

Appendix 8 – Configuration Standards

Table of Contents

PURPOSE.....	3
ORGANIZATION OF DOCUMENT	3
GENERAL STRUCTURE & BEST PRACTICES	3
SQL (STRUCTURED QUERY LANGUAGE)	5
CONFIGURATION SYNTAX	6

Purpose

The purpose of this document is to provide common business structure and practices while configuring the OIPA Transactions and Business Rules. These standards reflect the most frequently used policies. Please adhere to these standards for the purpose of ensuring the configured 'code' is highly maintainable.

Organization of Document

This document is organized into three sections:

- General Structure & Best Practices
- SQL (Structured Query Language)
- Configuration Syntax

General Structure & Best Practices

Refer to Appendix 1 of the Technical Manual for a list of transaction elements.

The general sequence of the elements should be:

1. EffectiveDate
2. Allocations (if applicable)
3. Suspense
4. Withholding
5. Valuation (if applicable)
6. Fields
7. OnLoad
8. OnChange
9. Validation
10. Math (Within the <Math> section, any MathVariable constants should come first.)
11. Assignments (if applicable)
12. ActivityAmounts (if applicable)
13. MultiFields
14. Spawns.

(When defining the constant of 'Checked' or 'UnChecked', the values should ALWAYS be either 'CHECKED' or "" in all caps, respectively.)

- 1) When defining OnChange, provide logic/values for the defaults (i.e. the values and behavior from the initial screen load). This is done in case you make a change that triggers OnChange but then changes the value back.
- 2) TransactionCosmetics should not appear in a TransactionBusinessRulePacket business rule.

- 3) Do not include the "Valuation" element/tag for transactions if valuation is not necessary.
- 4) Give thought to the order of business rules listed in TransactionBusinessRulePacket.

Assuming all these rules apply to an example transaction, the order of the rules should be as follows:

1. ValidateExpressions
2. GeneratePendingRequirement
3. DeliveryRequirements
4. CopyToPolicyFields
5. CopyToSegmentFields.

The rules listed in the packet are processed top to bottom.

- 5) Ideally, the EffectiveDate element should be the first tag in a transaction.
- 6) When using the required element/tag, ensure nonexistent fields are not listed.
- 7) Hidden fields that are independent of other fields should be placed at the end of the fields section.
Exception: If the value of a hidden field is needed in a subsequent field. Also the hidden tag should be placed after the disabled tag.
- 8) PlanDefinitions: In general, use PlanDefinitions any time a particular fact, amount, threshold value, etc. is tied to a product (plan), and is used in a constant fashion across multiple transactions and/or calculations. It should be defined in PlanDefinitions rather than defined repeatedly in a transaction or 'calcGeneral' XML.
- 9) Do not attach a ReportFileTypes business rule to policy-financial transactions.
- 10) Ensure new transactions are defined with the correct rule type and name. For example, don't define Policy-Financial activities as Policy-Document activities. Names should have no spaces and should be camel case.
- 11) Use Data Dictionary defined field names for consistency across products.
- 12) CopyBook naming standard: <CopyBook> CopyBook-CopyBookName</Copybook>
Note: The naming convention uses Hungarian format or camel case format.
- 13) When trying to retrieve CashValue from PolicyValues use Type="FIELD">Valuation:Policy:CashValue.
- 14) When using Calculated and Query tags with a Field, the calculated tag should come after the query tag.

- 15) DefaultValue tag should be placed between DataType and Query Tags in a Field.
- 16) In an IIF EXPRESSION and in a Spawn IF, when using the word 'AND', use And, not AND.
- 17) XPATHs should always use the fully qualified path.

Ex. /Valuation/Policy/FundList/Fund[FundGUID='\$\$\$LoanCreditFundGUID1\$\$\$']/CashValue not
//FundList/Fund[FundGUID='\$\$\$LoanCreditFundGUID1\$\$\$']/CashValue.

- 18) Suggestions for variable and field names.
 - a) Capitalize the first letter of each word except for use of upper case in SQL reserved words and desired case in validation messages that will appear to the user. Example: DateOfDeath not DateofDeath
 - b) No spaces or underscores. Ex. DateOfBirth not Date_of_Birth
 - c) Do not abbreviate. Ex. PolicyAnniversary not PolAnniv.

SQL (Structured Query Language)

- 1) When performing SQL to select one or more AsActivity records, be sure to consider AsActivity.StatusCode and AsActivity.TypeCode values to include/exclude records as desired (e.g. to avoid returning pending, shadow, undo, redo, etc. records).
- 2) Consider whether the query could return one or more than one result and deal with it appropriately. Functions can be used to ensure one record is returned if appropriate (e.g. SUM, MIN, MAX, etc.).
- 3) Where possible, subqueries should be avoided in favor of JOINS for better performance.
- 4) When querying for the most recent transaction of a certain type: Order by EffectiveDate DESC, ProcessingOrder DESC, ActivityGMT DESC and then FETCH FIRST 1 ROW ONLY. (Also see #1 above).
- 5) If a piece of information is already defined as a field, don't use an SQL to retrieve this information again from the database. Since this value is already a <Field>, change to TYPE="FIELD" and reference the fieldname.
- 6) String (text) Variables in SQL statements should have brackets and then single quotes around them. Integer variables should have brackets around them.
- 7) Use SQL queries calling ASCODE rather than redefining as fixed queries in business rules where possible (for standardization and ease of maintenance).

- 8) Don't join in tables unnecessarily in SQL. For example, AsActivity contains PolicyGUID, therefore there is no need to JOIN AsPolicy to retrieve PolicyGUID.
- 9) Capitalize SQL keywords.
- 10) SQL should return one value for MathVariables.
- 11) SQL should return two columns and multiple rows for combo boxes.
- 12) Use SQL sparingly. Use collections to bring back as much useable data as possible in one round trip to the database. Use CollectionValue to parse the individual pieces of data from the Collection.
- 13) Use SQL to retrieve necessary data from the database, not to use SQL to manipulate dates or use the SQL case statement.

Configuration Syntax

- 1) Do not use XML 'shortcut element syntax'.
Ex. <EffectiveDate/>. Each opening tag should have a closing tag. <EffectiveDate></EffectiveDate>
- 2) When trying to use Issue State within a transaction, do not use a SQL query. Instead define a variable with TYPE="FIELD" and use Policy:IssueStateCode for efficiency.
- 3) Do not include <Hidden> tags if a field is not hidden.
- 4) Do not include <Disabled> tags if a field is not disabled.
- 5) Quotes (") aren't needed in a MathIF when comparing numbers, eg, ActiveSegmentCode=01.
- 6) Use <!-- TBD.... --> for comments indicating future work is to be done.
- 7) For the Validations within a transaction/business rule, the logic follows javascripting, (ie, reverse logic. Validate Expressions simply uses MathVariables/Fields within the transaction/business rule, not javascripting, ie, not reverse logic).
- 8) Use Date as <DataType> for date fields.
- 9) Use ".valueOf()" at the end of date parameters in JavaScript,
Ex. new Date(document.frmAsActivityDetail.txtActivityIssueDate.value).valueOf().

- 10) Use "parseFloat()" in JavaScript to ensure numeric values in text fields are treated as decimal/float values,
Ex. parseFloat(document.frmAsActivityDetail.txtLoanExists.value) > 0.
- 11) The effective date of spawned transactions, when that spawned activity is only ever system generated, should be "Disabled".
- 12) Don't define MathVariables that aren't used anywhere.
- 13) Spawn fields should be in the same sequence as defined in the spawned transaction definition.
- 14) Any GUID fields passed as spawn fields should be disabled in the spawned transaction.
- 15) A MathVariable should define its VARIABLENAME and TYPE attributes prior to any other attribute (i.e. <MathVariable VARIABLENAME="Example" TYPE="SQL" DEFAULT ="01">...)
- 16) Validations: When consecutive words are displayed in the message as bold, do not specify markup for each word. Instead provide one markup for the phrase.
- 17) Only use ROUND when there is a reason.
- 18) Use same icons for like transactions across products (in TransactionCosmetics).
- 19) Filler: Display and DataType on all Filler fields should be Blank.
- 20) Spawned transactions should use SPAWNCODE="01" when the spawned transaction is effective on the same day as the spawning transaction. Use SPAWNCODE="03" when the spawned transaction is effective on a different or calculated date. The FIELD attribute is necessary with SPAWNCODE="03".
- 21) Consolidate logic (e.g. expressions) into a single MathVariable where appropriate (i.e. where an intermediate calculation value is not needed – don't break the calculation into many steps).
- 22) When to use MathIF versus TYPE="IIF": Use MathIF when the value(s) of one or more MathVariables is conditional. That is – if the condition is not met, the MathVariables do not need to be defined. MathIF also helps avoid multiple IIF's when the desired result is one value. Use IIF if this specific MathVariable always needs to be evaluated but the resulting value is conditional between two values.
- 23) In defining a <Field> the <DefaultValue> tag (if used) should come after <DataType>.
- 24) When Withholding is needed, use the Withholding tags (<Withholding>Activity</Withholding>) instead of creating fields for withholding.

- 25) DataTypes in spawnfields must match the data type in receiving fields.
- 26) Dates can not be directly compared in ValidateExpressions and MathVariables. They can be directly compared in JavaScript.
- 27) Combo Boxes and Radio Buttons:
 - a) When using a combo box, start with a 00 value followed by 01, 02, etc, incrementing by one. Always use the two digit format (00, 01, 02, 03, etc).
 - b) The 00 value should always stand for a not applicable value or for a “*Select an Option*” value in a combo box. It will 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 or one whose entry is non-required should not have a “00” selection.
 - c) Radio Buttons and Combo boxes should have a “00”value for the negative response and 01 for the affirmative response.