



Developer's Guide

Acumen Fuse[®] Software

www.projectacumen.com

Last Revised: March 2011

1 INTRODUCTION	4
2 INTRODUCING METRICS IN ACUMEN FUSE	4
2.1 Overview – What is a Metric?	4
2.2 What do Metrics apply to?	4
<i>Figure 2-1 Spreadsheet Example of Project Data</i>	4
<i>Figure 2-2 View of Ribbon and Phase Analysis in Acumen Fuse</i>	5
<i>Table 2-3 – Acumen Fuse Analysis Modes Summary</i>	5
2.2.1 Grouping by Ribbons	5
2.2.2 Grouping by Phases	5
2.2.3 Grouping by Intersections	5
2.3 Types of Acumen Fuse Fields	6
2.3.1 Activity Fields	6
2.3.2 Project Fields	6
2.3.3 Workbook Fields	7
2.3.4 Dynamic Fields	7
2.4 Metric Formulas	7
2.4.1 Formula Syntax	7
2.4.2 Array Formulas	7
<i>Table 2-4 – Ribbon and Phase Calculation of Metrics</i>	8
2.4.3 Formula Types	8
3 METRIC FORMULA DEVELOPMENT	9
3.1 Fuse 2.0 Metric Editor	9
<i>Figure 3-1 The Fuse 2.0 Metric Editor</i>	9
<i>Figure 3-2 Hierarchical Development of Metrics</i>	10
3.2 Writing Formulas	10
3.3 Single Function Formulas	10
3.3.1 Step 1 – Start with the Base Formula	11
3.3.2 Step 2 – Adding Conditions	11
3.3.3 Step 3– Returning Field Values Rather Than Counts	12
3.4 Compound Formulas – AND Conditions	14
3.4.1 Count-based Compound AND Formulas	14
<i>Table 3-2 – Table of Multiple Condition Outcomes with AND Syntax</i>	14
3.4.2 Value-based Compound AND Formulas	15
3.5 Compound Formula – OR Conditions	15
3.5.1 Count-based Compound OR Formulas	15
<i>Table 3-3 – Table of Multiple Condition Outcomes with OR Syntax</i>	16
3.6 Compound Formula – Using AND and OR Together	16
3.7 Other Math Functions	17
4 WRITING THE PRIMARY METRIC FORMULA	18
<i>Figure 4.1 Basic and Advanced Primary Formula Definition</i>	18
5 WRITING THE SECONDARY METRIC FORMULA	18
<i>Figure 5.1 Simple Percentage of Primary Formula and Advanced Secondary Formula Definition</i>	19
5.1 Advanced Percentage Examples	19
5.2 Ratio Example	20
6 WRITING THE TRIPWIRE METRIC FORMULA	21
<i>Figure 6 Auto Calculated and Advanced Tripwire Formula Definition</i>	21

6.1	Cost Example	22
6.2	Schedule Example.....	22
6.3	Weighting.....	22
7	DEFINING THE THRESHOLDS	23
7.1	Defining Tripwire Threshold Scales.....	23
	Figure 6-1 Tripwire Threshold Scales.....	23
7.2	Normal and Gradient Scales	23
7.3	Tripwire Thresholds.....	24
	Figure 6-3 Setting Ranges for Tripwire Thresholds.....	24
7.4	Including/Excluding Metrics from Analysis	25
	Figure 6-4 Inclusion/Exclusion of Metrics to Analyzers	25
8	SHORTCUTS AND RULES	26
8.1	Counting Shorthand with One Criterion.....	26
8.2	Counting Shorthand with Multiple Criteria	26
8.3	AND Shorthand with Multiple Criteria (Tripwire formulas)	26
9	DEFINITIONS	27
9.1	Tripwire Threshold.....	27
9.2	Primary Metric Formula	27
9.3	Secondary Metric Formula	27
9.4	Tripwire Formula.....	27
9.5	Ribbons	27
9.6	Phases.....	27
9.7	Intersections	27
10	COMMONLY USED SYNTAX	28
10.1	IF(logical_test, value_if_true, [value_if_false])	28
10.2	SUM(number1, [number2], [number3], [number4], ...)	28
10.3	AND(logical1, [logical2], ...)	28
10.4	MAX(number1,number2,...).....	29
10.5	AVERAGE(number1, [number2],...)	29
10.6	COUNTIF(range, criteria)	29

Introduction

This document is a reference and guide for Fuse users looking to develop and customize advanced metrics within Acumen Fuse.

INTRODUCING METRICS IN ACUMEN FUSE

Overview – What is a Metric?

Metrics are used to evaluate criteria and then to determine if these criteria are met or not. This gives insight into how well a project is planned and executed.

Metrics contain formulas, weightings and tripwires (thresholds). Formulas are used to calculate results as part of an analysis. Tripwire thresholds are used to flag and filter activities that exceed given levels.

Metric results can be numeric (e.g. cost or duration) or percentages (e.g. percentage of total project duration). Percentages are useful for portraying results within a given context. Acumen Fuse uses metric libraries to group metrics for project analysis. Standard metric libraries pertaining to schedule quality, cost, project performance, risk exposure, Earned Value and more are included within the tool. For example, metrics can be organized by categories, project attributes, or along a project lifecycle. Organizing the metrics differently produces allows you to customize your application to your project or program. It is important to understand how the metrics are constructed, calculated, and applied.

What do Metrics apply to?

All project data is stored in a tabular manner with each activity represented as a row of data. For that activity, each activity attribute is listed in a different column.

	A	B	C	D	E
1		Attributes			
2		Critical	Start Date	Remaining Cost	
3	Activity 1	Yes	3/5/2010	0	
4	Activity 2	No	4/20/2010	0	
5	Activity 3	No	6/15/2009	1200	
6					
7					
8					

Figure 2-1 Spreadsheet Example of Project Data

The data is then aggregated either across the spreadsheet or down the spreadsheet. Or, as viewed in Acumen Fuse, the analysis can be conducted for Ribbons or Phases or Intersection points. Acumen Fuse metrics are:

- *Calculated across a Ribbon (green box)*
- *Calculated down a Phase (blue box)*
- *Calculated for the intersection of a phase/ribbon (red box)*

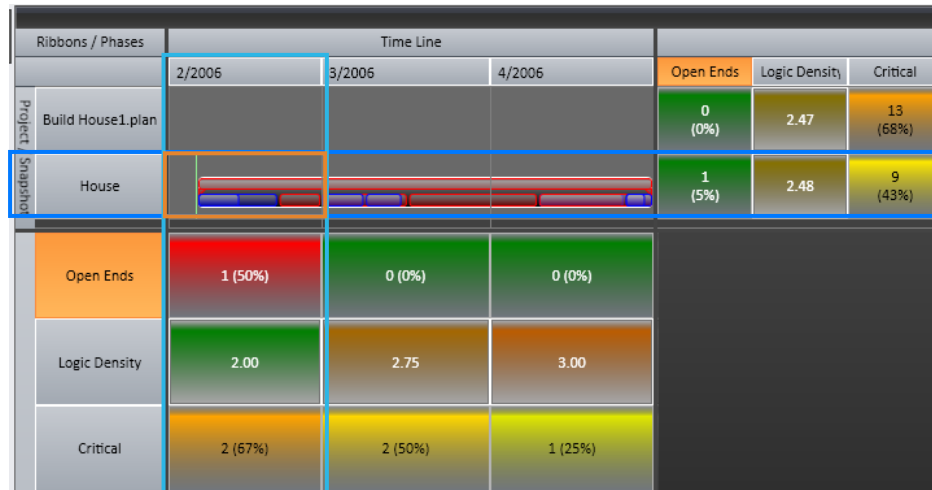


Figure 2-2 View of Ribbon and Phase Analysis in Acumen Fuse

A summary table of the analyses is shown below. Additional details are included after the table.

Analysis Type	Summary	Details
Ribbon Analysis	Cross ribbon comparison	Metric analysis by ribbon enabling cross-ribbon comparisons to be drawn.
Phase Analysis	Cross phase comparison	Metric analysis by phase enabling cross-phase comparisons to be drawn. Trending can also be carried out.
Intersection Analysis	Specific ribbon/phase analysis	Metric analysis for a specific ribbon/phase intersection enabling pinpointing of project hot spots and problem areas.

Table 2-3 – Acumen Fuse Analysis Modes Summary

Grouping by Ribbons

Ribbons are groupings of activities based on a given criteria. By default, ribbons are grouped by project but can be grouped in multiple ways including activity attribute and path. If a workbook contains multiple projects, then a separate ribbon for each of these projects will be shown. Ribbons also contain activities. These activities can be hidden from within a ribbon if desired. Critical activities are shaded in red; non-critical in blue. Normal activities sit beneath summaries and milestones within a ribbon. Ribbons are segmented by phases.

Grouping by Phases

Phases are user definable 'segments' of time against which the Acumen Fuse analysis is run. Phases can be days, weeks, months, quarters, years, custom periods or the entire project duration.

Grouping by Intersections

Intersections are where a ribbon and a phase intersect.

Along with understanding the grouping of the data, it is also important to understand the variety of fields that are referenced within Acumen Fuse.

Types of Acumen Fuse Fields

When creating metric formulas, there are four types of field that can be referenced:

- Activity Fields
- Project Fields
- Workbook Fields
- Dynamic Fields

Metric formulas are generally written with the context of an activity. However, if you reference fields outside of the activity context (e.g. project and workbook), you are able to model how activities relate to, and potentially impact, other contexts such as project and workbook.

Activity Fields

Activity fields are the most commonly used type of field in an Acumen Fuse metric formula. All fields that are defined in the field mapping during a project import are exposed as activity fields in the metric editor. These include user defined and code fields.

Project Fields

Some project level fields get automatically imported during a project import. These fields are automatically exposed and can be used within metric formulas. When a metric is calculated that contains a project field reference, the specific project field value for the activity in question is used. A single metric calculation may contain activities from multiple projects. In this instance, the appropriate project level field value will be used for each activity (e.g. “time now” may be different for each of the projects).

Project fields include:

- *Project Start [ProjectStart]*
- *Project Finish [ProjectFinish]*
- *Project Time Now [ProjectTimeNow]*

Workbook Fields

Workbook fields are summated values that are calculated at the workbook level (that take into account all activities within the workbook).

Workbook fields include:

- *Workbook Cost (total) [WorkbookCost]*
- *Workbook Actual Cost [WorkbookActualCost]*
- *Workbook Remaining Cost [WorkbookRemainingCost]*
- *Workbook Budget Cost [WorkbookBudgetCost]*
- *Workbook Budget Duration [WorkbookBudgetDuration]*
- *Workbook Actual Duration [WorkbookActualDuration]*
- *Workbook Remaining Duration [WorkbookRemainingDuration]*
- *Workbook Duration (total) [WorkbookDuration]*
- *Workbook # of Activities [WorkbookNumberofactivities]*

Dynamic Fields

Dynamic fields have different values depending on the context within which they are being used within an analysis. “Period Start” and “Period End” are both dynamic fields. When Period Start and Period Finish are being applied to a phase analysis, Period Start and Period Finish relate to the start and finish of the phase in question. When being used within the context of a ribbon, Period Start and Period End relate to the start and end date of the ribbon.

- *Period Start: [_PeriodStart]*
- *Period Finish: [_PeriodFinish]*

In the context of Ribbons – the earliest start of the first activity in the Ribbon (time independent) is Period Start. Similarly, the end of the last activity in the Ribbon is Period Finish.

In the context of the Phase, Period Start is the start of a Phase and Period End is the end of the Phase.

Metric Formulas

Formula Syntax

Fuse metrics are defined using standard MS Excel syntax/scripting language. Formulas can be built within Fuse either through the freeform formula editor or by selecting functions/fields from the various menus within the metric editor. The Check Formula feature ensures correct syntax during metric development.

Array Formulas

Acumen Fuse metric formulas are based on what is known as “Single Value Result Array formulas”. Single Value Result Array formulas work with a series of data (activities), aggregate it (typically using the likes of SUM, AVERAGE or COUNT) and return a single value to the (ribbon, phase or intersection) analyzer.

Array formulas typically return a series of values. For example, in Excel, the formula =Row(A1:A5) returns only a single value (the first value in the list). Instead, an array formula will return all values for A1 to A5. Against the results of an array formula, you typically apply a container function such as SUM or AVERAGE or COUNT. These functions enable you to apply the function to the list of values and return a single value result.

Relating this back to Acumen Fuse, a ribbon, phase and intersection all contain one or more activities. In the case of phases and intersections, the activities may span across more than one phase or intersection and so certain data (duration, work and cost field types) gets pro-rated. When metric functions are applied during an Acumen Fuse analysis, they are applied to the ribbon, phase or intersection indirectly being applied to all activities within that segment through the use of an array formula.

Ribbons / Phases	Time Line		Ribbon Analyzer
Orig. Dur.	11/15/2009	11/22/2009	Orig. Dur.
A	5 (28%)	4 (19%)	9 (47%)
B	5 (26%)	5 (26%)	10 (53%)
Phase Analyzer	10 (54%)	9 (46%)	

Array Formula Original Duration = 5 + 4 = 9

Array formula Original Duration = 5 + 5 = 10

Table 2-4 – Ribbon and Phase Calculation of Metrics

Formula Types

There are three formulas in Acumen Fuse and they all use MS Excel Array formula syntax.

- *Primary formula*
- *Secondary formula*
- *Tripwire formula*

These are discussed in greater detail in Chapters 4-6 of this guide.

METRIC FORMULA DEVELOPMENT

Fuse 2.0 Metric Editor

Version 2.0 of Fuse saw the introduction of a new metric editor. This metric editor provides a means of simplifying the development of metrics by reducing the amount of times a metric formula needs to be manually written.

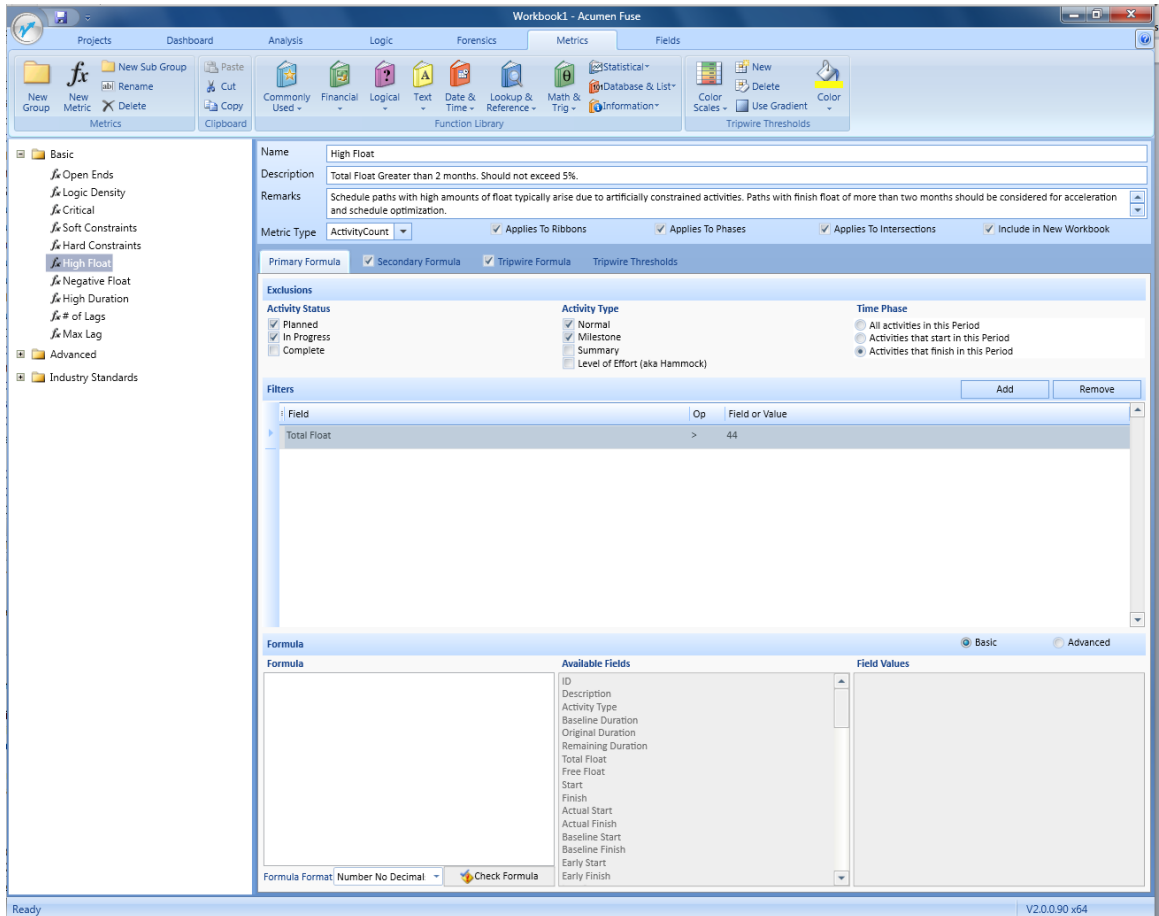


Figure 3-1 The Fuse 2.0 Metric Editor

Each metric contains three formulas (primary, secondary and tripwire). Each of these three formulas can be built in Fuse 2.0 and beyond using a three level hierarchy:

- Exclusions: a top level set of filters to exclude specific activities based on type, status and time period
- Filters: standard filters that further pinpoint specific activities
- Formula: advanced custom formulas to further specify advanced criteria sets.

Exclusions, filters and formula are hierarchical.

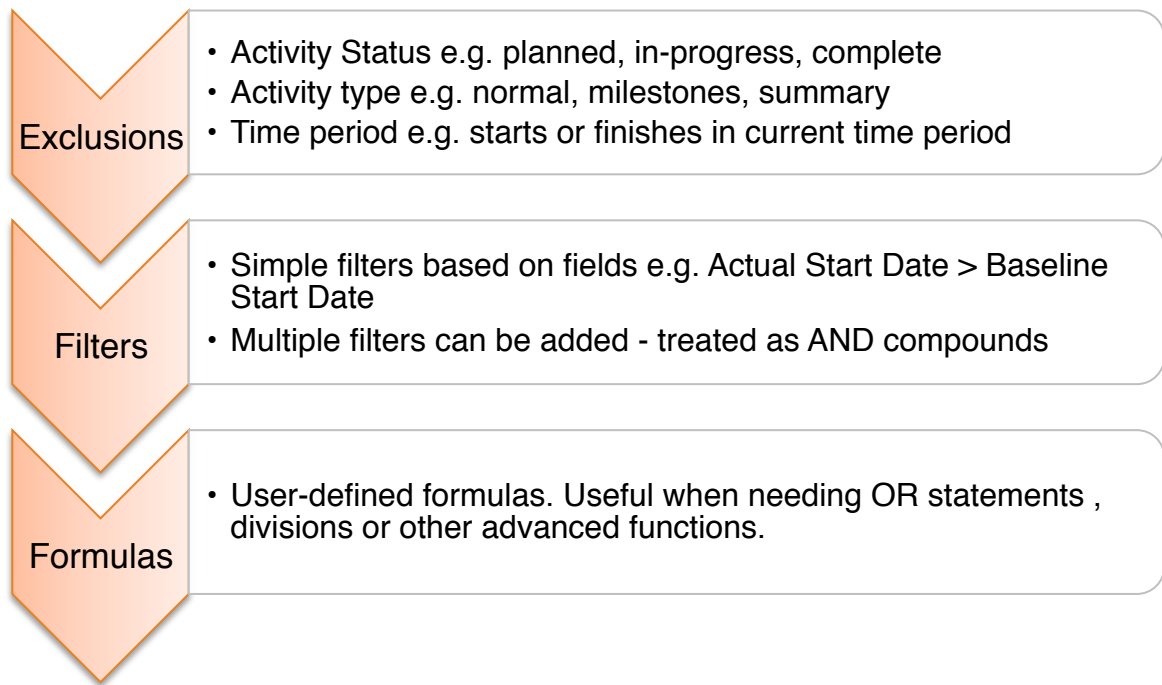


Figure 3-2 Hierarchical Development of Metrics

Writing Formulas

Acumen Fuse metrics can be developed using either a basic or advanced approach (or a combination of the two):

- *Basic:* essentially a filter-based set of metrics that don't require detailed formula definition
- *Advanced:* detailed formulas used to define a metric beyond a simple filter

This developer guide focuses on the syntax required to develop advanced metric formulas in Fuse.

The easiest way to start creating advanced formulas is to first modify an existing formula. The syntax for all formulas, new or modified, follows a standard format of:

Function(fieldvalue)

Common functions include COUNT, IF, COUNTIF, SUM, etc. and can be used in combination. If conditions are used, then the syntax is modified and follows the format of:

Value1 – if condition is met
 Value 2 – if condition is not
 Function(fieldvalue, {condition}, {value1}, {value2})

This chapter will start with working with single formulas and then progress to including multiple formulas in combinations.

Single Function Formulas

In this example, a metric formula is developed that will show the total remaining cost of the project for the activities whose remaining cost is greater than \$1,000. Since Fuse calculates the remaining cost for each activity in the project, a base formula for calculating remaining cost will be used. Then the formula will be modified using a series of conditions. The final

formula will sum all of the activity remaining costs, if they are greater than \$1,000, and present a total.

Step 1 – Start with the Base Formula

Objective: Create a formula for calculating the total remaining cost of the project.

The formula for calculating total remaining cost is:

SUM(RemainingCost)

This formula has no conditions and returns the sum of the remaining costs for each activity.

Step 2 – Adding Conditions

There are several ways to modify the initial formula with conditions. The first way is to modify the formula to return a count or a sum based on a condition.

Objective: Give a sum of the Number of Normal Activities whose Remaining Cost > \$1,000.

While the conditional formula in a MS Excel worksheet cell would be COUNTIF(RemainingCost,">1000"), the equivalent function for MS Excel array syntax is slightly different. In array syntax, this formula is:

Function(fieldvalue,value1,value2)

So the base formula must be modified to include the criteria of only counting the activity when its Remaining Cost is greater than \$1,000.

The conditional syntax is written as:

RemainingCost>1000,1,0

Where the 1,0 are important because they tell the formula what to return as the result. The formula will return a '1' if the condition is met, and a '0' if the condition is not met. In absence of these return values, an IF statement will return a Boolean true/false.

Finally, the SUM function is added on the outside of the conditional function to create the entire formula:

SUM(IF(RemainingCost>1000,1,0))

Additionally, if the Acumen Fuse metric returns a 'count' – that is, the formula is asking for a summation or a number of something where it counts, then the formula can be written in shorthand. In a counting or summing formula, the return of a '1' if the condition is met and a '0' if the condition is not met, is implied through the default return of true/false.

Many of the standard Acumen Fuse metrics included in the tool are written in shorthand. It is important to be able to recognize when a formula is written in shorthand. This cuts down on syntax. More information about shorthand formulas can be found in Chapter 8 of this manual. Therefore,

If the formula is returning a count (and not actual values), then the formula can be written in shorthand and the IF, 1 and the 0, along with the extra commas can be dropped in exchange for a single function called COUNTIF.

The original formula:

SUM(IF(RemainingCost>1000,1,0))

The resulting shortcut formula is:

COUNTIF(RemainingCost,">1000")

Note: the syntax for "SUM(IF" is very different to that of "COUNTIF"

Step 3– Returning Field Values Rather Than Counts

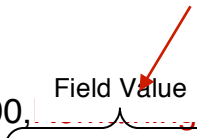
In addition to having counts returned as the result for a metric formula, actual field values can also be returned and then either reported individually or summed. Values may include, for example, the total costs of completed activities or expected durations of planned activities. This is the final modification of the metric formula for a summing all of the activity remaining costs, if they are greater than \$1,000, and presenting a total.

Objective: Sum the Remaining Cost for all Normal Activities whose Remaining Cost > \$1,000

Start with the original FULL formula.

SUM(IF(RemainingCost>1000,1,0))

To summarize the remaining costs vs. having a count, the formula becomes:

SUM(IF(RemainingCost>1000,,0))

This formula will return the 'Remaining Cost' of each activity where the condition is met – in this case Remaining Cost > \$1000. A '0' will be returned if the condition is not met.

Since this is not a counting formula, there is no shorthand way to write the formula.

So the final formula for showing a sum of all of the remaining costs where those individual activity remaining costs are greater than \$1,000 is:

SUM(IF(RemainingCost>1000,RemainingCost,0))

Compound Formulas – AND Conditions

Frequently, more than one condition is needed in a metric formula to return the desired result. When these multiple conditions must be met in order to return an answer, it is called a compound formula. One type of compound formula is an AND formula –i.e. all of the conditions must be met to return an answer.

Standard syntax for an AND statement:

`SUM(IF(Return_from_Condition1*Return_from_Condition2>0,1,0))`

This formula is stating that if the sum of the return from Condition1 and 2 is greater than 0 then return a 1.

This could also be shorthanded as:

`SUM(Return_from_Condition1*Return_from_Condition2,">0")`

Count-based Compound AND Formulas

As an example, we want to know the number of normal activities whose remaining cost is greater than \$1,000.

Objective: Count the Number of Normal Activities whose Remaining Cost > \$1,000

Start with the original full formula (no shorthand):

`SUM(IF(RemainingCost>1000,1,0))`

In this example, only Normal activities are wanted. The formula for normal activities would be:

`SUM(IF(ActivityType="Normal",1,0))`

The syntax for AND functions is created by placing a “*” between the two conditions. Then BOTH conditions must be met for the statement to be true and a count to be returned.

To write this into the formula, a second IF condition is created and ‘**multiplied**’ by the first condition. The IF statements are placed inside brackets so that the number of activities meeting both criteria are summed.

`SUM(IF(RemainingCost>1000,1,0)*` Insert the new condition. `,`

By inserting the additional condition inside the parentheses, the modified formula is:

`SUM(IF(RemainingCost>1000,1,0)*IF(ActivityType="Normal",1,0))`

This can also be written in shorthand as:

`SUM((RemainingCost>1000)*(ActivityType="Normal"))`

The following table shows the possible outcomes of this multiple condition formula.

Remaining Cost > 0	Activity Type = Normal	Return
No	No	0
No	Yes	0
Yes	No	0
Yes	Yes	1

Table 3-2 – Table of Multiple Condition Outcomes with AND Syntax

Only when BOTH conditions are met does the formula return a value of 1.

Value-based Compound AND Formulas

In the previous example we were counting results. In this example, the same criteria are applied but instead of counting results we are going to sum up the values (remaining cost) of those activities that pass the compound criteria test.

Objective: Remaining Cost Total for Normal Activities whose Remaining Cost > \$1,000

The original count formula is used as the basis:

```
SUM(IF(RemainingCost>1000,1,0)*IF(ActivityType="Normal",1,0))
```

The field name is then substituted for the '1' – changing the formula from a count-formula to one that brings back the remaining cost. The resulting formula is:

```
SUM(IF(RemainingCost>1000,RemainingCost,0)*IF(ActivityType="Normal",1,0))
```

Instead of a '1' that would return a count, the field value name is

This formula had can be written in shorthand as well – but, only for the count side of the formula.

The shorthand formula becomes:

```
SUM(IF(RemainingCost>1000,RemainingCost,0)*(ActivityType="Normal"))
```

Compound Formula – OR Conditions

Another type of compound formula is an OR formula – one condition or another condition must be met in order to return an answer. For example, in checking project logic, it is helpful to know how many activities are there that have no Predecessors OR no Successors. By placing a + sign between the two conditions, EITHER condition being met makes the statement true and a count to be returned.

Standard syntax for an OR statement:

```
SUM(IF(Return_from_Condition1+Return_from_Condition2>0,1,0))
```

This formula is stating that if the sum of the return from Condition1 or 2 is greater than 0 then return a 1.

This could also be shorthanded as:

```
COUNTIF(Return_from_Condition1+Return_from_Condition2,">0")
```

Count-based Compound OR Formulas

Objective: Sum the Number of Activities with No Predecessors OR No Successors

Create the base formula logic:

```
SUM(IF(Return_from_Condition1+Return_from_Condition2>0,1,0))
```

Where the returns will return either a 1 or a 0

Next, detail out the two conditions:

IF(NumberofPredecessors=0,1,0)

IF(NumberofSuccessors=0,1,0)

Add the two detailed conditions to the main function:

SUM(IF(IF(NumberofPredecessors=0,1,0)+IF(NumberofSuccessors=0,1,0)>0,1,0))

The following table shows the possible outcomes of this multiple condition formula.

Number of Preds = 0	Number of Successors = 0	Return
No	No	0
No	Yes	1
Yes	No	1
Yes	Yes	1

Table 3-3 – Table of Multiple Condition Outcomes with OR Syntax

The shorthand form for an OR formula is slightly different.

SUM(IF((NumberofPredecessors=0)+(NumberofSuccessors=0)>0,1,0))

Compound Formula – Using AND and OR Together

Finally, AND and OR conditions can be combined together to create powerful formulas to show all types of results. Always start with the FULL formulas – do not begin with the shorthand formulas as characters can easily be left out. Once the formula is written and successfully tested, then create the shorthand formula.

For example:

Objective: Sum the Project Remaining Cost for Activities with No Predecessors OR No Successors and whose individual Remaining Cost is >\$1,000.

First create the OR formula to find the number of activities with No Predecessors or No Successors formula:

IF((NumberofPredecessors=0)+(NumberofSuccessors=0)>0,1,0)

Then, create the formula for returning Remaining Cost:

IF(RemainingCost>1000,RemainingCost,0)

Finally, combine the two formulas using the SUM syntax and the ‘*’ character:

SUM(IF((NumberofPredecessors=0)+(NumberofSuccessors=0)>0,1,0)
* IF(RemainingCost>1000,RemainingCost,0))

Other Math Functions

There are several other Math Functions in addition to AND and OR functions. Two specific functions that are frequently used are AVERAGE and MAX.

Objective: Average Remaining Duration for Normal Activities.

AVERAGE(IF(ActivityType="Normal",RemainingDuration,false))

This formula will first return the Remaining Duration for each of the Normal Activities. If the Activity is not Normal then it will return a false. False is specifically used vs. a 0 because False is ignored during an averaging function. Finally the returned remained durations will be averaged.

Objective: Maximum amount of lag on Remaining Durations of Incomplete Activities.

MAX(maxlag*IF(ActivityType<>"Complete",1,0))

This formula will return the maximum lag for each activity that is not complete.

WRITING THE PRIMARY METRIC FORMULA

The primary formula is the formula used to calculate the primary result calculated in an Acumen Fuse analysis. Primary formulas can return any type of numeric or text-based result. Primary formulas are applied to groupings of activities (depending on the ribbon, phase or intersection context).

Primary formulas can return any type of numeric or text-based result. To create a Primary formula:

Step 1) Define exclusions – these are overarching filters that limit which activities get included in the search by type/status/phase.

Step 2) Define Filters – these are the next level of filters further filtering out specific activities. Many metric definitions can be completed by just using exclusions and filters (e.g. Critical activities).

Step 3) Optional formula – if additional advanced criteria definition is needed, then select the Advanced mode and define the function using the advanced metric editor.

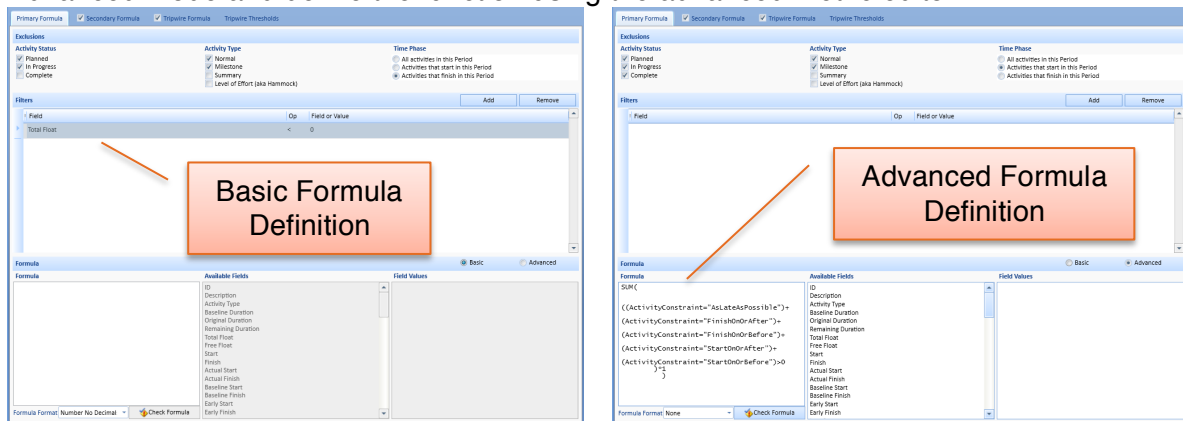


Figure 4.1 Basic and Advanced Primary Formula Definition

For both basic and advanced Primary formula definition, formatting of primary formula results is achieved using the “Formula Format” drop down list.

WRITING THE SECONDARY METRIC FORMULA

Once the Primary Metric Formula is created and successfully tested, the Secondary Formula can be written. The syntax for the Secondary Formula is similar to that of the syntax for the Primary Formula. The difference is that the Secondary Metric Formula is usually stated as a ratio, a percentage, or a portion of the total vs. the discrete number in the Primary Metric Formula.

A Secondary formula is additional information shown in a ribbon/phase or intersection analysis window.

There are two ways to create a secondary formula:

Simple percentage relative to the primary formula: if the secondary formula is representing a percentage of the primary formula, then there is no need to manually create formulas to create this result. Instead, simply select the relevant exclusions and filters (in order to define the

population against which you are going to divide the primary formula in order to calculate the percentage) and then set the mode to “Percentage of Primary Formula”. A simple percentage secondary formula can be auto-calculated in this mode irrespective of whether the primary formula has been defined in basic or advanced mode.

Advanced Secondary Formula: if the required secondary formula is not a simple percentage of the primary formula, then set the mode to “Advanced” and define the exclusions, filters and advanced formula manually as described below.

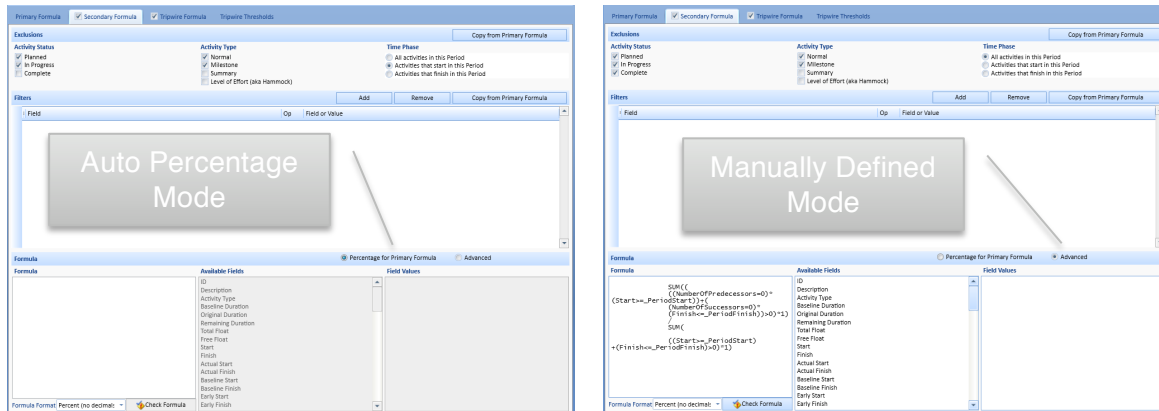


Figure 5.1 Simple Percentage of Primary Formula and Advanced Secondary Formula Definition

The secondary formula is an optional attribute of a metric and if not defined, it will not display in the analyzer windows. Secondary formulas are also applied to groupings of activities (depending on the ribbon, phase or intersection context).

For example, what if we want to know what the cost overrun is on the project. Cost Overrun is defined by Total Cost being greater than Budget Cost.

Advanced Percentage Examples

Objective: Percent of Non-Summary Activities that have a Cost Overrun as a portion of the total Number of Non-Summary Activities.

The first step is to create the full Primary Metric Formula to count the number of non-summary activities that have a Cost Overrun. This is the numerator for the Secondary Metric Formula.

$$\text{SUM}(\text{IF}(\text{TotalCost} > \text{BudgetCost}, 1, 0) * \text{IF}(\text{ActivityType} \neq \text{"Summary"}, 1, 0))$$

The second step is to create formula for the denominator of the Secondary Metric Formula. If the Secondary Formula will be expressed as a percentage, then usually the denominator is a total.

$$\text{SUM}(\text{IF}(\text{ActivityType} \neq \text{"Summary"}, 1, 0))$$

The full Secondary Formula divides the two individual formulas:

$$\frac{\text{SUM}(\text{IF}(\text{TotalCost} > \text{BudgetCost}, 1, 0) * \text{IF}(\text{ActivityType} \neq \text{"Summary"}, 1, 0))}{\text{SUM}(\text{IF}(\text{ActivityType} \neq \text{"summary"}, 1, 0))}$$

Or, in shorthand form:

$$\frac{\text{SUM}(\text{IF}(\text{TotalCost} > \text{BudgetCost}) * \text{IF}(\text{ActivityType} \neq \text{"summary"}))}{\text{COUNTIF}(\text{ActivityType} \neq \text{"summary"})}$$

NOTE: In this instance, the SUM(IF) was changed to COUNTIF since there is only one criteria used.

Let's look at another example using a schedule criterion.

Objective: Percent of Uncompleted Activities are Critical.

The first step is to again create the Primary Metric Formula to count the number of Uncompleted Activities that are Critical. This is the numerator for the Secondary Metric Formula.

$$\text{SUM}(\text{IF}(\text{Critical}, 1, 0) * \text{IF}(\text{ActivityStatus} \neq \text{"Complete"}, 1, 0) * \text{IF}(\text{ActivityType} = \text{"Normal"}, 1, 0))$$

The second step is to create formula for the denominator of the Secondary Metric Formula.

$$\text{COUNTIF}(\text{ActivityType} = \text{"Normal"}, 1, 0)$$

Therefore – the complete Secondary Formula divides the two individual formulas:

$$\frac{\text{SUM}(\text{IF}(\text{Critical}, 1, 0) * \text{IF}(\text{ActivityStatus} \neq \text{"Complete"}, 1, 0) * \text{IF}(\text{ActivityType} = \text{"Normal"}, 1, 0))}{\text{COUNTIF}(\text{ActivityType} = \text{"Normal"}, 1, 0)}$$

Or, in shorthand form:

$$\frac{\text{SUM}(\text{IF}(\text{Critical}) * \text{IF}(\text{ActivityStatus} \neq \text{"Complete"}) * \text{IF}(\text{ActivityType} = \text{"Normal"}))}{\text{COUNTIF}(\text{ActivityType} = \text{"Normal"})}$$

Ratio Example

Objective: Number of Milestones vs. Number of Normal Activities

The base formula is:

$$1: \frac{\sum \text{Milestones}}{\sum \text{Normal Activities}}$$

The \sum Milestones Numerator is:

$$\text{COUNTIF}(\text{ActivityType} = \text{"Milestone"})$$

The \sum Normal Activities Denominator is:

$$\text{COUNTIF}(\text{ActivityType} = \text{"Normal"})$$

The complete formula is:

$$(\text{"1:"} \& (\text{COUNTIF}(\text{ActivityType} = \text{"Milestone"}) / \text{COUNTIF}(\text{ActivityType} = \text{"Normal"})))$$

It is possible that this ratio could be 'upside down', so use the above formula and calculate the ratio. If the ratio is OK, no changes are needed.

If the ratio is not OK, then modify the formula to:

$$((\text{COUNTIF}(\text{ActivityType} = \text{"Milestone"}) / \text{COUNTIF}(\text{ActivityType} = \text{"Normal"})) : \text{"1"})$$

WRITING THE TRIPWIRE METRIC FORMULA

The tripwire formula is (optionally) used to determine the individual exceptions that are listed in the Activity Browser. Writing the Tripwire Metric Formula is simpler than writing the Primary or Secondary Metric Formulas as the Tripwire Metric Formula is not an array formula, but a classic MS Excel cell formula.

Tripwire formulas apply to individual activities and must return a Boolean (for each activity) in the form of either a “True” or “False” value. Metrics that do not contain a Threshold formula cannot be used to display activities in the Activity Browser and also cannot be used in the Comparison Analyzer. Tripwires are a very useful means of graphically depicting when a particular metric threshold is reached. Acumen Fuse tripwires are flexible with regards to the number of thresholds per metric that can be defined, the type of thresholds (absolute and gradient) and the formulas against which they can be based (primary and secondary). When creating or editing any of the metric formulas, you can use the “Check Formula” button to validate the syntax of the formula. Note that when using the check formula button the test calculation is applied to all activities within the workbook.

Tripwire formulas can be created in one of two ways:

Auto Calculated by Primary Formula: if the Primary formula was created using the basic mode, you can opt to automatically create the tripwire formula without defining any exclusions, filters or formulas for the tripwire definition. Instead, Fuse will automatically create a Tripwire formula based on the exclusions and filters defined in the primary formula. This mode cannot be used if the primary formula was created in advanced mode. In Auto Calculated by Primary Formula mode, the tripwire exclusions and filters options are disabled as they are not needed in light of the fact these settings are automatically inherited from the primary exclusions and filters.

Advanced: This mode enables you to manually create exclusions, filters and advanced functions that together return the required set of activities.

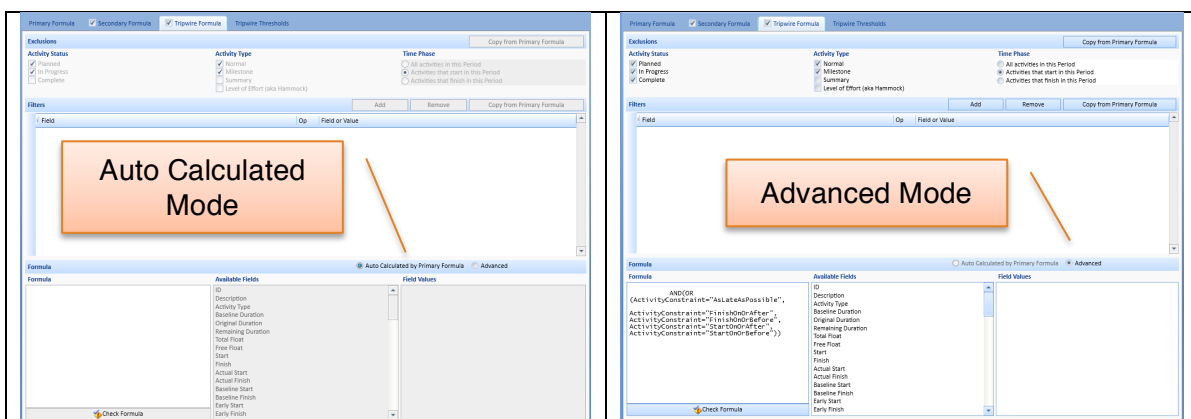


Figure 6 Auto Calculated and Advanced Tripwire Formula Definition

The following are several examples of advanced tripwire formulas using the Advanced Metric mode.

Cost Example

Objective: Yes or No – Is the Non-Summary Activity over Budget?

The formula is:

$IF(\text{TotalCost} > \text{BudgetCost}, 1, 0) * IF(\text{ActivityType} \neq \text{"Summary"}, 1, 0)$

Or, in shorthand form:

$AND(\text{TotalCost} > \text{BudgetCost}, \text{ActivityType} \neq \text{"Summary"})$

Schedule Example

Objective: Yes or No – Does the Activity with a Remaining Cost of >\$1,000 have Predecessors or Successors?

The formula is:

$IF(\text{RemainingCost} > 1000, 1, 0) * IF((\text{NumberofPredecessors} = 0, 1, 0) + (\text{NumberofSuccessors} = 0, 1, 0))$

Or, in shorthand form:

$AND(\text{RemainingCost} > 1000, (\text{NumberofPredecessors} = 0, \text{NumberofSuccessors} = 0))$

Once the Tripwire Metric formula is written, the thresholds must be defined.

Weighting

The weighting feature is only used for score carding or the executive report. It uses the tripwire formula for the scorecard. All metrics in the scorecard are, by default, equally weighted. They are weighted on a sliding scale from -10 to +10. The weighting formula is:

$\frac{(\text{metric A} * \text{weighting for A}) * (\text{metric B} * \text{weighting for B})}{10}$

10

DEFINING THE THRESHOLDS

The threshold editor enables customizable thresholds to be defined and associated colors set. Simply put, it is a means of classifying the activity results into buckets. The threshold can be shown as discrete (i.e. traffic lights), gradient, or mixed (combination of discrete and gradient).

Thresholds cannot be set for a range of values but must be defined for the boundary that defines a range.

They enable the grouping and aggregating of multiple activities together so that results for a ribbon or phase or intersection can be calculated.

Each metric includes an optional set of tripwire thresholds. These thresholds are used to graphically show when a defined threshold is exceeded. Tripwire thresholds can be based on either the primary or secondary formula. If the secondary formula is enabled (by checking the checkbox for the secondary metric), then the tripwire threshold is automatically associated with the secondary metric. If this checkbox is not checked, the tripwire threshold is automatically associated with the primary formula.

Defining Tripwire Threshold Scales

Tripwire threshold scales can be defined as having any number of intervals. To help with the creation of such scales, use the Tripwire Thresholds > Color Scales menu to automatically create standard scales.

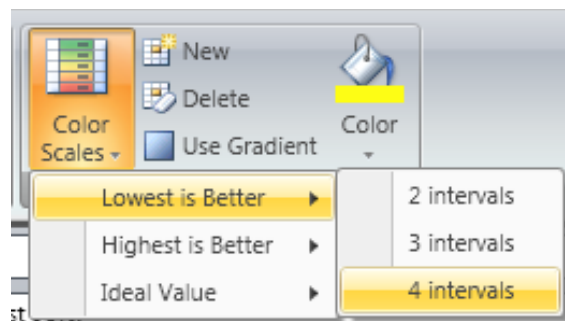


Figure 6-1 Tripwire Threshold Scales

This option provides three types of standard scale:

- *Lowest is better* – creates a scale where the lowest values are preferable
- *Highest is better* – creates a scale where the highest values are preferable
- *Ideal Value* - creates a scale where the middle values are preferable

For each of the three scale types, varying numbers of intervals can be created. In addition to using the standard scale types, additional intervals can be added through the Color Scales menu.

Normal and Gradient Scales

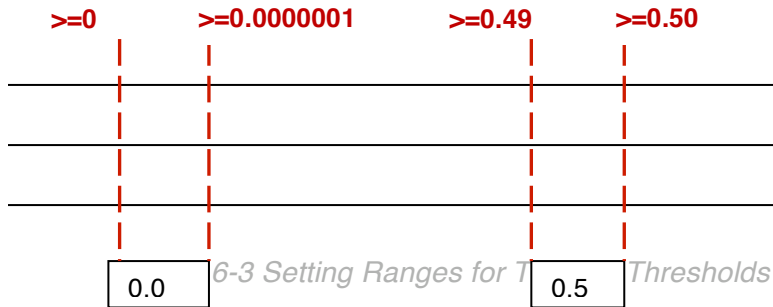
Threshold intervals can be defined as either normal or gradient. By default, scales are defined as “normal”. All threshold intervals within a single metric are either normal or gradient-based

(they cannot be mixed within a metric). Normal scales behave in an absolute or binary manner – that is, a metric result either triggers a threshold or it doesn't. A Gradient Scale behaves differently, in that a metric result, while falling within a given interval, can be represented as being close to an interval boundary. This type of scale is useful when determining how close to a tripwire boundary does a metric result get. When using gradient scales, instead of discrete colors for the intervals being used, gradient scales of color are used (based on where the metric falls in the scale).

Tripwire Thresholds

Once the Tripwire Scales have been defined, the threshold limits must be set. Typically a different color is setup for each different limit. Since Acumen Fuse defines ranges vs. limits – small ranges must be created around a limit. Identify the threshold or level for each color, then, identify a small limit around that threshold to create the tripwire threshold.

For example, to detect how many activities are at zero and at 0.5, levels are defined around zero and 0.5 as follows:



Once these ranges are defined, enter them into the table to create the tripwires. If color is associated with each one of the levels, then the colors will appear on the Ribbon View after analysis.

Including/Excluding Metrics from Analysis

By default, each metric is available in all three analyzers (ribbon, phase, intersection). Optionally, metrics can be excluded from a particular analysis (e.g. phase) if, for example, the context is not valid. Including/excluding metrics from each of the three analyzers is done through the three check boxes in the “Applies To” menu.

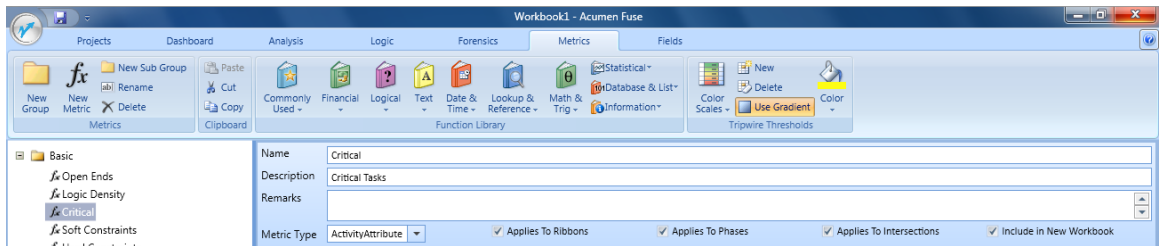


Figure 6-4 Inclusion/Exclusion of Metrics to Analyzers

SHORTCUTS AND RULES

For several of the Acumen Fuse metrics, the syntax for the metric formula can be written in shorthand. The formula works exactly the same way, just in fewer characters. The following are examples of the most common shortcuts in metric formulas.

Counting Shorthand with One Criterion

If there is only one criterion listed in the formula, the full formula must be written:

`SUM(IF(ActivityType<>"Summary",1,0))`

This formula cannot be shorthanded using SUM(IF) with one criterion. To write it in shorthand SUM(IF) must be changed to COUNTIF.

Therefore the shorthand formula is written as:

`COUNTIF(ActivityType,"<>Summary")`

Counting Shorthand with Multiple Criteria

For a formula that has multiple criteria, the full formula is written as:

`SUM(IF(RemainingCost>1000,1,0)*IF(ActivityType="Normal",1,0))`

The red characters and the IF statement can be eliminated:

`SUM(IF(RemainingCost>1000,1,0)*IF(ActivityType="Normal",1,0))`

The shorthand formula is written as:

`SUM((RemainingCost>1000)*(ActivityType="Normal"))`

AND Shorthand with Multiple Criteria (Tripwire formulas)

If the formula uses multiple criteria, the full formula is written as:

`IF(TotalCost>BudgetCost,1,0)*IF(ActivityType<>"Summary",1,0)`

The shorthand formula can be written as as series of conditions within an AND statement:

`AND(TotalCost>BudgetCost, ActivityType<>"Summary")`

DEFINITIONS

Tripwire Threshold

The Tripwire Threshold is used to graphically show when a defined threshold is exceeded.

Primary Metric Formula

The primary formula is the formula used to calculate the primary result calculated in an Acumen Fuse analysis.

Secondary Metric Formula

The secondary formula is an optional attribute of a metric and is usually used to show the metric as a percentage.

Tripwire Formula

The tripwire formula is (optionally) used to determine the individual exceptions that are listed in the Activity Browser.

Ribbons

Ribbons are groupings of activities based on a given criteria regardless of the time period.

Phases

Phases are groupings of activities based on a given criteria within a specific time period.

Intersections

Intersections are user definable places where a ribbon and a phase connect.

COMMONLY USED SYNTAX

IF(logical_test, value_if_true, [value_if_false])

The IF function returns one value if the specified condition is TRUE and returns another value if the specified condition is FALSE.

Logical_test – Required - Any value or expression that can be evaluated to TRUE or FALSE

Value_if_true - Required -The value to be returned if the “logical_test” argument evaluates to TRUE

Value_if_false – Optional - The value to be returned if if the “logical_test” argument evaluates to FALSE. If omitted then zero is returned

Example: IF(TaskStatus="InProgress",1,0) returns 1 if the activity status is equal to “InProgress” otherwise 0 is returned.

IF statements can be written in shorthand within Acumen Fuse. If the IF function name and Value_if_true and Value_if_false parameters are omitted, the Acumen Fuse engine will assume that the function is an IF statement returning either a 1 or a 0.

e.g. IF(TaskStatus="InProgress",1,0) can be written in shorthand as (TaskStatus="InProgress")

SUM(number1, [number2], [number3], [number4], ...)

The SUM function adds all the numbers specified as arguments.

number1 – Required - The first item that you want to add

number2, number3, number4, ... –Optional - The remaining items that you want to add

Example: SUM(ActualCost) returns the sum of the Actual Cost.

AND(logical1, [logical2], ...)

Returns TRUE if all its arguments evaluate to TRUE; returns FALSE if one or more arguments evaluate to FALSE. Most commonly used in the Tripwire formula.

logical1 – Required - The first condition that you want to test that can evaluate to either TRUE or FALSE.

logical2, ... - Optional - Additional conditions that you want to test that can evaluate to either TRUE or FALSE

Example: AND(ActivityType="Normal", ActivityStatus<>"Complete") returns TRUE if the activity type is “NORMAL” and activity status is not equal to “COMPLETE”.

MAX(number1,number2,...)

Returns the largest value in a set of values.

Number1, number2, ... - are 1 to 255 numbers for which you want to find the maximum value.

Example: MAX(TotalFloat) returns the maximum Total Float.

AVERAGE(number1, [number2],...)

Returns the average (arithmetic mean) of the arguments.

number1 – Required - The first number for which you want the average.

number2, ... - Optional - Additional numbers for which you want the average, up to a maximum of 255

Example: AVERAGE(TotalFloat) returns the average Total Float.

COUNTIF(range, criteria)

Counts the number of occurrences that meet a given criteria.

Range - Required - One or more fields that contain numbers.

Criteria - Required - A number, expression, or text string that defines which records to be counted. For example, criteria can be expressed as 3, ">3", "Normal", or "3".

Example: COUNTIF(TotalFloat,">5") counts the number of activities who have a Total Float value greater than 5.