointer to the Value. Value is a user-friendly description and is visible in the combo box. **FIXED:** Uses Options tag to fill the options.
<Options>		Individual option value. Used with the Combo and Radio datatypes.	
<Option>		Each option set needs to be contained in this opening and closing tag.	
<OptionValue>		Code Value for the option. This is the key value that is stored in the database.	**Date, Decimal, Integer, or String** Value
<OptionText>		Text that will display in drop down box or radio buttons.	**String** Value
<Calculated>		Allows execution of a calculated element to populate the current field's value.	If the TYPE attribute is set to SQL, this element should contain a SQL query that is used to populate the field. Entity screens (e.g. AddressScreen, PolicyScreen) can access GUIDs belonging to their corresponding entity (e.g. AddressGUID, PolicyGUID) using this query. Additionally, the Client

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
			screen can access PolicyGUID if the screen is accessed via a policy, and the Role screen can access the ClientGUID for the currently selected client.
	TYPE	Defines the type of calculation to be performed.	**SQL:** Allows for a SQL query to be executed with the result being the data displayed in the field. It must be a single column, single row return. **REPLACE:** Forces the field to accept the value of the field referenced by the Calculated element's value. **MESSAGE:** Sets the element to contain a static message (containing only text) or dynamic message (with substitution of values from embedded fields). **FUNCTION:** Specifies a function to be called if the Calculated element is executed.
	METHOD		**IFEMPTY:** Execute the Calculated element if the field is empty. **FORCE:** Used to force execution of the Calculated element every time the screen is loaded.
<ClearOnRecycle>		**Optional**Allows recycled activities to remove the value of a field that was previously populated.	**Yes:** The value of this field is cleared when the activity is manually recycled. **No:** The value of this field is not changed when the activity is manually recycled.
<Disabled>		Allows field entry.	**Yes:** Field is locked down and grayed out. **No:**Field is open to user entry. **ReadOnly:**Field is locked down and greyed out.
<Hidden>		Controls the visibility of the field on the screen.	**Yes:** Field is hidden from user. **No:** Field is visible.
<Expanded>		**Optional:** Used on Client screen configuration with datatype of Combo, Radio, or Text.	**Yes:** Field is expanded. If a field is expanded, it is the only field in the line. **No:** Field is not expanded.
<Required>		**Optional:** Indicates if the field is a required field for	**Yes** or **No** **Yes** – Field is required. **Note:** If a user is prevented by

Element/Tag	Attribute	Definition	Element/Attribute Value and Description
		entry. A required field must not be blank or null when the screen is submitted. **Note:** This element applies only to the following screens: Client, Policy, Segments, Activities, Suspense, and Inquiry. It is not valid if the DataType is Identifier.	the security configuration from entering data in a required field, the screen will not pass validation when submitted. **No** – Field is not required (this is the default value).
<Currency>		**Optional:** Used when the DataType is set to "Money". When added to Inquiry's Output section's Fields <Currency> element it will be ignored.	A list of one or multiple currency codes from the AsCurrency table (e.g. USD, JPY, THB, etc…)
<DefaultCurrency>		**Optional:** Used when DataType is set to "Money". When added to Inquiry's Output section's Field, <DefaultCurrency> element it will be ignored.	One currency code from AsCurrency table (e.g. USD, JPY, THB, etc…)

XML Example

General Example

```
<Field>
   <Name>TaxID</Name>
   <Display>Tax ID</Display>
   <DataType MASK="SSN">Text</DataType>
   <Disabled>No</Disabled>
   <Hidden>No</Hidden>
</Field>
```

Line DataType Example

```
<Field>
   <Name>Line1</Name
```

```
            <Display>Line</Display>
            <DataType>Line</DataType>
        </Field>
```

Radio DataType/OptionText population Example

```
        <Field>
            <Name>AccountType</Name>
            <Display>Account Type</Display>
            <DataType>Radio</DataType>
            <Query TYPE="FIXED">
                <Options>
<Option>
    <OptionValue>C</OptionValue>
    <OptionText>Checking</OptionText>
</Option>
<Option>
    <OptionValue>S</OptionValue>
    <OptionText>Savings</OptionText>
</Option>
                </Options>
            </Query>
        </Field>
```

Calculated Example

```
        <Field>
            <Name>Prefix</Name>
            <Display>Prefix</Display>
            <DataType>Text</DataType>
            <Calculated TYPE="SQL">SELECT CASE WHEN '[PolicyStatus]' =
            '08' THEN 'B' ELSE 'R' END</Calculated>
        </Field>
```

Combo DataType/SQL Population Example

```
        <Field>
            <Name>BankStateLocation</Name>
            <Display>Bank State</Display>
            <DataType>Combo</DataType>
            <Query TYPE="SQL">SELECT CodeValue, ShortDescription FROM
            AsCode
              UNION SELECT '$$$Blank$$$', ' ' ORDER BY
            ShortDescription</Query>
        </Field>
```

Currency Example

```
<Field>
   <Name>AnnualPremium</Name>
   <Display>Annual Premium</Display>
   <DataType>Money</DataType>
   <Disabled>Yes</Disabled>
   <DefaultValue>0</DefaultValue>
   <Currency>KRW,THB,INR,USD</Currency>
   <DefaultCurrency>USD</DefaultCurrency>
</Field>
```

Parts Example #1

```
<Field>
   <Name>ClientID</Name>
   <Display>Client Name ID</Display>
   <DataType>Identifier</DataType>
   <Parts>
      <Part TYPE="SEQUENCE"
      FORMAT="0000000000">ClientID</Part>
   </Parts>
</Field>
```

Parts Example #2

```
<Field>
   <Name>ActivityWarrantNumber</Name>
   <Display>Activity Warrant Number</Display>
   <DataType>Identifier</DataType>
   <Parts>
      <Part TYPE="FIELD" LEFT="1">Prefix</Part>
      <Part TYPE="SEQUENCE"
      FORMAT="0000000">AsActivity_RBWarrantNumber</Part>
   </Parts>
</Field>
```

XML Schema

```
<Field>
   <Name>. . .</Name>
   <Display>. . .</Display>
   <DataType>Identifier</DataType>
   <Parts>
      <Part TYPE="VALUE">[literal]</Part>
```

```
          <Part TYPE="SYSTEMDATE"
          FORMAT="YY|MM|DD|YYYY|YYMM|YYYYMM|YYMMDD|YYYYMMDD"></Par
          t>
          <Part TYPE="FIELD" LEFT="[integer]" FORMAT="[padding
          characters]">[field name]</Part>
          <Part TYPE="FIELD" RIGHT="[integer]" FORMAT="[padding
          characters]">[field name]</Part>
          <Part TYPE="FIELD" MID="[integer, integer]"
          FORMAT="[padding characters]">[field name]</Part>
          <Part TYPE="SEQUENCE" FORMAT="[padding characters]"
          SEQUENCEDATE="[field
          name]|SYSTEMDATE|EffectiveDate">[sequence  name]</Part>
      </Parts>
  </Field>
```

COLLECTION and KEY

Description

This math element can be used to create a collection, set the value of a math variable from an existing math variable or activity field, or execute a SQL statement that will return two result columns as a key-value pair. In the latter case, the first column is considered the key with the second column a value. A subsequent statement of type COLLECTIONVALUE is used to access the key to retrieve its associated value (see the last XML example).

COLLECTION and KEY Element/Attribute Table

TYPE=COLLECTION KEY=GUID			
Element	Attribute	Attribute Value	Element Value
OPERATION="CREATE"			
<MathVariable>			Empty
	VARIABLENAME	String: The name of the math variable.	
	TYPE	COLLECTION	
	OPERATION	CREATE Creates a COLLECTION math variable.	
	DATATYPE	MAP	
	ORDERED	**Yes**: The order of the collection will be persisted until the final instance of the math variable. **No**: The order of the collection will not be persisted. This is the default behavior if this attribute is omitted from configuration.	
OPERATION="SETVALUE"			
<MathVariable>			String: Refers to the name of a COLLECTION math variable created with OPERATION="CREATE".
	VARIABLENAME	String: The name of the math variable whose value is set by this configuration.	

TYPE=COLLECTION KEY=GUID			
	TYPE	COLLECTION	
	OPERATION	SETVALUE Assigns a value to an existing or new math variable once the collection is created. Use a previously defined Math Variable or Activity Field to replace the value /	
	DATATYPE	BIGTEXT DATE TEXT INTEGER DECIMAL	
	ROUND	Integer If DATATYPE="DECIMAL"	
	KEY	The KEY indicates the type of value the configuror is using to map or insert into a new math variable name within the collection. For example a SegmentGUID for a SegmentLoop to update the face amount or active code, a unique naming convention, or fund code.	
	DEFAULT	String	
	LOG	**Yes:** the result is stored in AsActivityMath table to be used by other activities. **No:** the result is not stored. This is the default behavior.	
	ORDERED	**Yes**: The order of the collection will be persisted until the final instance of the math variable. This behavior is akin to the LOG attribute being	

TYPE=COLLECTION KEY=GUID			
		set to "Yes". **No**: The order of the collection will not be persisted. This is the default behavior if this attribute is omitted from configuration. Note:{/b} If the ORDERED attribute is used on a COLLECTION math variable with OPERATION set to "SETVALUE", ORDERED must be used on the first instance of the collection that inserts a value.	
OPERATION attribute omitted			
<MathVariable>			**Option 1: One of the following:** <SqlServer>[SqlStatement]</SqlServer> <Oracle>[SqlStatement]</Oracle> <DB2>[SqlStatement]</DB2> **Option 2:** [SqlStatement]
	VARIABLENAME	String	
	TYPE	COLLECTION	
	DATATYPE	MAP	
	ORDERED	**Yes**: The order of the collection will be persisted until the final instance of the math variable. **No**: The order of the collection will not be persisted. This is the default behavior if this attribute is omitted from configuration.	

XML Examples

OPERATION="CREATE"

```
<MathVariable VARIABLENAME="SegmentAnnPremiumAmt"
TYPE="COLLECTION"
   OPERATION="CREATE" DATATYPE="MAP"/>
```

OPERATION="SETVALUE"

```
<MathVariable VARIABLENAME="SegmentAnnPremiumAmt"
TYPE="COLLECTION"
   OPERATION="SETVALUE" KEY="CurrentSegmentGUID"
   DATATYPE="CURRENCY"
   ORDERED="YES">CurrentYearsAnnualPremiumUSD
</MathVariable>
```

OPERATION Attribute Omitted

```
          <MathVariable VARIABLENAME="SegmentCollection"
          TYPE="COLLECTION" DATATYPE="MAP">
             SELECT Field.FieldName,
                CASE
     WHEN Field.FieldTypeCode = '02' THEN Field.TextValue
     WHEN Field.FieldTypeCode = '03' THEN CAST(Field.IntValue AS
     CHAR(10))
     WHEN Field.FieldTypeCode = '04' THEN CAST(Field.FloatValue AS
     CHAR(32))
                END
             FROM AsSegment
                JOIN AsSegmentName ON AsSegmentName.SegmentNameGUID =
                AsSegment.SegmentNameGUID
     AND AsSegmentName.TypeCode = '04'
                JOIN AsSegmentField Field ON Field.SegmentGUID =
                AsSegment.SegmentGUID
     AND Field.FieldName IN ('SegmentIssueGender',
                             'SegmentUWClass',
                             'SegmentTobaccoPremBasis',
                             'SegmentAmount',
                             'SegmentIssueAge')
             WHERE AsSegment.PolicyGUID = '[Policy:PolicyGUID]'
          </MathVariable>
```

COLLECTIONVALUE Example

```
<MathVariable VARIABLENAME="BaseFace" TYPE="COLLECTIONVALUE"
   KEY="SegmentAmount" DATATYPE="DECIMAL" DEFAULT="-
   999999999">SegmentCollection
```

```
</MathVariable>
```

XML Schema

OPERATION="CREATE"

```
<MathVariable VARIABLENAME="[String]" TYPE="COLLECTION"
   OPERATION="CREATE"
   DATATYPE="MAP">
</MathVariable>
```

OPERATION="SETVALUE"

```
<MathVariable VARIABLENAME="[String]" TYPE="COLLECTION"
   OPERATION="SETVALUE" KEY="[String]"
   DATATYPE="DATE|TEXT|INTEGER|DECIMAL"
   ROUND="[Integer]" DEFAULT="[String]" LOG="Yes">[String]
</MathVariable>
```

OPERATION Attribute Omitted

```
<MathVariable VARIABLENAME="[String]" TYPE="COLLECTION"
DATATYPE="MAP">
   <!-- option 1 -->
      <SqlServer>[SqlStatement]</SqlServer>|<Oracle>[SqlStatem
      ent]</Oracle>|<DB2>[SqlStatement]</DB2>
   <!-- option 2 -->
      [SqlStatement]
</MathVariable>
```

ActionEvents

Description

ActionEvents provide the ability to perform actions on a field as an event occurs on the screen. Configuration establishes the important events and links them to action sets via ID attributes. When the system detects one of these events, it automatically executes the action set assigned to the event. ActionEvents can be used to:

- create custom error and warning messages that appear to the end user after the system validates what was entered.
- change fields from enabled to disabled (and vice versa) when an event occurs.
- change fields from hidden to displayed (and vice versa) when an event occurs.
- change a field's data according to the value in a trigger field.

ActionEvents Elements/Attributes Table

Element/Tag	Definition	Attributes	Element/Attribute Value and Description
<Events>	The opening and closing tag for the element.		
<Event>	**Repeatable:** Identifies the event that is being defined.	TYPE	ONLOAD ONCHANGE ONSUBMIT CALLEDEVENT: this will be used to call an EVENT which is configured outside the transaction in a business rule which is associated with the transaction. Currently this feature is available in the TransactionAllocationScreen BR and MultiFields BR. Please refer to the prototype configuration example available in Rules Palette guide for details of how and when this can be used.
		FIELD	Any field name in the <Fields> section. Used with ONCHANGE to identify the trigger field, which when it changes, causes the actions to process.
		ID	The unique ID for the event being invoked by the transaction. The ID value should match the ID in the CALLEXTERNALEVENT definition in the transaction XML.
<ScreenMath>	Opening and closing tags for Event Math section.		
<Math>	Call the <Math> that is defined in the	ID	Any ID name given in the Math section under <ScreenMath>.

Element/Tag	Definition	Attributes	Element/Attribute Value and Description
	<ScreenMath> section to be run whose name matches an ID.		
<MathVariables>			
<MathVariable>		TYPE	SUSPENSEFIELD VALUE
<Actions>			
<QuerySet>	Defines QuerySet.	ID	Any name (no spaces).
<Condition>	Optional conditional logic.	IF	Any expression resulting in a true or false value. When the expression results in a true value, the system passes execution to the first statement contained by the opening and closing tags of the parent element. Statement execution continues serially to the next statements until the closing parent tag, an Else element or an ElseIf element is reached. When the expression results in a false value, the system passes execution control to an Else or ElseIf element or closing parent tag, whichever is closest. Functions and direct date comparisons are available for use in the expression. Screen math variables and fields are available for use in the expression.
		VALUE	String.
<Action>	Defines whether to fill combo box with SQL query, or use fixed options.		**String:** SQL query string.
		ACTIONTYPE	**SQL QUERY** **MATHCOLLECTION**: References a screen math variable containing a collection. If this value is used, the value of the <Action> element should be a COLLECTION math variable defined in screen math. The following action types are the only ones available to Events in multifields: **HIDE** **SHOW** **DISABLE** **DISABLEALL**: Disables all fields, both dynamic and fixed, including multifields within a configuration section. This can only be used with the ONLOAD event

Element/Tag	Definition	Attributes	Element/Attribute Value and Description
			type. If used with ONSUBMIT or ONCHANGE it will be ignored. **ENABLE** **ASSIGN** **CALLEXTERNALEVENT**: This value is used to invoke an event that is configured outside the transaction in a business rule associated with the transaction. Currently this feature is available in the TransactionAllocationScreen BR and MultiFields BR. Please refer to the prototype configuration example available in Rules Palette guide for details of how and when this can be used.
		FIELD	The field name where the action is to occur. When used with MULTIFIELD this indicates which field within the multifield will be impacted. If INDEX is not indicated all instances of that field within the multifield will be updated.
		INDEX	Used with MULTIFIELD. Indicates the position/row of the field the Action is to occur on (starts at 0). If INDEX is used only the field for that index will be impacted.
		MULTIFIELD	The multifield name where the action is to occur. If only the MultiField is referenced then the event occurs on the entire section.
		ID	To be used in conjunction with CALLEXTERNALEVENT. The ID value should match the ID in the CALLEDEVENT definition in the MultiFields BR/TransactionAllocation BR.
<ElseIf>	ElseIf is executed when prior conditions in IF and ElseIf elements are not true. Use this element when there are further conditions to express in the condition structure. IF attribute must be included. Multiple ElseIf elements are	IF	Any expression resulting in a true or false value. When the expression results in a true value, the system passes execution to the first statement contained by the opening and closing tags of the parent element. Statement execution continues serially to the next statements until the closing parent tag, an Else element or an ElseIf element is reached. When the expression results in a false value, the system passes execution control to an Else or ElseIf element or

Element/Tag	Definition	Attributes	Element/Attribute Value and Description
	possible within a condition structure.		closing parent tag, whichever is closest. Functions and direct date comparisons are available for use in the expression. Screen math variables and fields are available for use in the expression.
		VALUE	String.
<Else>	Else is executed when prior conditions in IF and ElseIf elements are not true. Use this element when there are no further conditions to express in the condition structure. Only one Else is possible within a condition structure.		Execution is passed to the first Action statement contained within the opening and closing <Else> element. Each statement is serially executed until the closing tag is reached.
<ActionSet>	Defines ActionSet	ID	Any name (no spaces).
<Condition>	Same as defined above.		
<Action>	Defines the type of action that is to occur.		**Optional element value: String:** Message to be displayed if ACTIONTYPE is ERROR or WARNING, and the IF condition is met.
		FIELD	The field name where the action is to occur. This attribute is required when action type is equal to assign. It is also available when action type is equal to error or warning.
		ACTIONTYPE	ERROR Provides a message to the user. The user cannot proceed until the all errors are fixed. All errors will display in one section of the screen (if more than one occurs). When FIELD is used, the error will appear in the Validation section at the top of the screen. **Note:** The value of a Math Variable or a Segment Field can be substituted in the error message surrounded by $$$. See General Structure and Best Practices
			WARNING Provides a warning message to the

Element/Tag	Definition	Attributes	Element/Attribute Value and Description
			user. This is a warning to the user and the user may proceed even with the error messages. All errors will display in one section of the screen (if more than one occurs). When FIELD is used, the warning will appear in the Validation section at the top of the screen. **Note:** The value of a Math Variable or a Segment Field can be substituted in the warning message surrounded by $$$. See General Structure and Best Practices
			SHOW Will make hidden field visible.
			HIDE Will hide a visible field.
			ENABLE Allows data entry in an editable field.
			DISABLE Prohibits data entry in an editable field, and changes the background color.
			DISABLEALL Prohibits data entry in all editable fields and multifields within a configuration section. **Note:** The DISABLEALL option is available for the following screens: • ClientScreen • AddressScreen
			READONLY Prohibits data entry in a field and does not change background color.
			ASSIGN Allows a value to be set to the field defined in the FIELD attribute. This action does not trigger an OnChange event on the receiving field.
<ElseIf>	Same as defined above.		
<Else>	Same as defined above.		

XML Example Without Multifield Events

```
        <Events>
           <Event TYPE="ONLOAD">
   <QuerySet ID="OnLoadQualType" FIELD="FundingMethod"></QuerySet>
           </Event>
```

```
          <Event TYPE="ONCHANGE" FIELD="FundingMethod">
             <ActionSet ID="OnChangeFundingMethod"></ActionSet>
          </Event>
          <Event TYPE="ONSUBMIT">
             <ActionSet ID="FinalValidation"></ActionSet>
          </Event>
       </Events>
       <ScreenMath>
          <Math ID="GlobalScreenMath" GLOBAL="Yes">
             <MathVariables>
<MathVariable VARIABLENAME="NA" TYPE="VALUE"
DATATYPE="TEXT">00</MathVariable>
<MathVariable VARIABLENAME="NonQualified" TYPE="VALUE"
DATATYPE="TEXT">13</MathVariable>
<MathVariable VARIABLENAME="Replacement" TYPE="VALUE"
DATATYPE="TEXT">98</MathVariable>
             </MathVariables>
          </Math>
       </ScreenMath>
       <Actions>
          <QuerySet ID="OnLoadQualType">
             <Condition IF="QualType=GlobalScreenMath:NonQualified">
<Action ACTIONTYPE="SQLQUERY">SELECT AsCode.CodeValue,
AsCode.ShortDescription FROM AsCode WHERE AsCode.CodeName =
'AsCodeMoneyType' AND AsCode.CodeValue IN ('00','01','50') ORDER BY
AsCode.CodeValue</Action>
<Else>
<Action ACTIONTYPE="SQLQUERY">SELECT AsCode.CodeValue,
AsCode.ShortDescription FROM AsCode WHERE AsCode.CodeName =
'AsCodeMoneyType' AND AsCode.CodeValue IN ('00','01','98') ORDER BY
AsCode.CodeValue</Action>
</Else>
             </Condition>
          </QuerySet>
          <ActionSet ID="OnChangeFundingMethod">
             <Condition
             IF="FundingMethod=GlobalScreenMath:Replacement">
<Action ACTIONTYPE="ENABLE" FIELD="ReplacementType"></Action>
<Else>
<Action ACTIONTYPE="DISABLE" FIELD="ReplacementType"></Action>
<Action ACTIONTYPE="ASSIGN"
FIELD="ReplacementType">GlobalScreenMath:NA</Action>
</Else>
             </Condition>
          </ActionSet>
          <ActionSet ID="FinalValidation">
             <Condition IF="IsEmpty(GrossAmount)">
<Action ACTIONTYPE="ERROR">Gross Amount is a required
field.</Action>
```

```
<ElseIf IF="GrossAmount &lt;= 0">
<Action ACTIONTYPE="ERROR">Gross Amount must be greater than
$0.00.</Action>
</ElseIf>
            </Condition>
            <Condition IF="IsEmpty(EffectiveDate)">
<Action ACTIONTYPE="ERROR">Effective Date is a required
field.</Action>
<ElseIf IF="EffectiveDate &gt; SystemDate">
<Action ACTIONTYPE="ERROR">Effective Date cannot be in the
future.</Action>
</ElseIf>
            </Condition>
            <Condition IF="ReplacementType=GlobalScreenMath:NA And
            FundingMethod=GlobalScreenMath:Replacement">
<Action ACTIONTYPE="ERROR">A Replacement Type of
$$$ReplacementType$$$ must be chosen.</Action>
            </Condition>
        </ActionSet>
     </Actions>
```

XML Example With MultiField Events

```
     <Transaction>
        <EffectiveDate STATUS="Enabled"
        TYPE="SYSTEM"></EffectiveDate>
        <MultiFields RULE="MultiField-
        SGMultiFieldPrototype">Yes</MultiFields>
           <Fields>
<Field>
   <Name>TestField</Name>
   <Display>Non-MultiValue Field Test</Display>
   <DataType>Text</DataType>
</Field>
<Field>
   <Name>TransactionField</Name>
   <Display>TransactionField</Display>
   <DataType>Text</DataType>
</Field>
           </Fields>
           <ScreenMath>
<Math ID="GlobalScreenMath" GLOBAL="Yes">
   <MathVariables>
      <MathVariable VARIABLENAME="ValueFromScreenMath" TYPE="VALUE"
      DATATYPE="TEXT">ScreenMath</MathVariable>
   </MathVariables>
</Math>
           </ScreenMath>
           <Events>
```

```
<Event TYPE="ONCHANGE" FIELD="TestField">
   <ActionSet ID="TestChangeExternalAction"></ActionSet>
</Event>
<Event TYPE="ONCHANGE" FIELD="TestField">
   <ActionSet ID="TestChangeInternalAction"></ActionSet>
</Event>
<Event TYPE="ONLOAD">
   <ActionSet ID="OnLoadExternalAction"></ActionSet>
</Event>
<Event TYPE="ONCHANGE" FIELD="EffectiveDate">
    <ActionSet ID="ActionOnChange"/>
 </Event>

            </Events>
            <Actions>
<ActionSet ID="OnLoadExternalAction">
   <Action ACTIONTYPE="ASSIGN"
   FIELD="TransactionField">'OnLoadValue'</Action>
   <Action ACTIONTYPE="ASSIGN" MULTIFIELD="TextTestMF"
   FIELD="TextTest2" INDEX="1">'ValueFromTransaction'</Action>
</ActionSet>
<ActionSet ID="TestChangeExternalAction">
   <Condition IF="Not IsEmpty(TestField) And
   TestField='HideCombo'">
      <Action ACTIONTYPE="HIDE" MULTIFIELD="ComboTestMF"></Action>
   </Condition>
   <Condition IF="Not IsEmpty(TestField) And
   TestField='ShowCombo'">
      <Action ACTIONTYPE="SHOW" MULTIFIELD="ComboTestMF"></Action>
   </Condition>
</ActionSet>
<ActionSet ID="TestChangeInternalAction">
   <Condition IF="Not IsEmpty(TestField) And
   TestField='HideTransactionField'">
      <Action ACTIONTYPE="HIDE" FIELD="TransactionField"></Action>
   </Condition>
</ActionSet>
<ActionSet ID=”ActionOnChange“>
     <!--The MultiFields BR - MultiField-SGMultiFieldPrototype -
will contain an Event of type CALLEDEVENT with ID = ”MFEvent” -->
    <Action ACTIONTYPE="CALLEXTERNALEVENT" ID=”MFEvent”></Action>
 </ActionSet>

            </Actions>
            <Math>
<MathVariables>
   <MathVariable VARIABLENAME="One" TYPE="VALUE"
   DATATYPE="TEXT">One</MathVariable>
</MathVariables>
```

```
                </Math>
            </Transaction>
```

XML Example With ACTIONTYPE="MATHCOLLECTION"

```
            <Fields>
               <Field>
                  <Name>QuerySetTrigger</Name>
                  <Display>QuerySet Trigger</Display>
                  <DataType>Check</DataType>
               </Field>
               <Field>
                  <Name>OptionTextField</Name>
                  <Display>Option Text</Display>
                  <DataType>Combo</DataType>
                  <Query TYPE="SQL">SELECT CodeValue, ShortDescription
                  FROM AsCode WHERE CodeName = 'AsCodeDisbursementStatus'
                  ORDER BY CodeValue ASC</Query>
               </Field>
            </Fields>
            <Events>
               <Event TYPE="ONCHANGE" FIELD="QuerySetTrigger">
                  <QuerySet ID="QuerySetAction" FIELD="OptionTextField">
                  </QuerySet>
               </Event>
            </Events>
            <ScreenMath>
               <Math ID="ScreenMath" GLOBAL="Yes">
                  <MathVariables>
         <MathVariable VARIABLENAME="Collection1" TYPE="COLLECTION"
         ORDERED="Yes" DATATYPE="MAP">SELECT CodeValue, ShortDescription
         FROM AsCode WHERE CodeName = AsCodeStatus</MathVariable>
         <MathVariable VARIABLENAME="Collection2" TYPE="COLLECTION"
         ORDERED="Yes" DATATYPE="MAP">SELECT CodeValue, ShortDescription
         FROM AsCode WHERE CodeName =
         AsCodeDisbursementStatus</MathVariable>
                  </MathVariables>
               </Math>
            </ScreenMath>
            <Actions>
               <QuerySet ID="QuerySetAction">
                  <Condition IF="QuerySetTrigger = 'CHECKED'">
         <Action ACTIONTYPE="MATHCOLLECTION">ScreenMath:Collection1</Action>
         <ElseIf IF="QuerySetTrigger = 'UNCHECKED'">
            <Action
            ACTIONTYPE="MATHCOLLECTION">ScreenMath:Collection1</Action>
         </ElseIf>
                  </Condition>
```

```
      </QuerySet>
   </Actions>
```

XML Example of a MultiFields BR to Explain CALLEDEVENT Configuration

```
<MultiFields>
         <MultiField>
                 <Name>AddressMultifield</Name>
                 <Title>Address Information</Title>
                 <Start>1</Start>
                 <End>3</End>
                 <Fields>
                         <Field>
                                 <Name>AddressGUID</Name>

<Display>AddressGUID</Display>
                                 <DataType>Text</DataType>
                         </Field>
                 </Fields>
                 <Events>
                         <Event TYPE="CALLEDEVENT"
ID="MFEvent">
                                 <ScreenMath ID="ScreenMath"/>
                                 <ActionSet
ID="MFActionOnChange"/>
                         </Event>
                 </Events>
                 <ScreenMath>
                         <Math ID="ScreenMath" GLOBAL="No">
                                 <MathVariables>
                                         <MathVariable
VARIABLENAME="RoleGUIDMV" TYPE="MULTIFIELD"
INDEX="MultiValueFieldIndex"
DATATYPE="TEXT">RoleGUID</MathVariable>
                                         <MathVariable
VARIABLENAME="AddressGUIDMV" TYPE="SQL" DATATYPE="TEXT">SELECT
* FROM TABLE(PKG_INQUIRY_SCREEN.fncEftRoles('[RoleGUIDMV]',
'[EffectiveDate]'))</MathVariable>
                                 </MathVariables>
                         </Math>
                 </ScreenMath>
                 <Actions>
                         <ActionSet ID="MFActionOnChange">
                                 <Action ACTIONTYPE="ASSIGN"
MULTIFIELD="AddressGUID">AddressGUIDMV</Action>
                         </ActionSet>
                 </Actions>
         </MultiField>
 </MultiFields>
```