Using calculated fields in data blocks

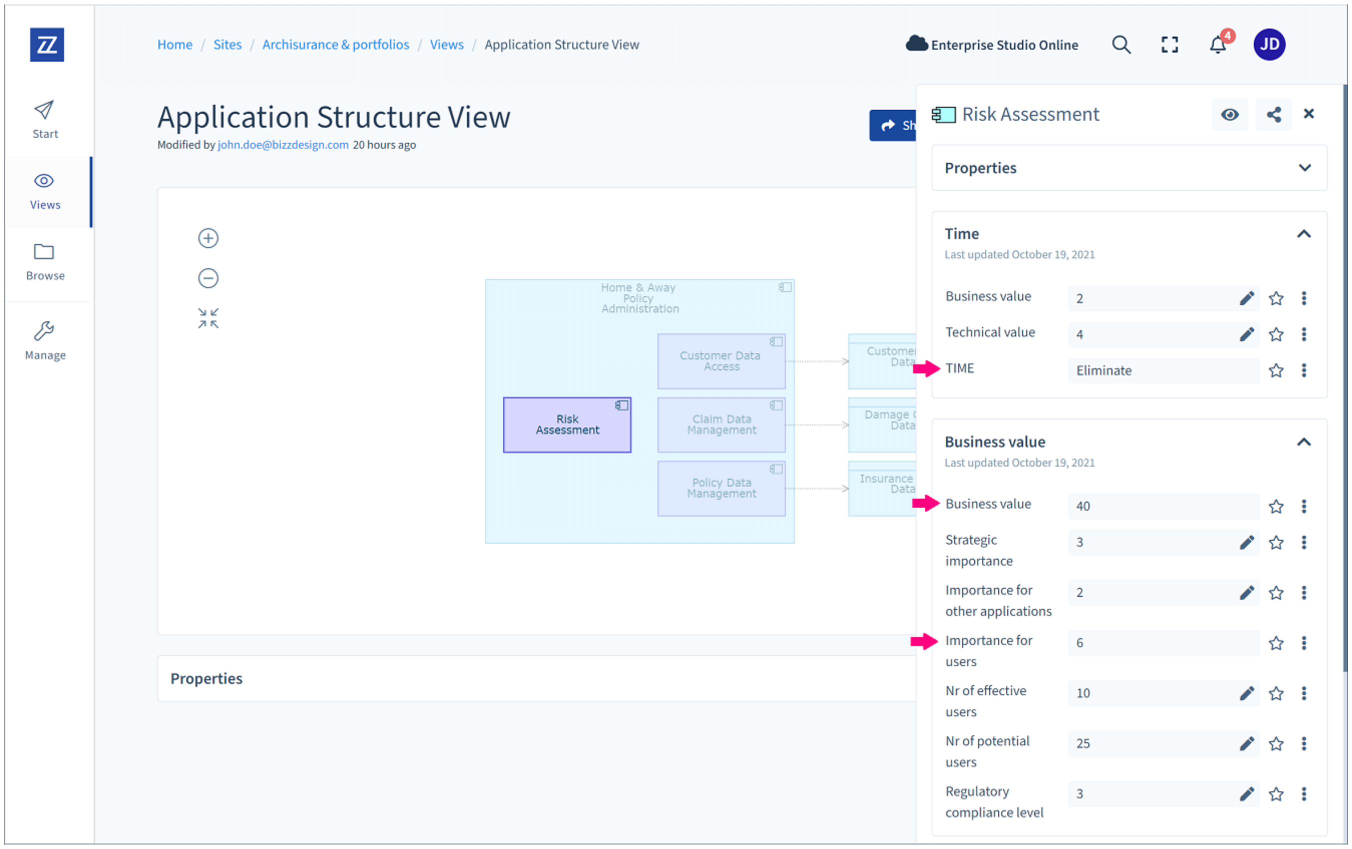
Attribute fields in data blocks can be of different types, for example, text, date, or enumeration (closed list of values). These fields can be edited in Horizzon by entering values. In addition to these editable fields, it is possible to define attribute fields as calculated fields.

For defining calculated fields it is assumed that you have some familiarity with scripting.

On this page:

What is a calculated field?

A calculated field is a read-only field in a data block in Horizzon. Its value is automatically calculated based on values from other fields within the same data block. These other fields can be regular fields in which you can enter a value, but they can also be calculated fields in itself, getting their value from other fields within the data block. The calculation is based on an expression that specifies which other fields in the data block are used to aggregate the calculated value, which operations and functions are performed, and any needed conversion of values. For examples, please refer to Examples of calculated fields in data blocks.



Home / Sites / Archisurance & portfolios / Views / Application Structure View
Enterprise Studio Online
JD
Start
Views
Browse
Manage
Application Structure View
Modified by john.doe@bizzdesign.com 20 hours ago
Home & Away Policy Administration
Customer Data Access
Risk Assessment
Claim Data Management
Policy Data Management
Properties
Risk Assessment
Properties

Time
Last updated October 19, 2021

Business value	2
Technical value	4
TIME	Eliminate

Business value
Last updated October 19, 2021

Business value	40
Strategic importance	3
Importance for other applications	2
Importance for users	6
Nr of effective users	10
Nr of potential users	25
Regulatory compliance level	3

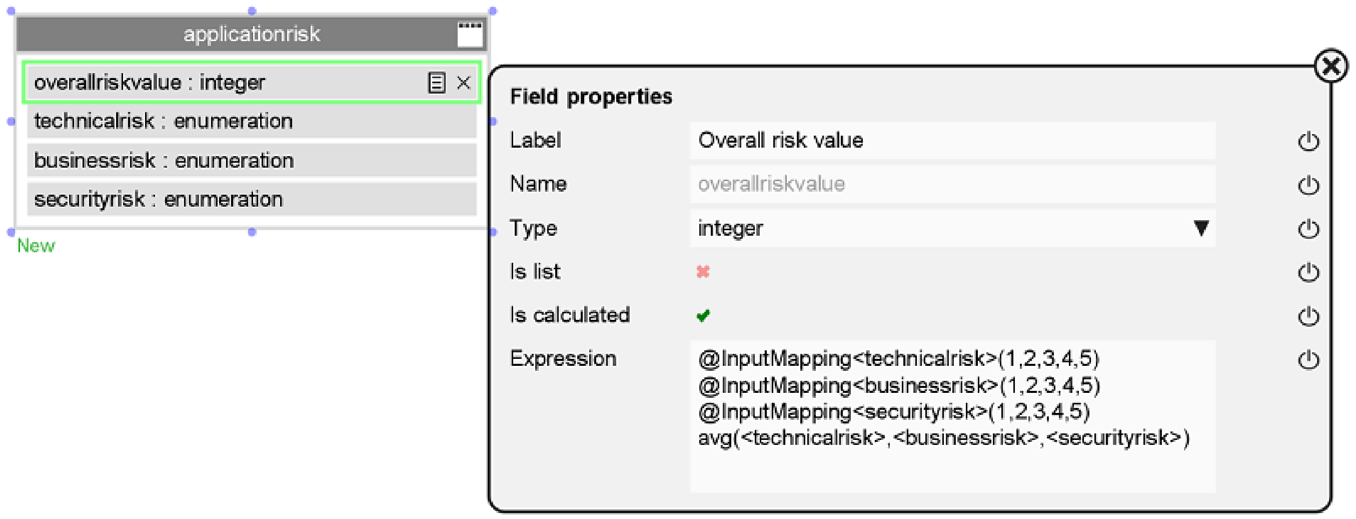
Examples of data blocks with calculated fields

Why using calculated fields?

Calculated fields can be useful for combining numerical and enumeration values to get a "total" score, or for mapping scores to a value. The latter is similar to aggregate metrics. Calculated fields could eventually replace aggregate metrics.

Defining a calculated field

Calculated fields are defined in the data block definition. One ore more attribute fields in the definition can be set as calculated field. Only attribute fields of type real, Boolean, money, and enumeration can be a calculated field.


applicationrisk
overallriskvalue : integer
technicalrisk : enumeration
businessrisk : enumeration
securityrisk : enumeration
New

Field properties

Label	Overall risk value
Name	overallriskvalue
Type	integer
Is list	✗
Is calculated	✓
Expression	@InputMapping<technicalrisk>(1,2,3,4,5) @InputMapping<businessrisk>(1,2,3,4,5) @InputMapping<securityrisk>(1,2,3,4,5) avg(<technicalrisk>,<businessrisk>,<securityrisk>)

Example of a calculated field in the data block definition

Enabling calculation

To define an attribute field as calculated field, the **Is calculated** setting at the field's properties needs to be enabled.

Defining the expression

Next, an expression needs to be defined that specifies the details of the calculation, containing the following:

- The calculation.
- Value conversions, if needed. Conversion of values is expressed in mappings. Conversion may not be always needed. It depends on the data type of the calculated field and that of the attribute fields used as input.

Structure of the expression

The expression of a calculated field always contains a calculation. As mentioned before, input and/or output mappings are only used if needed. The calculation and mappings have a fixed order in the expression. Mappings are always preceding the calculation, and if both output and input mappings are used, the input mappings need to precede the output mapping.

An expression can have multiple input mappings. When using multiple enumeration fields, an input mapping needs to be defined for each unique field. This allows you to specify a different mapping for different fields, even if these fields are of the same enumeration type. An expression can have only one output mapping.

When all components are used, the general structure of an expression would look as follows:

```
@InputMapping 1
@InputMapping 2
@InputMapping x
@OutputMapping
Calculation
```

An example of an expression:

```
@InputMapping<architecturalfit>(0,5,10)
@InputMapping<interoperability>(0,2.5,5,7.5,10)
@InputMapping<maintainability>(2,4,10,50,100)
@InputMapping<scalability>(0,2.5,5,7.5,10)
@OutputMapping(2,4,6,8)
avg(<agility>,4*<architecturalfit>,4*<availability>,-1.0*<complexity>,<interoperability>,4*<maintainability>,
3*<resilience>,<scalability>,<stability>)
```

In the example mappings and calculation shown below, you will see that sometimes spaces are used in the expression and sometimes not. It is optional. You can choose to use spaces for readability, but it is not required and it does not affect the functionality.

Input mapping

An expression can only handle numbers. When an enumeration field is used as input, its values (literals) need to be converted into numbers before they can be used.

For example, for conversion of an attribute field "risk" of type enumeration with a fixed set of values "low", "medium", and "high", an input mapping is needed to convert them into numbers:

```
@InputMapping<risk>(2,4,6)
```

If the enumeration is a list of random values, the literals must be explicitly named in the expression. Regardless of the order of the values in the enumeration, the calculation will arrive at the same result. Example:

```
@Inputmapping<region>(europe:0.13, asia:0.57, americas:0.43)
```

An input mapping must have exactly one number for each value in the enumeration. The numbers may be integer or real.

Output mapping

An output mapping is needed if the calculated field itself is of type enumeration. The result of the calculation (a number) needs to be mapped back to an enumeration value that fits the type of the calculated field.

The numbers in the mapping represent the thresholds (inclusive) between the enumeration's values: whenever the expression result value lies below a threshold number, the previous enumeration value applies. When it is equal or above, the next value applies. It means that the mapping must specify N-1 numbers for an enumeration that has N values.

For example, when the enumeration type of the calculated field has three values "low", "medium", and "high", the output mapping may be defined as shown below. In this case, the calculated field will show "low" if the expression result is below 3, "medium" when it is 3 or higher (but below 7), and "high" when it is 7 or higher.

```
@OutputMapping(3,7)
```

The order of enumeration values in the calculated field can also be subject to regular change. For example, when the enumeration does not represent an ordered concept (like "level"), but instead enumerates independent items that still inhabit the same number space (for example, "colors"). To avoid errors when the expression of the calculated field is not changed accordingly, an output mapping can be defined using explicitly mentioned values:

```
@OutputMapping(blue,13:green,21:red)
```

In the above example, the enumeration type of the calculated field has three values "blue", "green", and "red". With the output mapping defined as above, the calculated field will show "blue" if the expression result is below 13, "green" when it is 13 or higher (but below 21), and "red" when it is 21 or higher.

Calculation

This is where the operations and functions are specified. Allowed operations are addition, subtraction, multiplication, and division. In addition, there are a number of predefined functions that can be used, like max, prod, and sqrt. A few examples of calculations:

```
<licencecosts> + <maintenancecosts> + <operationalcosts>
prod(<incidents>,<severity>)
3.0*<strategicimportance> + 2.0*<importanceforotherapplications>
avg(<nrofeffectiveusers>, 0.1*<nrofpotentialusers>)
```

The names between the angled brackets are the names of attribute fields in the data block.

For a complete overview of the allowed operations and functions, please refer to Operations and functions for calculated fields in data blocks.

Related articles

- Examples of calculated fields in data blocks
- Using calculated fields in data blocks
- Operations and functions for calculated fields in data blocks

Introductory eLearning course

Check out the free Enterprise Studio introductory eLearning course to learn about data blocks.

View course