Deltek

Deltek Acumen 8.8

Metric Developer's Guide

October 7, 2022

While Deltek has attempted to verify that the information in this document is accurate and complete, some typographical or technical errors may exist. The recipient of this document is solely responsible for all decisions relating to or use of the information provided herein.

The information contained in this publication is effective as of the publication date below and is subject to change without notice.

This edition published October 2022.

Contents

Introduction

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

> **Attention:** For more information about metrics, including industry standards metrics such as DCMA, NASA Health Check, and GAO, see the Acumen online help Metrics Tab topics.

What Is a Metric?

A metric is a standard or measure for use in determining how well a project is planned and executed.

Metrics contain formulas 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 (for example, cost or duration) or percentages (for example, percentage of total project duration). Percentages are useful for portraying results within a given context. Acumen uses metric libraries to group metrics for project analysis. Standard metric libraries pertaining to schedule quality, cost, project performance, risk exposure, Earned Value, the DCMA 14 Point Assessment, 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 allows you to customize your project or program analysis.

It is important to understand how the metrics are constructed, calculated, and applied.

To What Do Metrics Apply?

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

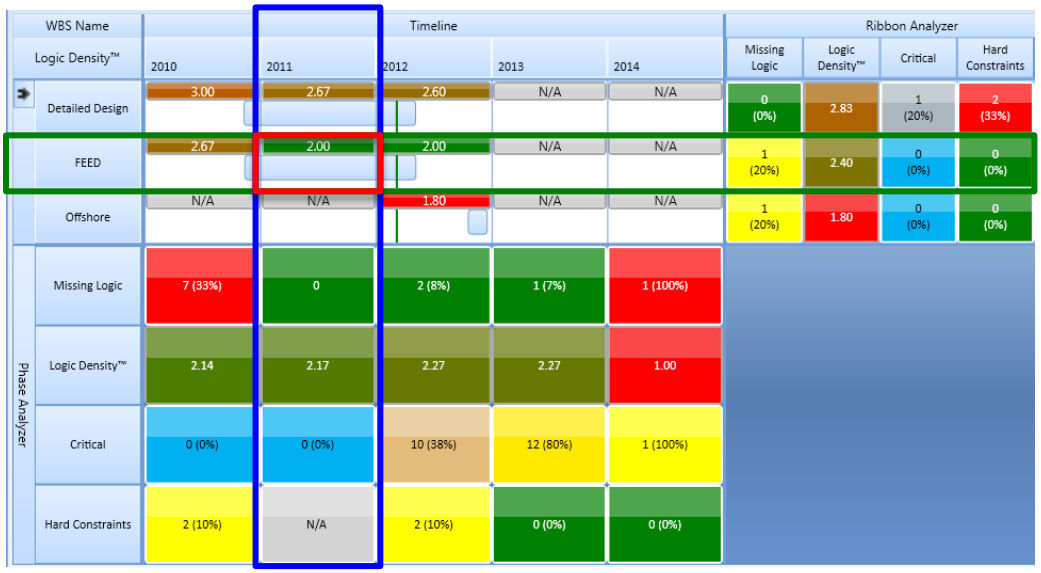
	A	B	C	D
1		Attributes		
2		Critical	Start Date	Remaining Cost
3	Activity 1	Yes	3/5/2013	0
4	Activity 2	No	4/20/2013	0
5	Activity 3	No	6/15/2012	1200

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

Acumen metrics are:

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

The screenshot below shows an example view of Ribbon and Phase Analysis in Deltek Acumen.



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

Analysis Type	Summary
Ribbon Analysis	Analyzing and comparing groups of activities (that is, activities grouped by WBS, contractor, or CAM).
Phase Analysis	Analyzing project time phases or identifying trends.
Intersection Analysis	Analyzing a group of activities in a certain time-phase. (that is, pinpointing project hot spots).

Grouping by Ribbons

Ribbons can be groupings of activities, control accounts, or work packages which are based on a given criteria. By default, ribbons are grouped by project but can be grouped in multiple ways, for example, 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 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.

Types of Acumen Fields

When creating metric formulas, you can reference several types of fields:

- Activity fields
- Project fields
- Workbook fields
- Dynamic fields
- Calculated fields
- Project variables
- Control account and work package fields (if you have loaded Cobra cost data)
- Work Authorization Document (WAD) fields (if you have loaded cost and WAD data)

Activity Fields

Activity fields are the most commonly used type of field in an Acumen 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 (for example, “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 in which they are being used. **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.

Calculated Fields

Calculated fields allow you to evaluate a standard part of a formula for reuse in metrics, either at the same level at which they are defined (activity, work package, control account), or to roll values up or down from one level to another.

You can use a calculated field to develop an activity metric to see data in the activity, control account, or WAD, or create a control account metric that sees, for example, the earliest baseline start of an activity.

When you are designing a metric, you start by creating a calculated field to pull the data and then use the calculated field in a metric.

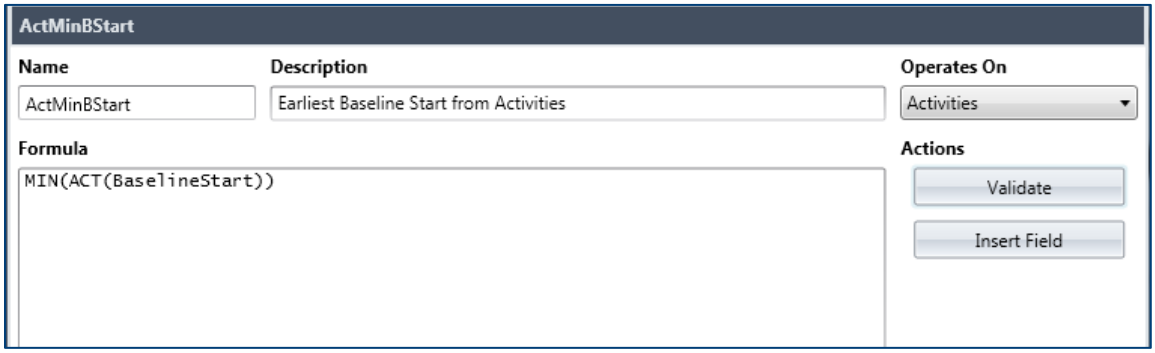
Calculated fields are available for use in the Filter and Formula sections of the metrics.

The Acumen installation includes several default activity, work package, and control account calculated fields (when using the DECM metric template). In order to use calculated fields, you must have imported the project cost data because that establishes control accounts and work packages.

The Functions menu group includes a few of the more common functions; however, you can use many other functions in a calculated field or formula.

Example of a Calculated Field

Using the formula below, you can view the earliest baseline start from activities. The **BaselineStart** field could exist in more than one place (activity, work package, control account, WAD); however, **ACT** shows you that it is from the activity. This is further confirmed by the **Operates On** field being set to **Activities** which means that there should only be activity fields in the formula.



The **Operates On** field also dictates where you can use the calculated field. For example, if the calculated field operates on activities, you can't use it in an activity metric. You can only use it in a work package or control account metric. Similarly, if the calculated field operates on a work package, you can only use the calculated field in an activity or control account metric.

Project Variables

Project variables are defined without a value on the metrics tab. The value comes from each project or snapshot. This allows you to have values that change over time. For example, you may want to see the number of remaining contract milestones. As the contract progresses, the number decreases because you have met some of those milestones.

A variable (as opposed to a project variable) has one value for an entire workbook. For example, if you have a **Duration Upper Limit** variable of **44** days, that value applies to everything in the workbook – it can't be varied by project. In addition, variables can only be used in metric filters and do not have data types.

After you define the project variable on the Metrics tab, you can add the value for each project or snapshot. You can either do this on the S2 // Diagnostics tab using the Variable Definition dialog box, or on the S1 // Projects Activity Details Pane Project Variables tab. One advantage of using the Variable Definition dialog box is that you can see the trend over time. In addition, you can click on the variable and use the **Variable Description** field at the bottom of the dialog box to see how the variable is to be used.

Variables vs. Project Variables

The table below summarizes the differences between a variable and a project variable.

Variable	Project Variable
One value for an entire workbook	Values defined for each project and snapshot
Use in filters only	Use in both filters and advanced formulas
No data type	Define the type: ▪ Number ▪ Text ▪ DateTime ▪ Duration ▪ Boolean

Control Account/Work Package Fields

A control account is also known as a cost account (CA). It is a management control point at which actual costs can be accumulated and compared to budgeted cost of work performed. A CA is a natural control point for cost/schedule planning and control since it represents the work assigned to one responsible organizational element on one contract work breakdown structure (WBS) element. From here, work effort is further broken down into Work Packages (WPs) and detailed work is planned. A control account may have hundreds of work packages.

A work package (WP) identifies the major tasks, groups of tasks, detailed short-span jobs, and material items (that are identified by the contractor) needed to accomplish the work required to complete the contract. It has a direct connection between the scheduled activities and the related cost data.

You can import control account/work package fields from both a Deltek Cobra data file and Work Authorization Document (WAD).

Work Authorization Document (WAD) Fields

The Work Authorization Document (WAD) is a formal document identifying scope of work and budget at a control account level. It is used to plan work packages. It is an open format. Some people use Excel spreadsheets or Word documents; some use simple text, and so on. To import WAD information into Acumen, the data must be in an Excel spreadsheet format (extension .xls or .xlsx).

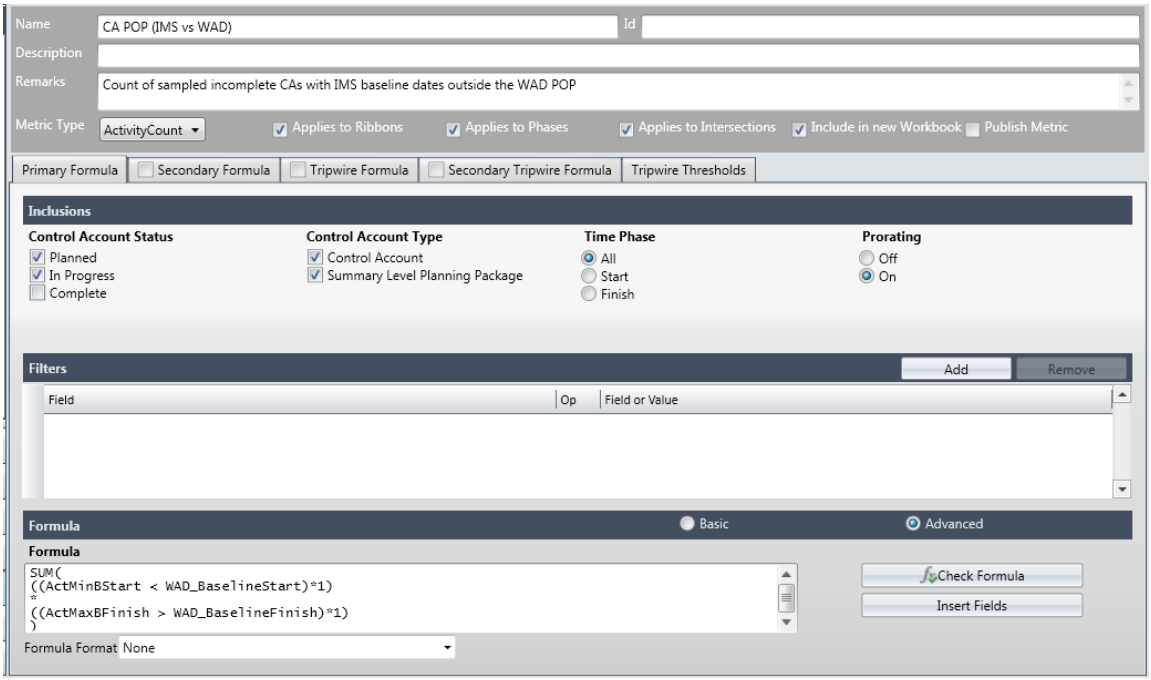
Control Account and Work Package Metrics

Control account and work package metrics are used on the data that comes from Cobra. They have additional capabilities over and above standard metrics. For example, the inclusions are different and they have a secondary tripwire formula.

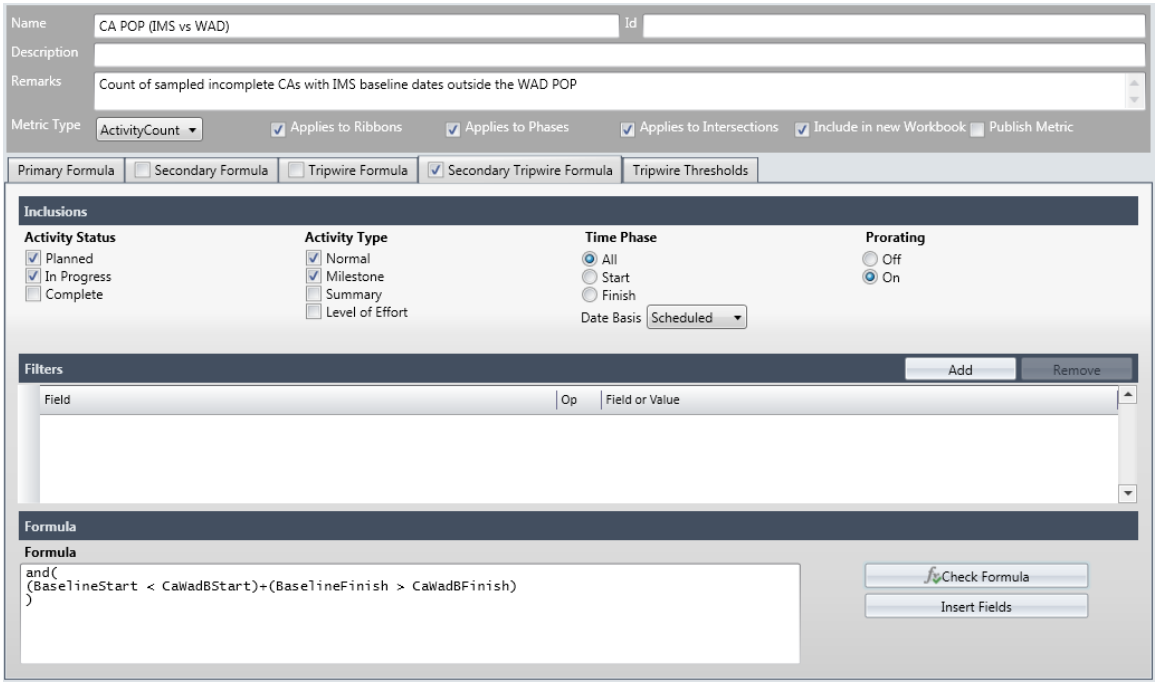
Using the Secondary Tripwire Formula

The Secondary Tripwire Formula tab is only available for control account and work package metrics. The secondary tripwire formula always looks at activities, regardless of whether you are using a control account or work package metric. Its purpose is to find the activities that are causing the work package or control account an issue.

In the example below, we want to know the number of control accounts that have activities that are outside of the dates that were authorized in the WAD. The metric uses activity-calculated fields to find the earliest baseline start and the latest baseline finish and compares them to the WAD baseline start and WAD baseline finish.



When you review the data, it tells us that we have three control accounts that meet the criteria. That gives us a general idea of where the problems lie but we need to drill down further in order to pinpoint exactly where the issue is. This is where the Secondary Tripwire Formula comes into play. It allows us to see which activities, by control account, are outside the WAD dates.



Performance Method

The table below lists the short and long performance method values. The short value either comes from Cobra in the DCDE or from the Cost .csv file. You use the long value when you create metrics that include/exclude various performance methods.

The long value displays in the Performance Method column on the Control Accounts / Work Packages tab of the S1 // Projects Activity Details Pane.

Short Value	Long Value
<blank>	None
C	PercentComplete
P	PercentCompleteManualEntry
G	HundredZero
F	ZeroHundred
E	FiftyFifty
J	Apportioned
L	AssignmentPercentageComplete
M	CalculatedApportionment
O	EarnedAsSpent

Short Value	Long Value
N	EarningRules
A	LevelOfEffort
B	Milestone
K	PlanningPackage
D	UnitsComplete
H	UserDefined
R	PERT

Ribbon and Phase Scoring

When weighting metrics to calculate a score, Acumen counts each record (activity, control account, work package) as one unit. For example, if you have 10 activities, 10 control accounts, and 10 work packages, each category (activity, work package, control account) accounts for 1/3 of the score.

Weighting Example

In the example below, there are 87 total records and 62 passed all of the metrics, we calculate 62 / 87 to get a ribbon score of 0.712.

	# of Records	# of Records that Passed	Ribbon Score
Activities	51	51	
Work Packages	28	4	
Control Accounts	8	7	
Total Count	87	62	0.712

- **# of Records** — This is the number of records for each type (activity, control account, work package). Dependent on inclusion filters. This is what ribbon metrics look at.
- **# of Records Passed** — This is the number of records that passed all of the metrics. If a record fails even one metric, it counts as a failure by default (For more information, see *Record-Based vs. Metric-Based Scoring* below).
- **Ribbon Score** — This score is the (Total Records Passed) / (Total Count).
- **Total Count** — This is the total number of records and the total ribbon score for all records.

Record-Based vs. Metric-Based Scoring

In Acumen, you can select record-based scoring (activity, control account, or work package) or metric-based scoring. You make this selection in Acumen on the Deltek Acumen Options dialog box User Interface tab.

Record-based scoring calculates the score on a per-record basis and the final score is an average of the per-record scores. This places a premium on absolute compliance with the assigned metrics. The breach of any single metric on a task will prevent those tasks from receiving a “passing grade.”

It is not always a bad thing when an activity fails a metric. For example, failing a ‘Negative Float’ metric means the activity has positive float, which is often beneficial. Metrics can be designated as "Bad," "Neutral," and "Good" using the metric editor within Acumen. In activity-based scoring, any degree of "Bad" will prevent the activity from having a positive impact on the overall score. Conversely, when a metric is rated as "Good" and a certain task fails to meet the criteria for the metric, it will not count toward the score.

With metric-base scoring, Acumen calculates the score using an average of the metric scores. Metric-based scoring allows incremental improvements to tasks to be reflected in the overall score. The metric-based method typically produces higher scores, as a single task would need to fail in multiple areas simultaneously in order to receive a very low score.

You can use sliders on the Metrics tab to adjust the weighting of each metric from "Bad" to "Good." The overall weight assigned to each metric appears as a numerical value on the right-hand side of the screen. For each condition that the metric meets, it will accumulate the point value assigned as the weight, whether the point value is negative or positive. The score is then normalized on a scale of 0 to 100, which is used to determine the score per activity.

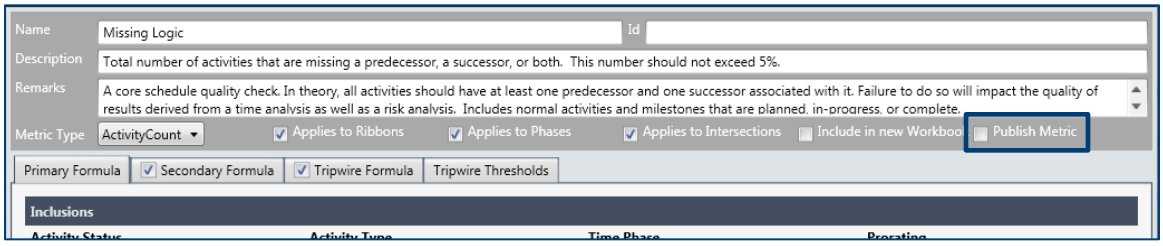
Phase Scoring

Phase Scoring uses the same calculation but applies it only to the specified phase.

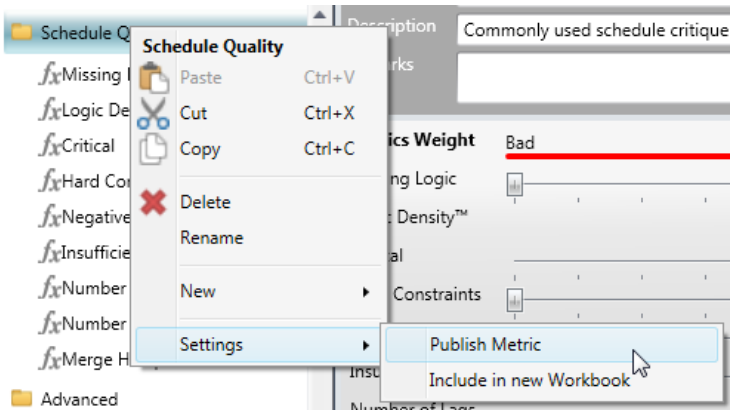
> **Attention:** For more information, see *Project Scoring in Acumen* in the Acumen online help (Before You Begin… » Project Scoring in Acumen).

Publishing Metrics

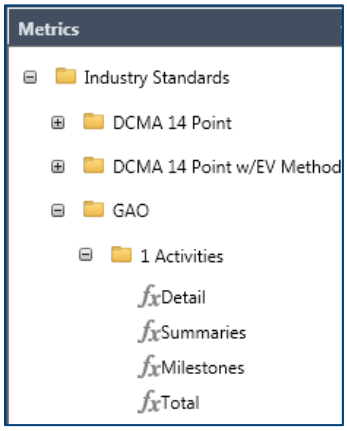
You can choose which metrics are published by selecting the **Publish Metric** option on the Metrics form for the selected metric.



To include or to exclude an entire metric library, right-click on the library folder in the Library Pane and click **Settings » Publish Metric** (the **Publish Metric** option has a blue checkmark next to it when selected.)



When you publish metrics, the report combines metrics based on the folder hierarchy on the Metrics tab. For example, if you have a **GAO** metrics folder with a **1 Activities** sub-folder on all of the Metrics forms, Acumen consolidates the data in the **1 Activities** folder on all metrics forms into a single tab when it generates the report. Each report tab is then sorted using the ID entered on the Metric form.



To ensure that all metrics are grouped correctly, you can click **Sync Groups** in the Metrics menu group to synchronize the folder structures across all of the Metrics forms (Metrics, Work Package Metrics, and Control Account Metrics).

If you don't have identical structures across the Metrics forms, the report will include more tabs since less data will be consolidated.

Metric Formula Development

Formula Syntax

Acumen 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

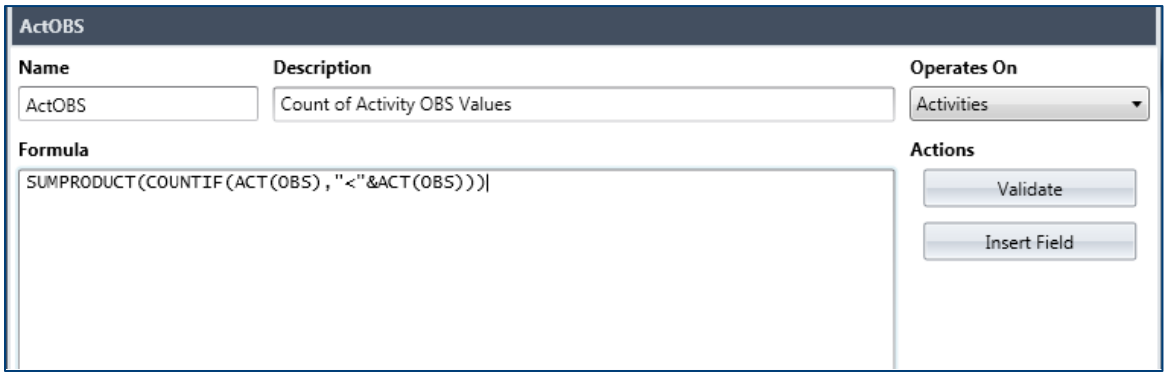
Acumen metric formulas are based on what is known as “Single Value Result Array formulas.” These work with a series of data (activities), aggregate it (typically using the likes of SUM, AVERAGE or COUNT) and return a single value to the (ribbon, phase or intersection) analyzer.

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). 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, 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 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.

Example of an Array Formula

The formula in the example below counts how many different OBS values the activities have that roll up to the control account or work package. You could use this calculated field at a control account level since a control account should only have one OBS. If it has more than one, then you know that there is an error.



Example of Ribbon and Phase Calculation of Metrics

Array Formula
Number of Lags
= 1+ 2 + 0 = 3

Formula Types

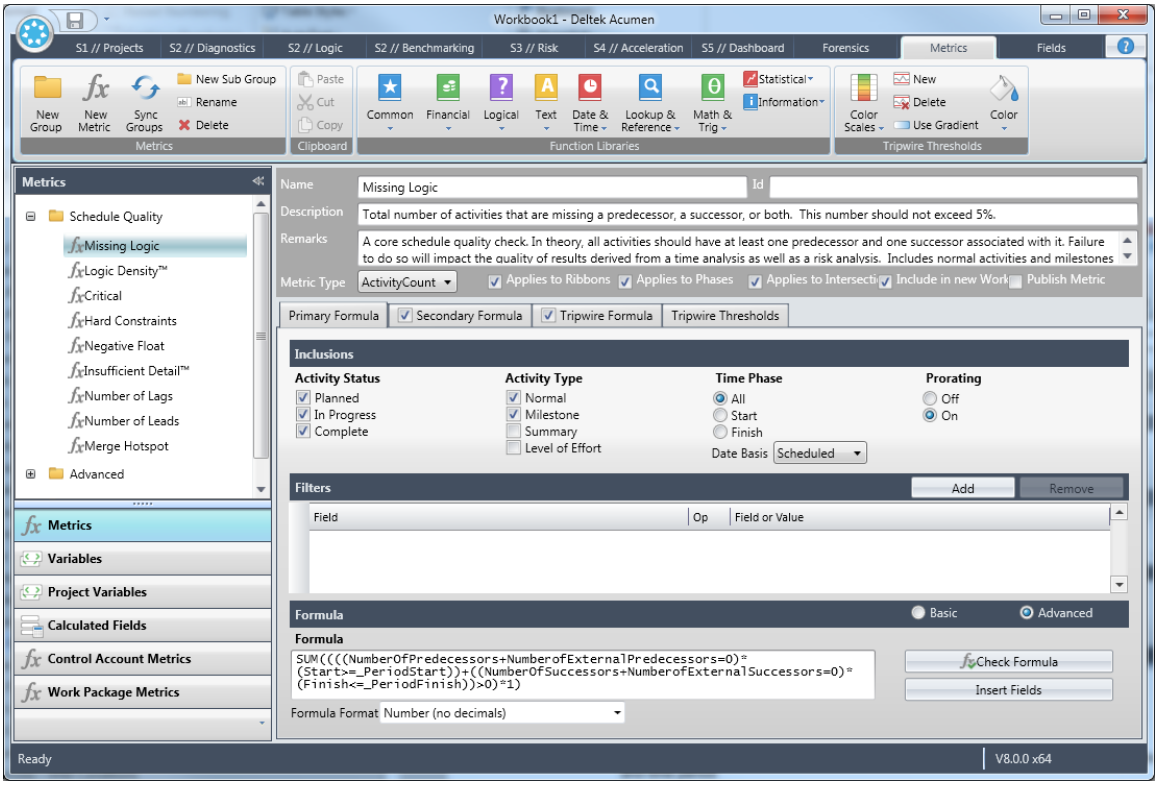
There are four formulas in Acumen and they all use MS Excel Array formula syntax.

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

These are discussed in greater detail later in this guide.

Acumen Metric Editor

The 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.

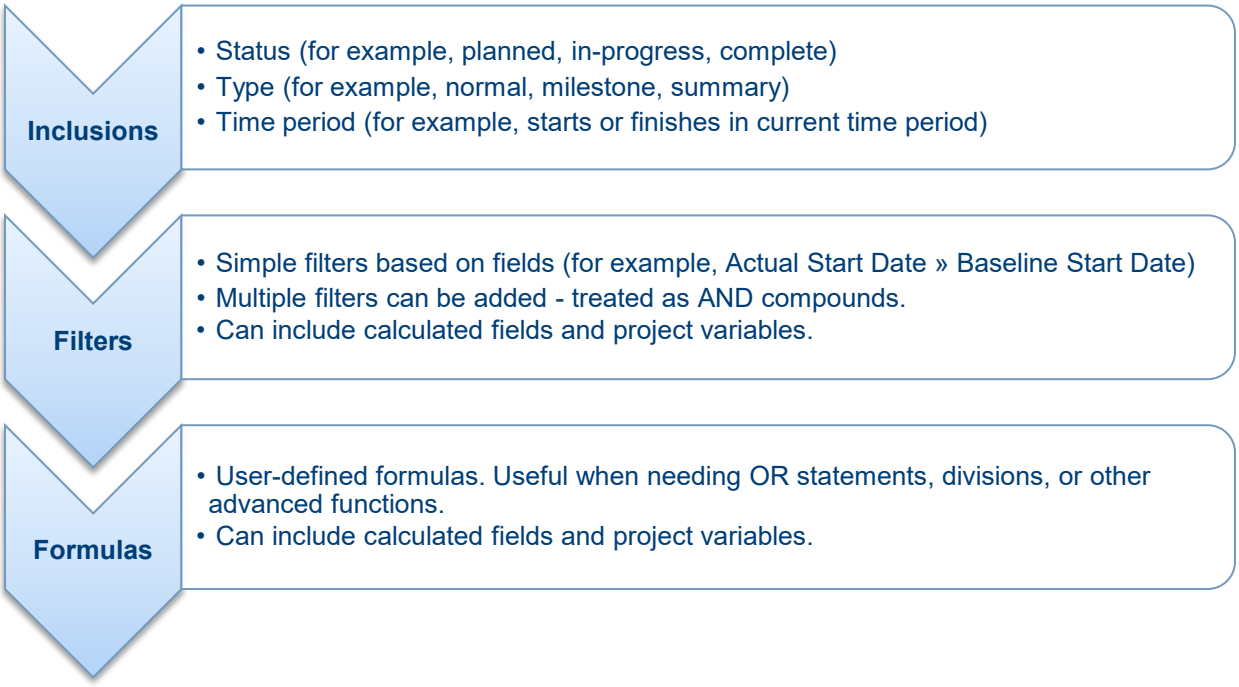


Note: The Control Account and Work Package Metrics are only available if you have the Cobra option enabled on the S1 // Projects tab in the Get External Data From menu group.

The activity metrics can include three formulas (primary, secondary, and tripwire). The control account and work package metrics can include four formulas (primary, secondary, tripwire, and secondary tripwire). Each of these formulas can be built using a three-level hierarchy:

- **Inclusions** — A top level set of filters to include specific data based on type, status and time period.
- **Filters** — Standard filters that further pinpoint specific activities.
- **Formula** — Basic and advanced custom formulas to further specify criteria sets.

Inclusions, filters, and formulas are hierarchical.



Writing Formulas

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

- **Basic** — 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 Acumen.

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 met

```
Function(fieldvalue,value1,value2)
```

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 Acumen 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 – Add 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 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' is the condition is not met, is implied through the default return of true/false.

Many of the standard Acumen 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 later in this guide.

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 – Return Field Values Instead of 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.

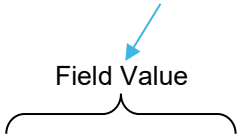
This is the final modification of the metric formula for a summing all of the activity remaining costs, if they are greater than $1,000, and presenting a total.

Objective: Sum the Remaining Cost for all Normal Activities where 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,RemainingCost,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 – that is, 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.

Insert the new condition.

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

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

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

Value-based Compound AND Formulas

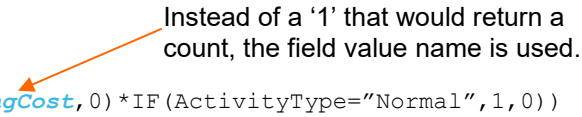
In the previous example we were counting results. In this example, the same criteria are applied but instead of counting results we are going to sum up the values (remaining cost) of those activities that pass the compound criteria test.

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))
```

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 Formulas – 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

The shorthand form for an OR formula is slightly different.

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

Compound Formulas – Using "AND" and "OR" Together

"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))
```

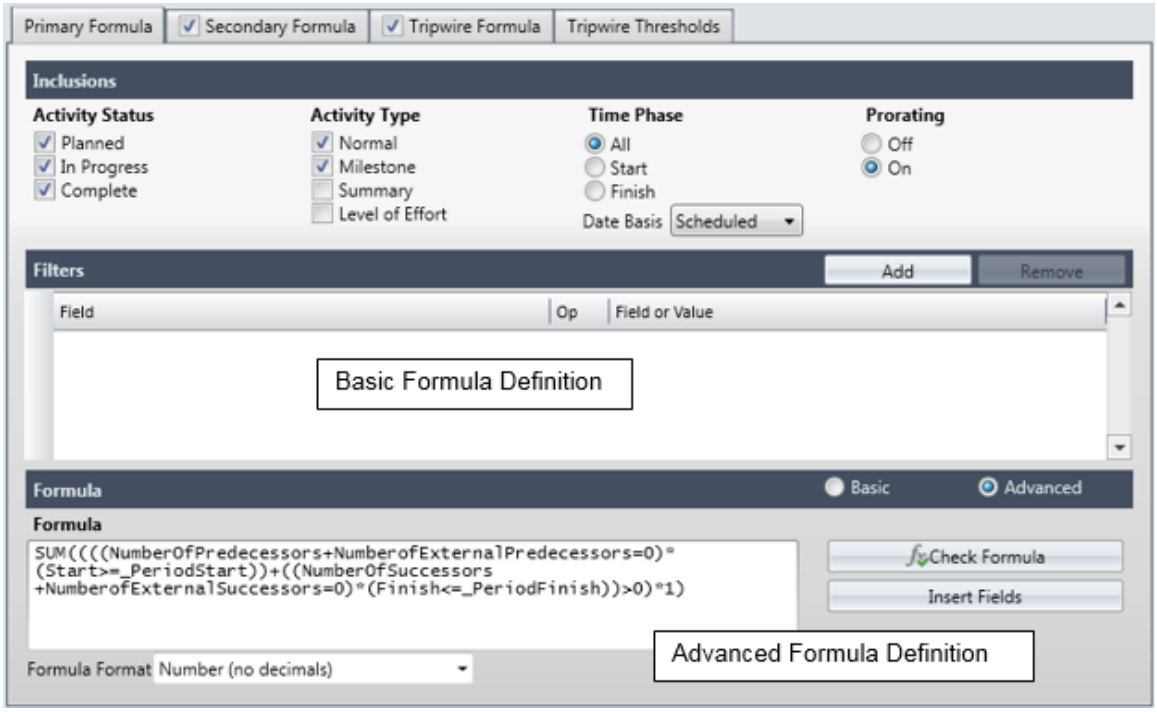
This formula will return the maximum lag for each activity that is not complete.

Write the Primary Metric Formula

The primary formula is the formula used to calculate the primary result calculated in an Acumen analysis. Primary formulas can return any type of numeric or text-based result. Primary formulas are applied to groupings of activities, work packages, or control accounts (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 inclusions — These are overarching filters that specify which activities, control accounts, or work packages 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, control accounts, or work packages. Many metric definitions can be completed by just using exclusions and filters (for example, 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.



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

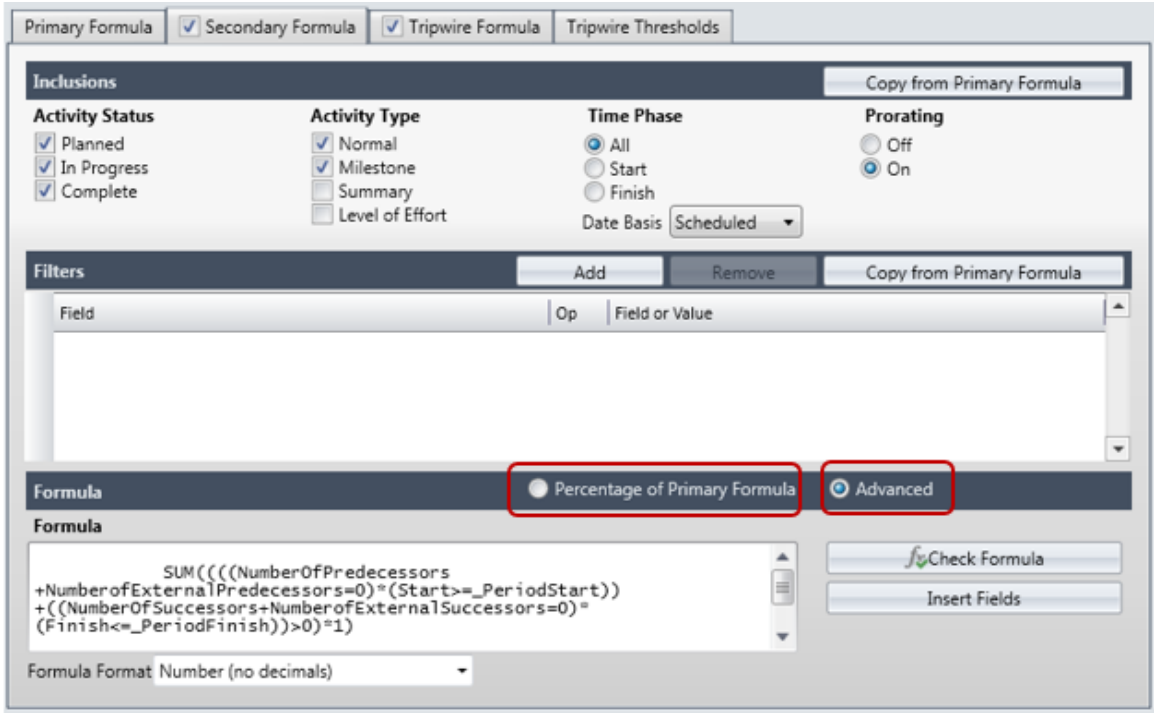
Write the Secondary Metric Formula

After the primary metric formula is created and successfully tested, you have the option to create a secondary formula. The syntax for the secondary formula is similar to that of the syntax for the primary formula. The difference is that the secondary formula is usually stated as a ratio, a percentage, or a portion of the total vs. the discrete number in the primary 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:

- **Percentage of 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** — 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.



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).

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.

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

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

The full Secondary Formula divides the two individual formulas:

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

Or, in shorthand form:

```
SUM(IF(TotalCost>BudgetCost)*IF(ActivityType<>"summary"))/
COUNTIF(ActivityType<>"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.

```
SUM(IF(Critical,1,0)*IF(ActivityStatus<>"Complete",1,0)
*IF(ActivityType="Normal",1,0))
```

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

```
COUNTIF(ActivityType="Normal",1,0)
```

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

```
SUM(IF(Critical,1,0)*IF(ActivityStatus<>"Complete",1,0)*IF
(ActivityType="Normal",1,0))/COUNTIF(ActivityType="Normal",1,0)
```

Or, in shorthand form:

```
SUM(IF(Critical)*IF(ActivityStatus<>"Complete")*IF(ActivityType=
"Normal"))/COUNTIF(ActivityType="Normal")
```

Ratio Example

Objective: Number of Milestones vs. Number of Normal Activities

The base formula is:

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

The Σ Milestones Numerator is:

```
COUNTIF(ActivityType="Milestone")
```

The Σ Normal Activities Denominator is:

```
COUNTIF(ActivityType="Normal")
```

The complete formula is:

```
("1:"&(COUNTIF(ActivityType="Milestone")/COUNTIF(ActivityType=
"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:

```
((COUNTIF(ActivityType=”Milestone”)/COUNTIF(ActivityType=
”Normal”))”:1”)
```

Write the Tripwire Metric Formula

Acumen tripwires are flexible with regards to:

- The number of thresholds per metric that can be defined
- The type of thresholds (absolute and gradient).
- The formulas against which thresholds can be based (primary and secondary).

The tripwire formula is optional. It is 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.

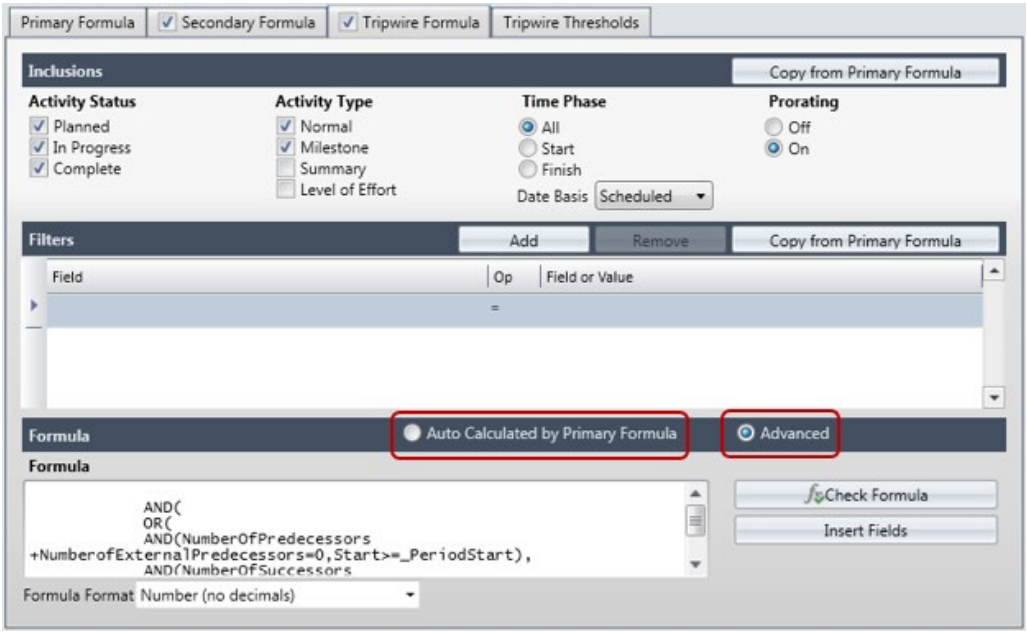
Tripwire formulas apply to individual activities, work packages, or control accounts and must return a Boolean (for each activity, work package, and control account) in the form of either a **True** or **False** value. Metrics that do not contain a threshold formula cannot be used to display activities 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 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, work packages, or control accounts within the workbook.

Tripwire formulas can be created in one of two ways:

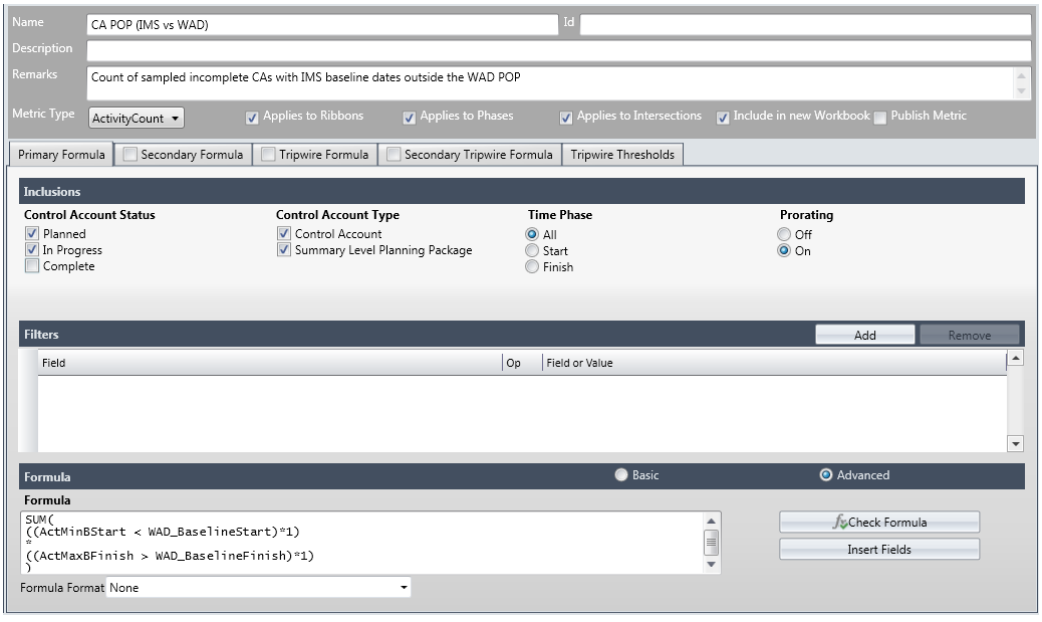
- **Auto Calculated by Primary Formula** — If the primary formula was created using the basic mode, you can opt to automatically create the tripwire formula without defining any exclusions, filters or formulas for the tripwire definition. Instead, Acumen will automatically create a tripwire formula based 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 this 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, work packages, or control accounts.



Secondary Tripwire Formula

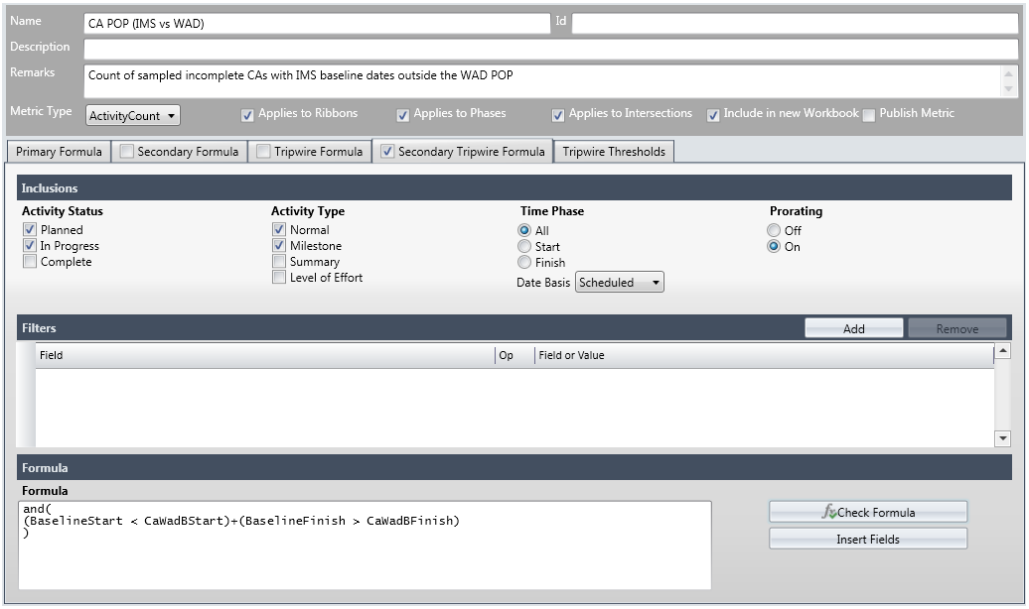
The secondary tripwire formula always looks at activities, regardless of whether you are using a control account or work package metric. Its purpose is to find the activities that are causing the work package or control account an issue.

In the example below, we want to know the number of control accounts that have activities that are outside of the dates that were authorized in the WAD. The metric uses activity calculated fields to find the earliest baseline start and the latest baseline finish and compares them to the WAD baseline start and WAD baseline finish.



When you review the data, it tells us that we have three control accounts that meet the criteria. That gives us a general idea of where the problems lie but we need to drill down further in order to pinpoint exactly

where the issue is. This is where the Secondary Tripwire Formula comes into play. It allows us to see which activities, by control account, are outside the WAD dates.



Cost and Schedule Examples

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

Or, in shorthand form:

```
AND(TotalCost>BudgetCost,ActivityType<>”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(RemainingCost>1000,1,0)*IF((NumberofPredecessors=0,1,0)+(NumberofSuccessors=0,1,
0))
```

Or, in shorthand form:

```
AND(RemainingCost>1000,(NumberofPredecessors=0,NumberofSuccessors=0))
```

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

Weighting

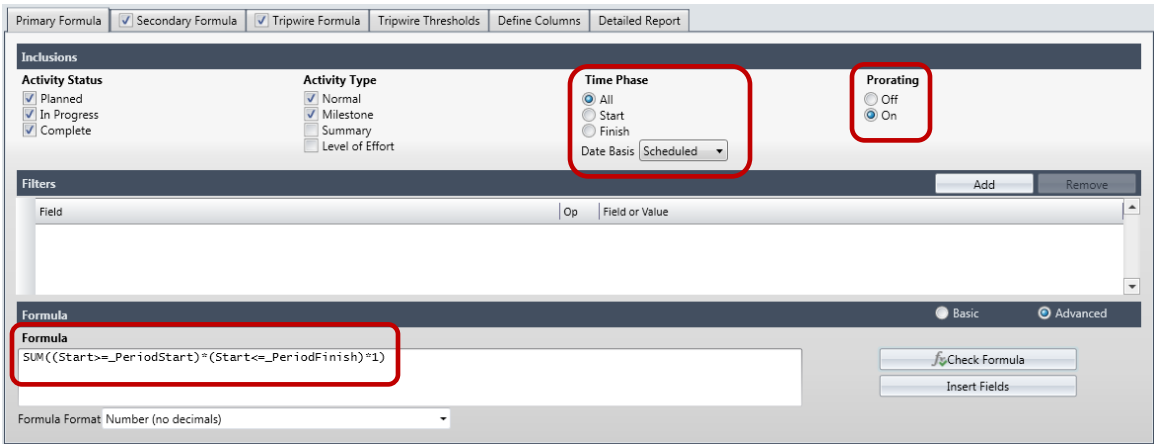
The weighting feature is only used for score carding or the executive report. It uses the tripwire formula for the scorecard. All metrics in the scorecard are, by default, equally weighted. They are weighted on a sliding scale from -10 to +10. The weighting formula is:

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

Phase Analyzer Formulas and Settings

There are several fields and settings that you can use to specify how the phase analyzer analyzes information. This topic describes the following:

- Period Start / Period End Dynamic Formula Fields
- Time Phase Options
- Prorating Options



Period Start and Period End

Period Start (**_PeriodStart**) and Period End (**_PeriodEnd**) are dynamic fields that indicate the start and finish of the current phase.

You can use these fields in metric formulas to compare dates on activities to evaluate whether they are in the period of the phase analyzer regardless of what periods are being shown (months, quarters, years, and so on).

For example, **Sum((Start>=_PeriodStart)*(Start<=_PeriodFinish)*1)** tells Acumen to count any activities where the start date is greater or equal to the period start date and less than or equal to the period finish date.

Time Phase

Use the **Time Phase** options on the Metrics tab to specify in which phase the Phase Analyzer counts the records. You can specify that records are:

- Counted in all periods that they span.
- Only counted in the period in which they start, even if they span multiple periods.
- Only counted in the period in which they end, even if they span multiple periods.

You can also use the **Date Basis** option to indicate which date will be used when inserting the record into the phase. You can select from **Scheduled** (schedule start and finish; early dates), **Late** (late start/finish dates), or **Baseline** (baseline start/finish dates).

Prorating

Prorating affects how the Phase Analyzer views records.

If the **Prorating** field is **On** (default), the Phase Analyzer on the S2 // Diagnostics tab takes time periods into account when it analyzes activities. If **Prorating** is **Off**, the Phase Analyzer ignores time periods when it analyzes activities.

Tip: If you apply prorating on the Primary Formula tab, Deltek recommends that you also apply it on the Secondary Formula tab (if you are using a secondary formula).

When you apply a metric filter to the matrix, Acumen doesn't prorate the filtered values when the data is analyzed. If you apply both a metric filter and formula, prorating will apply to the formula values but not the filter values.

Example

Activity A has a remaining duration of 40 days; 20 days in 2014 and 20 days in 2015. You create a metric to analyze activities that have a remaining duration greater than 25 days.

- If you set **Prorating** to **On**, the Phase Analyzer views the activity duration in each time period as a separate entity. The remaining duration in each time period is less than 25 (20 days in 2014 and 20 days in 2015) and the activity is not included in the analysis for any time period.
- If you set **Prorating** to **Off**, the Phase Analyzer ignores time and analyzes the activity as a whole unit. Since the activity has a total remaining duration greater than 25, the activity is included in the analysis.
  - If the Time Phase **All** option is selected, the activity is counted in 2014 and 2015.
  - If the Time Phase **Start** option is selected, the activity is only counted in 2014.
  - If the Time Phase **Finish** option is selected, the activity is only counted in 2015.

Define Thresholds

A tripwire threshold is a defined value that, if exceeded, causes a metric to be classified as "triggered." A metric can have multiple trigger points with corresponding color coding for each interval.

The threshold editor enables customizable thresholds to be defined and associated colors set. Simply put, it is a means of classifying the activity, work package, or control account results into buckets. The threshold can be shown as discrete (that is, 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.

Note: 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.

Define 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.

This option provides three types of standard scale:

- **Lowest is Better** — This creates a scale where the lowest values are preferable.
- **Highest is Better** — This creates a scale where the highest values are preferable.
- **Ideal Value** — This 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.

Note: When you see a threshold value of **<1E-05**, it means 1 to the negative 5th power. It indicates a very, very small number that is greater than zero. It is used to differentiate zero from a very small non-zero value.

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).

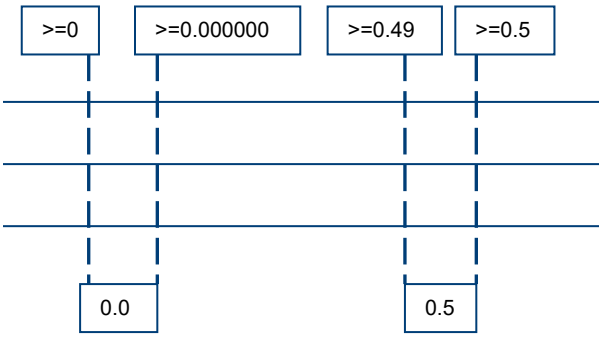
Normal scales behave in an absolute or binary manner – that is, a metric result either triggers a threshold or it doesn’t.

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

After the tripwire scales have been defined, the threshold limits must be set. Typically a different color is setup for each different limit. Since Acumen 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:

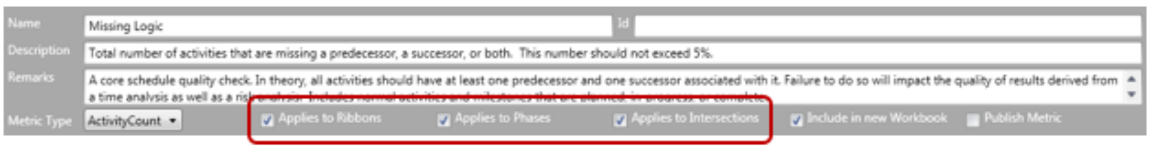


After these ranges are defined, enter them into the table to create the tripwires. If color is associated with each one of the levels, then the colors will appear on the Ribbon View after analysis.

> **Note: <1E-05** means 1 to the negative 5th power. It indicates a very, very small number that is greater than zero. It is used to differentiate zero from a very small non-zero value.

Include/Exclude 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 (for example, phase) if, for example, the context is not valid. You can include/exclude metrics from each of the three analyzers by using the three **Applies To** checkboxes.



Shortcuts and Rules

For several of the Acumen 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.

Count 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")
```

Count 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:

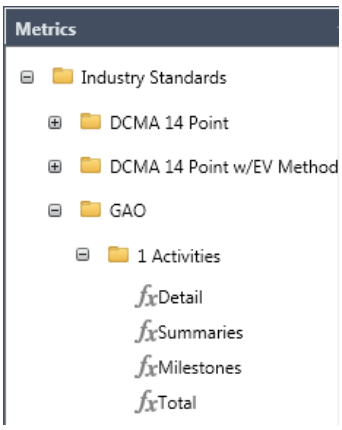
```
IF(TotalCost>BudgetCost,1,0)*IF(ActivityType<>”Summary”,1,0)
```

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

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

Publish Metrics

When you publish metrics, the report combines metrics based on the folder hierarchy on the Metrics tab. For example, if you have a **GAO** metrics folder with a **1 Activities** sub-folder on all of the Metrics forms, Acumen consolidates the data in the **1 Activities** folder on all metrics forms into a single tab when it generates the report. Each report tab is then sorted using the ID entered on the Metric form.



To ensure that all metrics are grouped correctly, you can click **Sync Groups** in the Metrics menu group to synchronize the folder structures across all of the Metrics forms (Metrics, Work Package Metrics, and Control Account Metrics).

If you don't have identical structures across the Metrics forms, the report will include more tabs since less data will be consolidated.

Definitions

Term	Definition
Primary Formula	This is the formula used to calculate the primary result calculated in an Acumen analysis.
Secondary Formula	This is usually used to show the metric as a percentage.
Tripwire Formula	This is used to determine the individual exceptions that are listed in the Activity Browser.
Secondary Tripwire Formula	The secondary tripwire formula always looks at activities, regardless of whether you are using a control account or work package metric. Its purpose is to find the activities that are causing the work package or control account an issue.
Tripwire Threshold	This is used to graphically show when a defined threshold is exceeded.
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 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. If the IF function name and Value_if_true and Value_if_false parameters are omitted, the Acumen engine will assume that the function is an IF statement returning either a 1 or a 0. For example, 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.

Special Fields

The table below lists special fields that are available to use in metric filters.

Field	Description
IsOutOfSequence	Returns a true value for any activity that is out of sequence for groups.
Number of Discrete Successors	Looks at activity successor earned value technique and counts how many are not LOE.
Number of External Predecessors	Returns the count of Predecessors that are from external projects.
Number of External Successors	Returns the count of Successors that are from external projects.
Number of FF Predecessors	Returns the count of Predecessors that are Finish to Finish (FF) relationships.
Number of FF Successors	Returns the count of Successors that are Finish to Finish (FF) relationships.
Number of FS Predecessors	Returns the count of Predecessors that are Finish to Start (FS) relationships.
Number of FS Successors	Returns the count of Successors that are Finish to Start (FS) relationships.
Number of Lags	Returns the count of Predecessor relationships with Lags.
Number of Leads	Returns the count of Predecessor relationships with Leads which are negative Lags.
Number of Predecessors	Returns the total count of Predecessors of any type.
Number of Resource Assignments	Returns the total count of Resource Assignments.
Number of SF Predecessors	Returns the count of Predecessors that are Start to Finish (SF) relationships.
Number of SF Successors	Returns the count of Successors that are Start to Finish (SF) relationships.
Number of SS Predecessors	Returns the count of Predecessors that are Start to Start (SS) relationships.
Number of SS Successors	Returns the count of Successors that are Start to Start (SS) relationships.
Number of Successor Lags	Returns the count of Successors relationships with Lags.

Field	Description
Number of Successor Leads	Returns the count of Successors relationships with Leads which are negative Lags.
Number of Successors	Returns the total count of Successors of any type.
Previous Actual Finish	This field is populated whenever a snapshot is created. ▪ When no other snapshots are present, the Actual Finish date from the base project is copied to this field. ▪ When other snapshots are present, the Actual Finish date from the latest snapshot is copied to this field. If the activity did not exist in the previous snapshot or baseline, this field is empty. This field is available as a column in the Activity grid and as a field option for activity metrics.
Previous Actual Start	This field is populated whenever a snapshot is created. ▪ When no other snapshots are present, the Actual Start date from the base project is copied to this field. ▪ When other snapshots are present, the Actual Start date from the latest snapshot is copied to this field. If the activity did not exist in the previous snapshot or baseline, this field is empty. This field is available as a column in the Activity grid and as a field option for activity metrics.

About Deltek

Better software means better projects. Deltek is the leading global provider of enterprise software and information solutions for project-based businesses. More than 23,000 organizations and millions of users in over 80 countries around the world rely on Deltek for superior levels of project intelligence, management and collaboration. Our industry-focused expertise powers project success by helping firms achieve performance that maximizes productivity and revenue. www.deltek.com