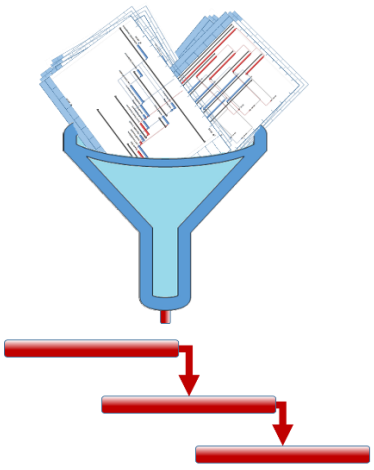
INTRODUCTION TO

BPC LOGIC FILTER FOR MICROSOFT PROJECT



Prepared For:

General Release

Prepared By:

Thomas M. Boyle, PE, PMP, PSP

BPC Project No. 15-001
First issue 29-May-15 (Updated 02-Dec-20)

Boyle Project Consulting, PLLC

Project Management and Construction Support Services

4236 Chandworth Rd.
Charlotte, NC 28210

Phone: 704-916-6765
info@boyleprojectconsulting.com

TABLE OF CONTENTS

1.0 EXECUTIVE SUMMARY

For the professional planner, scheduler, project manager, or administrator working with Microsoft Project (2010+), BPC Logic Filter is a tool that fills the gaps between Project's native functionality and the requirements of logic-driven schedule management in complex projects. BPC Logic Filter includes simple tools for examining and tracing the schedule network(s) for any given task or collection of tasks. It also offers tools for standardized reporting of a project's critical path and near-critical paths when Microsoft Project's internal reporting does not suffice. Finally, BPC Logic Filter introduces and implements several tools for advanced schedule network analysis, including (relationship) relative float, path relative float, driving path Drag, bounded network analysis, cross-project links, Gantt chart highlighting, and resource leveling drivers (i.e. resource-constrained driving path). The tool is implemented as a Microsoft Office customization (i.e. Project Add-In) and is deployed using secure Microsoft-developed technology that does not require Administrative Privileges.

.

2.0 BACKGROUND / MOTIVATION

2.1 History

Twenty-five years after its introduction to the market, Microsoft's Windows-based project scheduling software tool – which I will call MS Project, Project, or MSP here – has become ubiquitous worldwide among corporate (and government) users of the Microsoft Office suite of business software. During that time many serious project planners and schedulers have avoided the tool due to real and perceived shortcomings compared to the competition. Now most of the previous shortcomings have been addressed, and Project offers a task scheduling engine that *appears* to be generally compliant with the precedence diagramming method (PDM) and the critical path method (CPM) – two closely-related and often-specified methodologies for developing and implementing project schedules based on networks of logically-related tasks. Nevertheless, Project continues to provide only rudimentary built-in capability for logical network analysis – that is what is needed regularly to respond to schedule disruptions, and more importantly, to communicate the sequential plan behind the schedule dates.

2.2 Needs

The tool was needed to answer the following questions:

<u>For a given task:</u>

1. Which tasks are its predecessors? (The whole chain, not just the immediate dependencies.)
2. Which tasks are its successors?
3. Which predecessors are driving – i.e. effectively limiting the early start of – the task? (Also the whole chain, though identifying the immediate dependencies would be a good start. Project's Task Inspector and Task Path bar highlighters promise to do exactly this, but their results are inconsistent for non-FS relationships.)
4. Which successor tasks are being driven by the task?

5. If a predecessor is NOT driving, how much may it slip before it becomes driving.
6. If a successor is NOT being driven, how much may the given task slip before it has an impact on the successor?

For the project as a whole:

7. What is the real *critical path* and how is this related to Project's *critical* task flag? (Is it related at all?)
8. What is the longest path – i.e. that group of logically connected activities that drive the overall project completion date? (In my opinion a much better definition of critical path in the presence of deadlines and variable calendars.)
9. What is the critical/longest path for a project whose key milestone(s) don't coincide with the end of the project schedule? For example, construction schedules for industrial plants are often geared to starting production (and generating revenue) as soon as possible, while permanent administration and maintenance facilities may be completed later. A simple longest path doesn't seem to reflect the true schedule priorities.
10. Finally, how can we identify and differentiate parallel critical paths and near critical paths from each other based purely on the schedule data?

3.0 PROGRAM DEVELOPMENT

3.1 Feature Development

Over several years, the following features were developed:

1. Started with a simple logic tracer and filter for all dependencies of a task or group of tasks. (Answers questions 1 and 2 on the Needs list.)
2. Added limits for overall number of tasks to be checked (for terminating an endless loop during development) and the number of logical steps away from the selected task (i.e. *local network only*).
3. Added analysis and filtering for driving relationships only, based on *relationship relative float*. The concept is very simple: for a successor task in a relationship, what is the difference between the computed (early) date of the task and the earliest date that can satisfy the constraints of the relationship.[1] A value of zero means that the relationship is *driving.* (Answers questions 3 and 4 on the Needs list.) A value greater than zero reflects the amount of time that the predecessor may slip before it becomes one of several concurrent drivers of the successor task. Values less than zero are not encountered except in cases of unresolved logical conflicts. Figure 1 illustrates the relationship relative float computed for each relationship in a simple schedule network. Figure 2 shows that path which includes only driving relationships among the

[1] According to Dr. Gui Ponce de Leon (2010 PMI Global Proceedings), what I've called relationship relative float is essentially identical to the "link lag" originally described by Fondahl for FS relationships and subsequently established as "link gap" by Ponce de Leon for other relationship types. Link gaps are regularly computed by Ponce de Leon's Graphical Planning Method (e.g. PMA Netpoint.) Ron Winter (www.ronwinterconsulting.com) defined a similar quantity that he termed (relationship) "slack" in some papers aimed at Primavera users. While the word "slack" in American usage seems intuitively more appropriate for relationships, MSP already uses it in a distinctly different context. I've stayed away from it. A similar quantity in Oracle's Primavera Project Management P6 is called Relationship Successor Free Float.

predecessors of the last task of the schedule (*Task9*) – this is Task9's *driving path*.

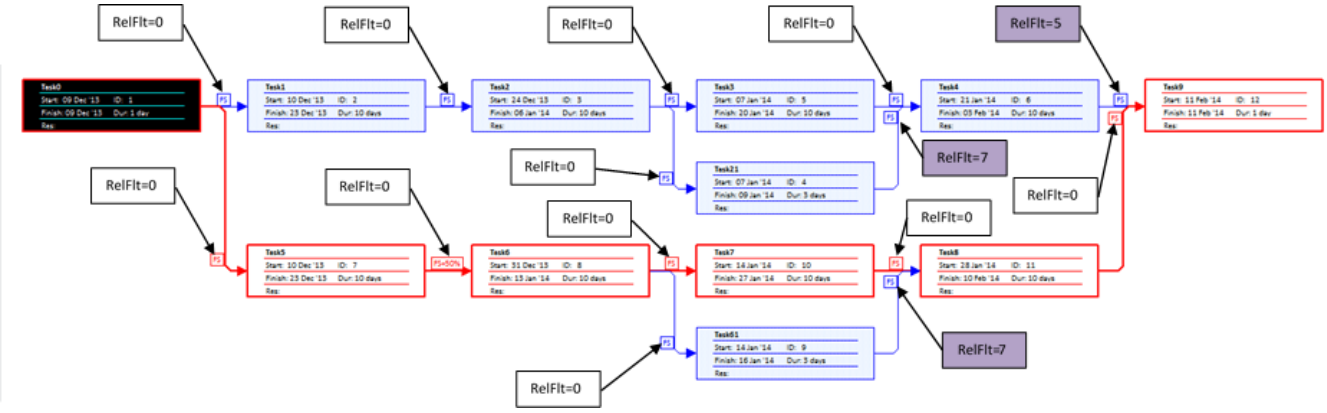


Figure 1. Relationship Relative Float in a Simple Project Schedule Network

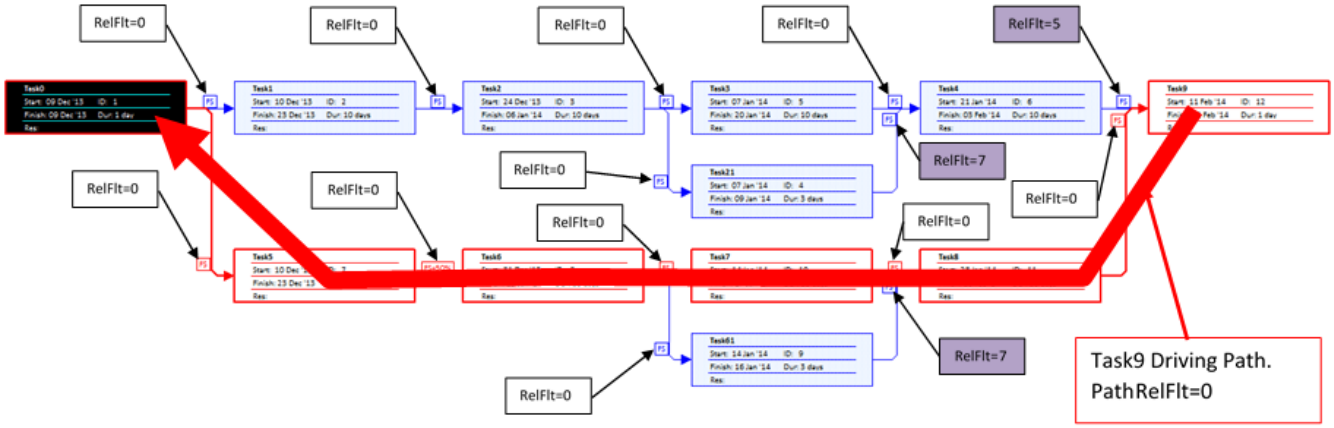


Figure 2. Driving Path Relationship Relative Float = 0

4. Introduced Path Relative Float, which aggregates relationship relative float to account for all the tasks and relationships between any two tasks in the network. For a certain *Selected* task, all of its predecessors (and successors) may be identified and grouped according to their path Relative Float values. Those predecessors with a value of zero represent the *driving paths(s)* for the selected task. (Answers questions 5 and 6 on the Needs list.) Those with low non-zero values represent the *near-driving paths* for the selected task. By starting the analysis from the last task in the project or from the key project completion milestone, the *driving path* becomes the *critical path* for the project according to the most commonly-applied *longest path* definition. (Answers questions 8 and 9). Applying special sorting criteria can also differentiate any concurrent- or near-critical paths. (Answers question 10.)[2] Figure 3 illustrates the Path Relative Float (relative to Task9) for three non-driving paths of the simple schedule network.

[2] For a project critical path or longest path application, Ron Winter's "Longest Path Value" seems to play a similar role as the more general "Path Relative Float" does here. The same could be said of Total Float and Forensic Total Float in the Graphical Path Method (e.g. PMA Netpoint.)

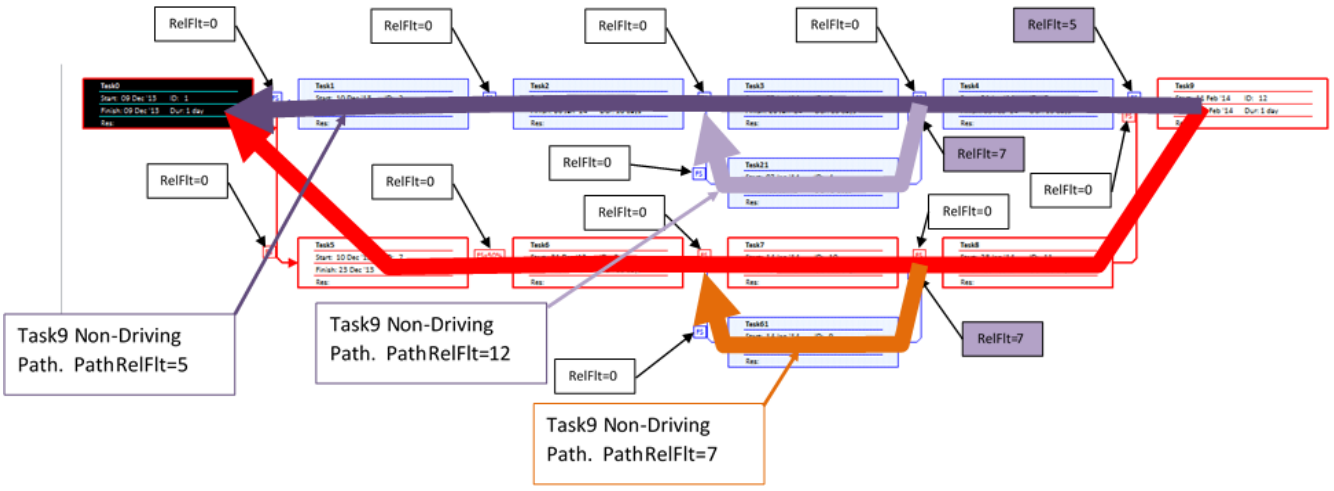


Figure 3. Path Relative Float for Driving and Non-Driving Paths

5. Introduced Longest Path Filter as a special application. This automatically selects the last task(s) in the project schedule and identifies the associated driving path(s).
6. Introduced Local Network Filter as a special application to facilitate task-jumping through the network a few steps at a time.
7. Introduced the Target Task concept to support finely bounded impact analysis. For example, given the prospect of a delay on one task, what will be the impact on another related but more important task, and where are the opportunities for recovering the delay between the two? Here the analysis is truncated to show only those logic paths connecting the two tasks (one the *selected task* and the other the *target task*), with the zero-Relative Float values presenting the first opportunities for recovery. Figure 4 demonstrates the concept for the simple schedule network introduced above, where the grayed-out area is automatically omitted.

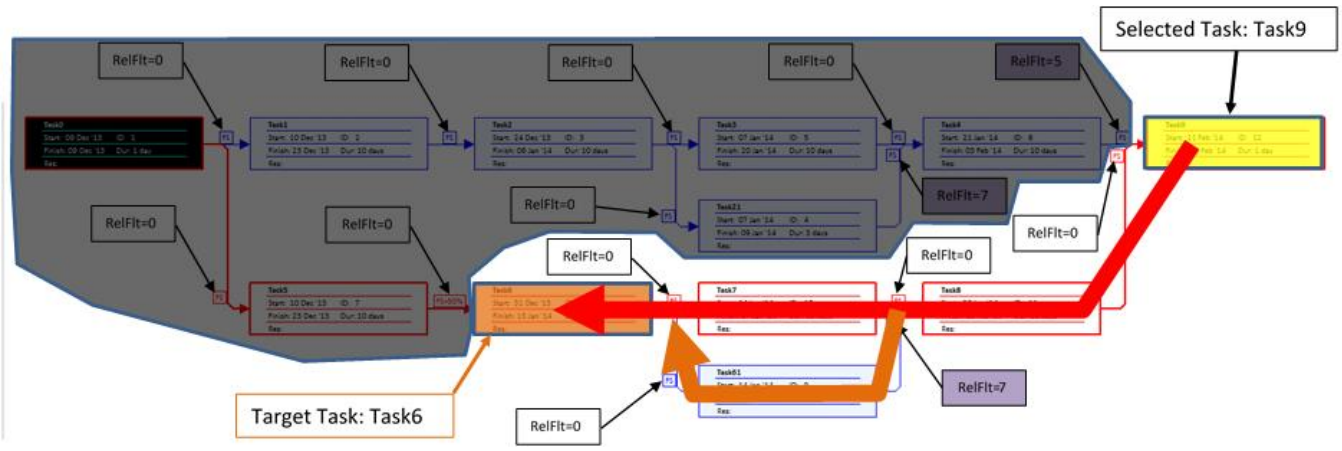


Figure 4. Bounded Analysis of a Simple Schedule Network

8. Added Cross-Project Links to apply logic tracing and all the other features across master- and sub-project boundaries. Adding this functionality was technically quite challenging, but it seems invaluable for those who are forced to rely on these powerful but problematic MS Project features.

9. Added driving path Drag as an advanced feature for prioritizing a schedule crashing process. Critical path Drag was introduced in 1999 as a formal metric to be applied to critical path tasks/activities only[3]. In that context, Drag represents the amount of time that each task contributes to the critical path of the Project. In simple terms, "how much could the critical path be compressed by eliminating this task?" It is computed for each CP task as either a) the duration of the CP task, or b) the lowest Total Float of any parallel task – whichever is less. Figure 5 is a simple network logic diagram with drag computed as originally presented to illustrate the concept. BPC Logic Filter adapts the concept for general use with respect to the *driving path* for any selected task, not just the last task in the network. In this context, the Drag for each task (on the Driving Path of a given selected task) is either its own duration or the lowest Path Relative Float of its parallel tasks, whichever is less. Figure 6 illustrates the concept as adapted for our own simple network schedule introduced above. In this figure, the Drag of Task8 (Dur =10 days) is limited to 5 days by the RelFlt=5 of the parallel path. Similarly, the Drag of Task7 is initially limited to 7 days by the RelFlt=7 of the single parallel task (Task61). Ultimately however the Drag of Task 7 and its two predecessors are limited to 5 days by the same parallel path that limited Task 8. In other words, the current *Critical* Path may be shortened a maximum of 5 days before the secondary path becomes critical.

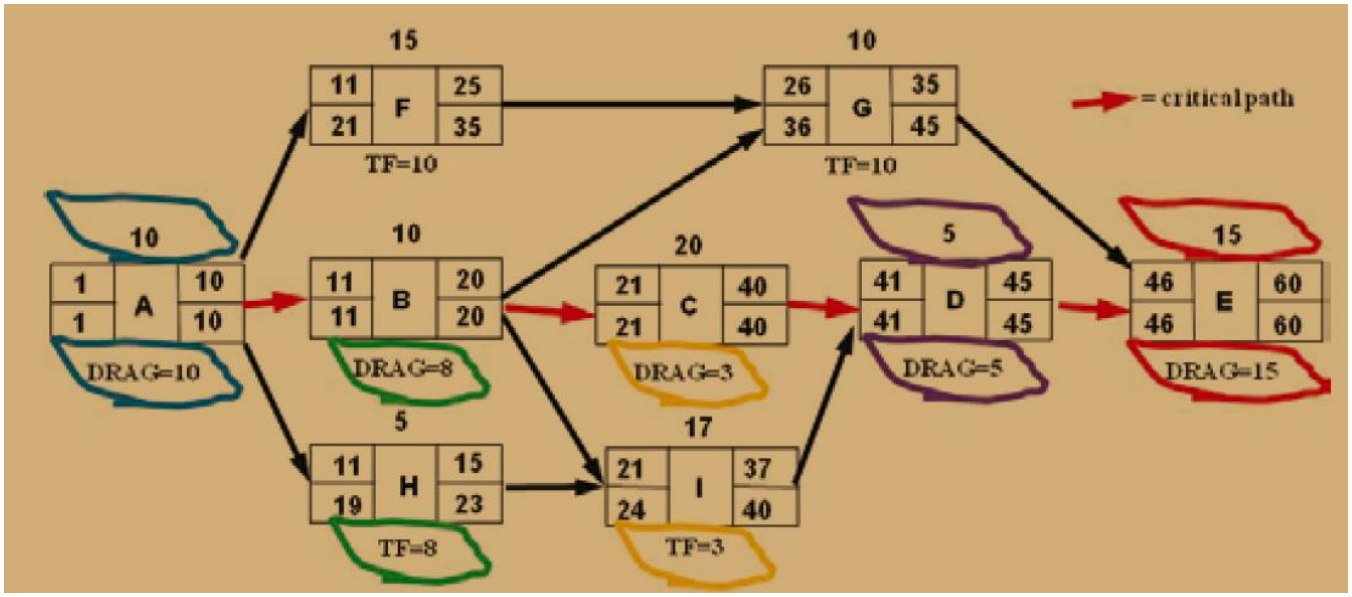


Figure 5[4]. Drag Definition

[3] See Devaux, Stephen A., Total Project Control: A Manager's Guide to Project Planning, Measuring and Tracking, 1999, John Wiley & Sons

[4] Figure taken from The Drag Efficient: The Missing Quantification of Time on the Critical Path, by Stephen A. Devaux, Defense AT&L: January–February 2012

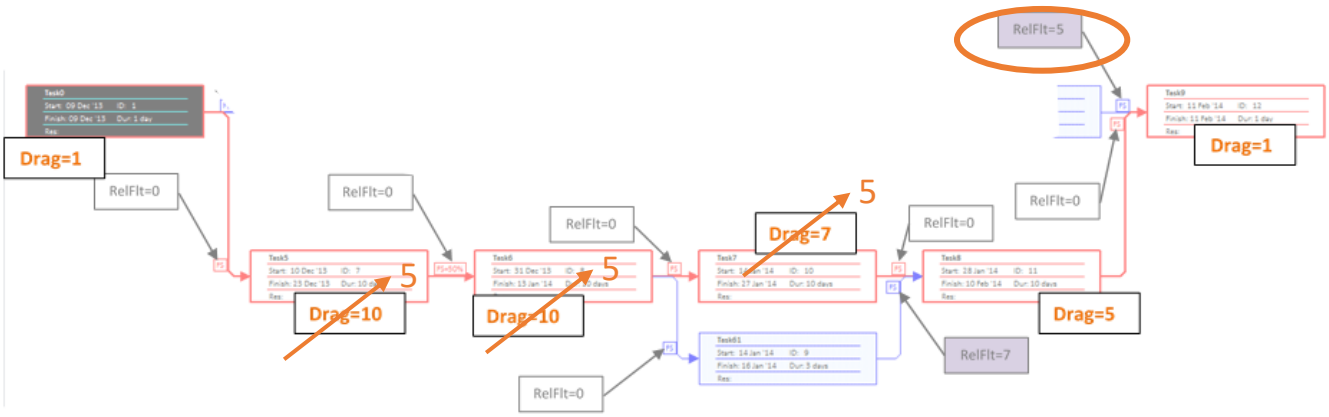


Figure 6. Drag Analysis for a Simple Schedule Network

10. Added a feature to analyze resource leveling drivers. In normal (forward) scheduling, MSP's resource leveling engine acts to delay lower-priority tasks in favor of higher-priority tasks demanding the same resources. BPC logic filter examines the leveling delays and assignment details to infer resource-driven, implied logical relationships between tasks that demand the same resources. As a result, the user can easily determine the Resource-Constrained driving path for any task in the project. Incorporating the feature in the Longest Path Filter reveals the resource constrained critical path for the overall project.

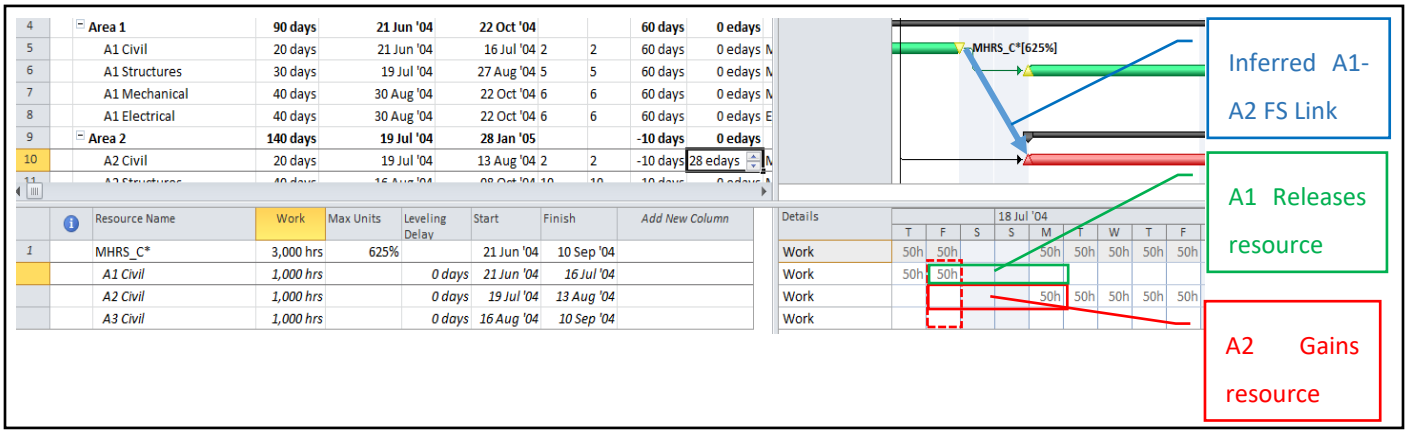


Figure 7. Inferred Links from Resource Leveling Drivers

11. In concert with the identification of resource drivers, added bar-coloring methods to clarify their display. These methods also allow presentation of logic traces within the context of the overall project schedule, not limited to filtered views.

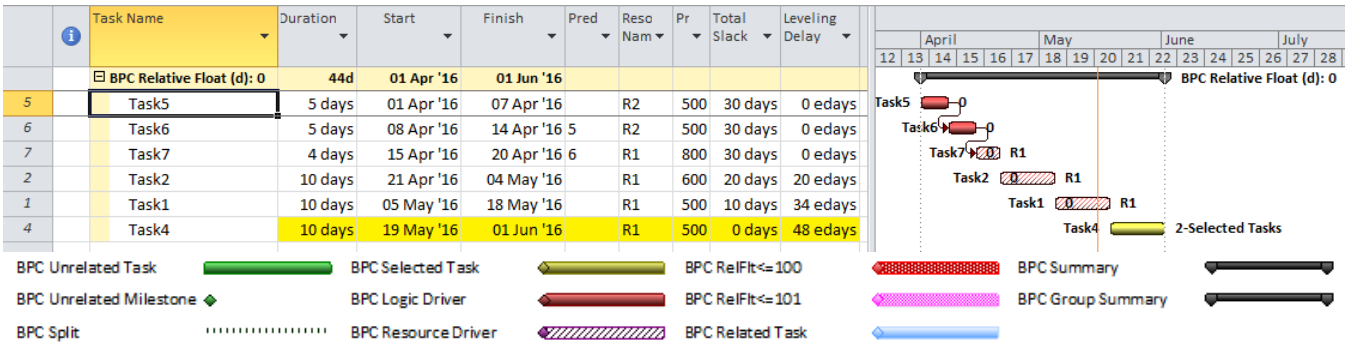


Figure 8. Resource Drivers and Bar Coloring Example

12. Added the Project Logic Checker function. This provides a method for precisely-controlled review and summary of various logic issues and topics of interest in the schedule.

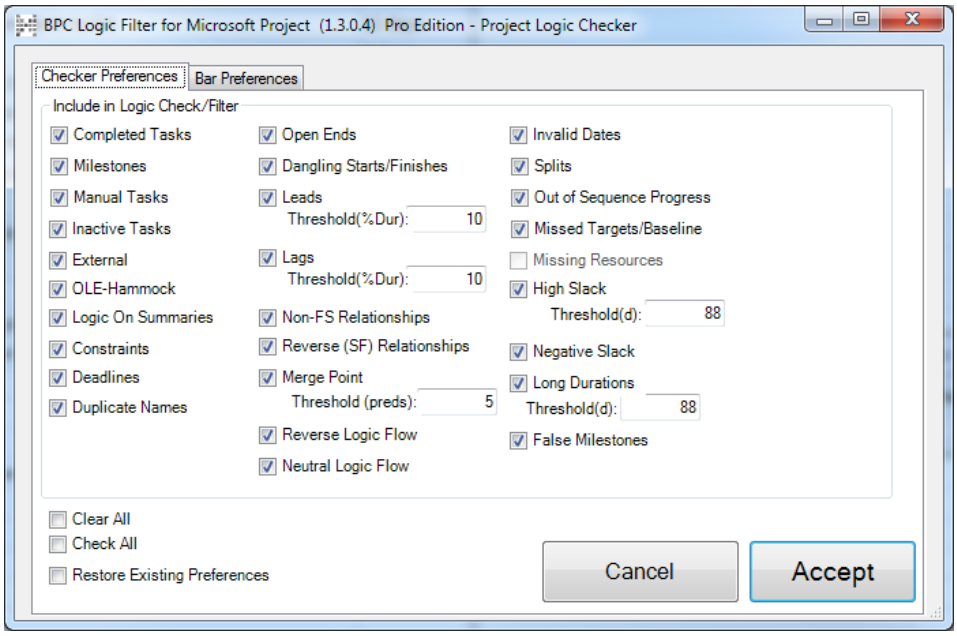


Figure 9. Project Logic Checker Preferences

The analysis concludes with the presentation of a dynamic output form that…

- Summarizes the findings on the face of the form;
- Provides a simple report (in the text window) of the details of the analysis, including a complete project logic statistical summary;
- Provides a dynamic filter controller (using simple checkboxes) to display only the tasks of interest for particular logic issues.

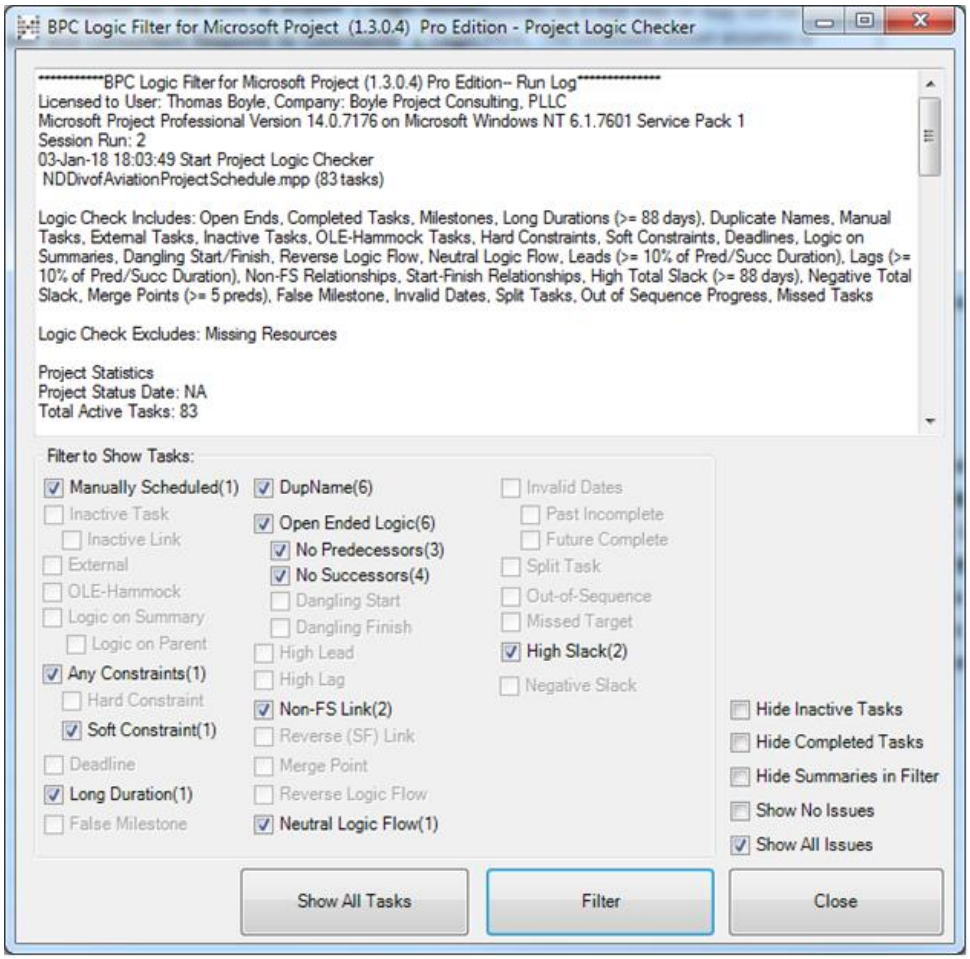


Figure 10. Project Logic Checker Dynamic Filter Control Form

If specified by the user, a detailed tabular report of findings is automatically generated and stored.

13. Added the Logic Quick Checker function. This provides a method for one-click examination of key project schedule health metrics, with only minimal user input required.

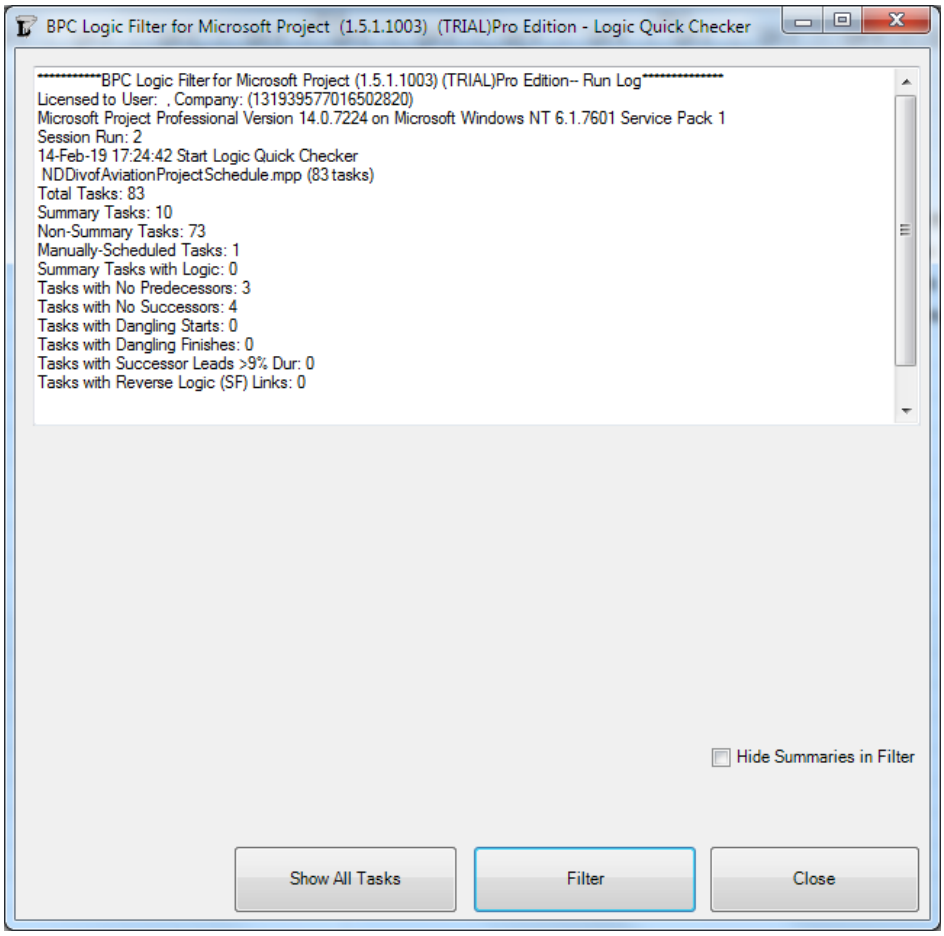


Figure 11. Logic Quick Checker Output Form

14. Later added a Task Logic Inspector. This function provides data-rich interface windows for examining schedule logic relationships – typically verifying the Tracer/Filter results on a task-by-task basis. In addition to the logic information, the Logic Inspector windows provide *jump* buttons to navigate through the schedule based solely on logical relationships. The logic inspector is the most-used tool in all of BPC Logic Filter, for good reason.

    Added the internal start-driver flag (as an asterisk) for task relationships that MSP identifies as driving relationships in the Task Inspector and the Task Path bar styles. Sometimes these do not agree with the driving relationships (yellow highlighting) derived using relative float, which are more correct in our view. The indicator makes it easy to see the differences.

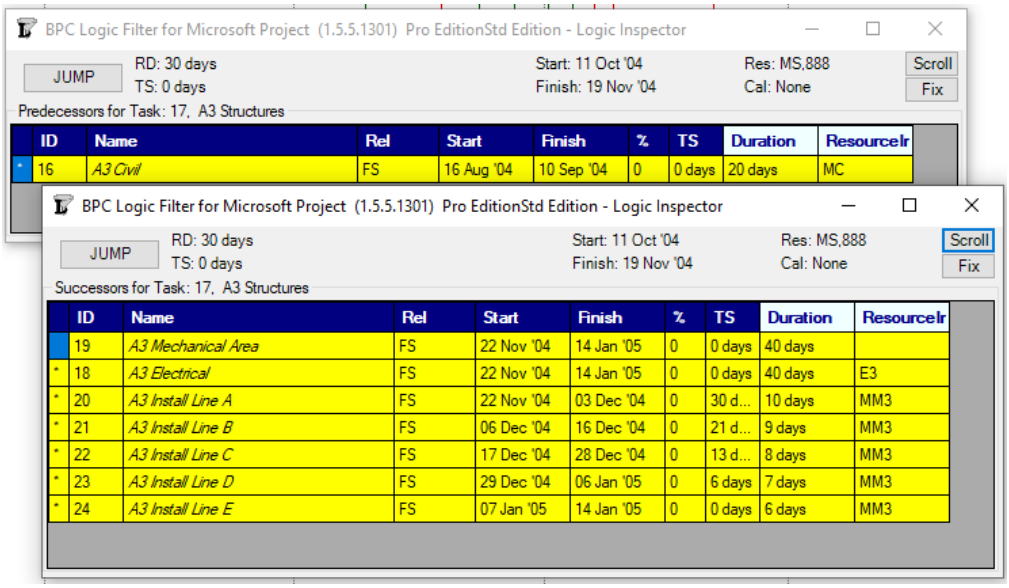


Figure 12. Task Logic Inspector Windows (Std Edition)

Added display of resource drivers and late-dates relative float (LRF). The latter helps identify late-driving and bi-directional-driving relationships, which are most useful for prioritizing successor logic paths in complex schedules. Figure 12 and Figure 13 demonstrate the impact of this capability by inspecting exactly the same task in a resource-leveled schedule using two different sets of parameters.

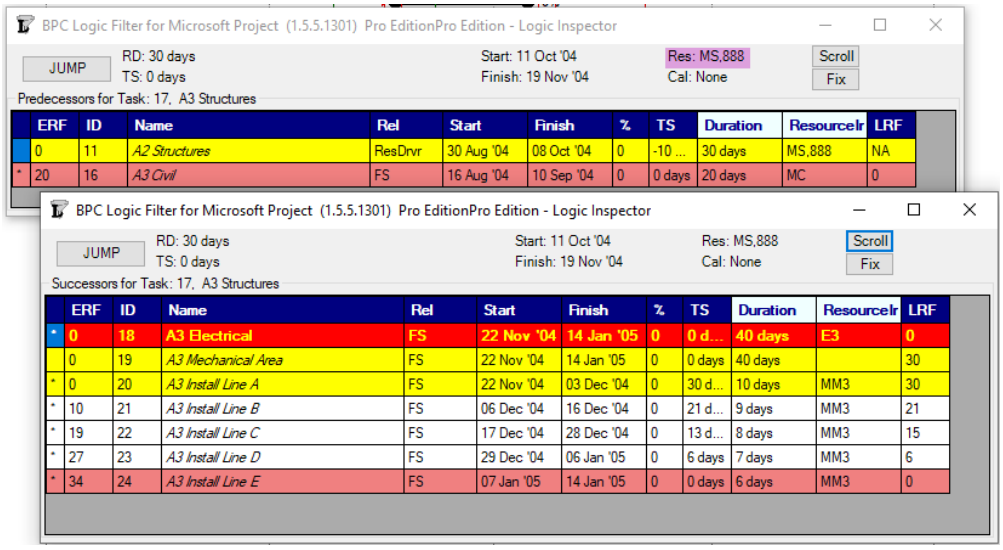


Figure 13. Task Logic Inspector Windows (Pro Edition, w/ LRF, w/ Resource Drivers)

In response to user requests, added driving hierarchical relationships. These relationships help to explain the bases of summary task start and finish dates (as *rolled-up* – *RU* – predecessor relationships from their sub-tasks). They also help to explain the bases of sub-task dates when their parent summary tasks have assigned predecessors or constraints (as *rolled-down* – *RD* – from predecessor summaries.)

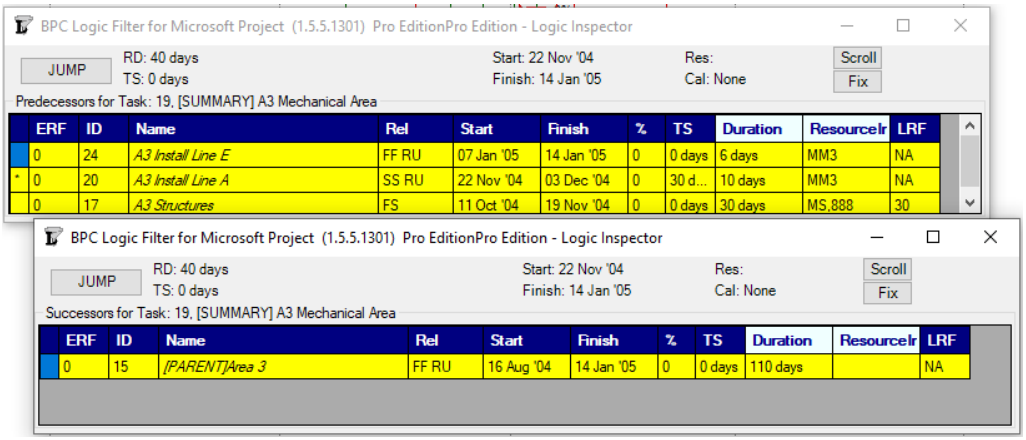


Figure 14. Task Logic Inspector Windows (Pro Edition, with Hierarchical Drivers)

In response to user requests, incorporated a stripped-down Task Logic Inspector for users who are less concerned with driving logic flow. Here the focus is on consolidating predecessor and successor schedule information for visual review – and, of course, logic navigation using the Jump button.

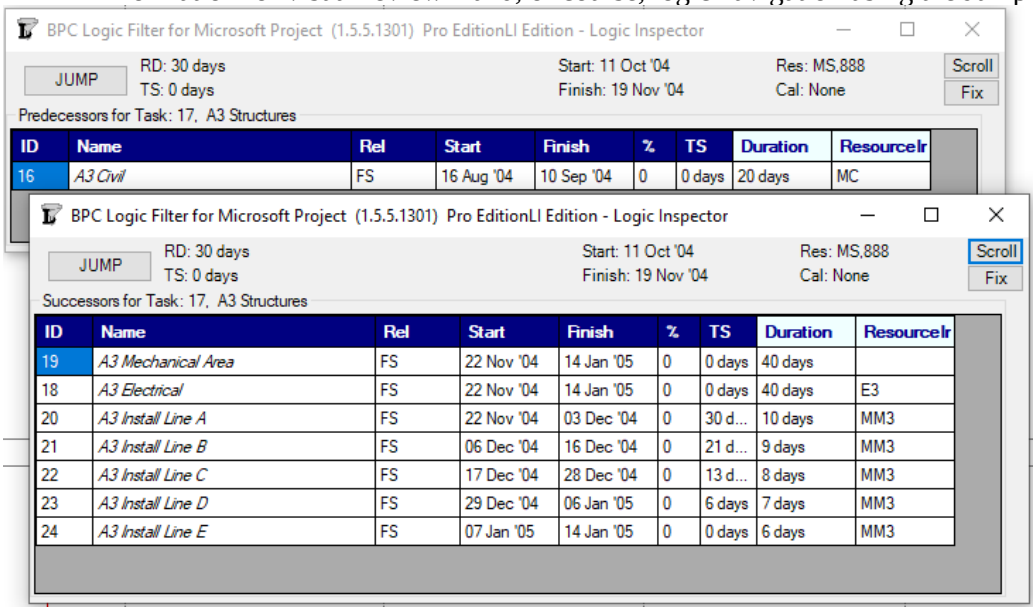
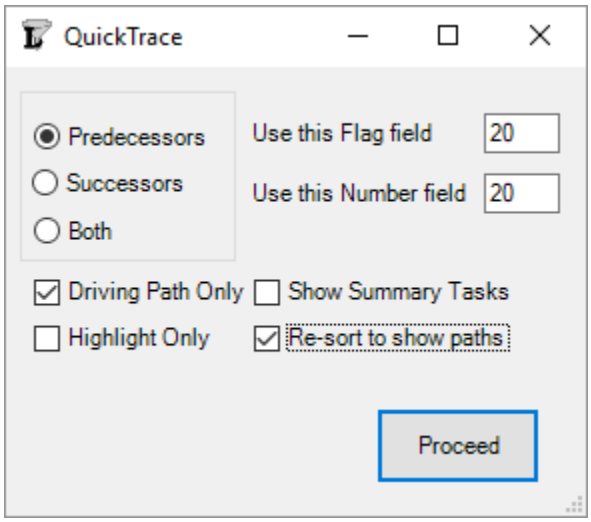


Figure 15. Task Logic Inspector Windows (LLI Edition)

15. Integrated the QuickTrace macros. This collection of simple programs – which I originally wrote and shared in a weekend blog entry – performs quick filtering, highlighting, and sorting of schedule tasks. QuickTrace results are 100% consistent with the results of MSP's *Task Inspector* tool and its *Task Paths* bar styles. Comparing these results with those obtained using the Task Logic Tracer can be enlightening, especially when *driving* relationships are examined.



The attentive reader may notice I did not add any features to answer Question 7 on the Needs list: *What is the real "critical path" and how is this related to Project's "Critical" task flag? (Is it related at all?)* I don't think there is a good answer except for relatively simple project schedules. With BPC Logic Filter, I don't rely on Project's Critical flag (or total slack) anymore.

3.2 Sharing the Tools

BPC Logic Filter was deployed for initial (internal) testing as a Project Add-In in April 2015. The first beta-testing began at client sites in North America in May 2015, and we actively sought other beta test sites to validate the software before proceeding with general deployment in Q3-Q4 2015. From that time through the beginning of 2019, BPC Logic Filter remained non-commercial software that we freely licensed to clients, other qualified professionals, and advanced academic users. Substantial improvements were made during this period, often in response to diverse user needs. Beginning in Q1 2019, we made the tool generally available for downloading, trial operation, and commercial licensing.

A Note on Language Packs: BPC Logic Filter uses English as its sole language for user interface, but it has been adapted for use with Microsoft language packs for a number of different locales. New language packs are continually added in response to specific user requests.

4.0 USER APPLICATIONS AND SETTINGS

A User Guide, separate from this document, provides detailed instructions related to the user interface, applications, settings, and interpretation of program output. The User Guide is made available to trial users and registered users after installation of the software.

5.0 SOFTWARE EDITIONS AND LICENSING

BPC Logic Filter is targeted for per-user licensing in Standard and Professional Editions, with the latter being

required for more advanced functions. These editions are NOT related to the Microsoft Project versions; both editions work with any modern desktop version of Microsoft Project. A Limited Logic Inspector (LLI) Edition is also available. Users who evaluate a Trial Version but elect not to upgrade the software retain limited (i.e. Free Edition) functionality. See Table 1. In general, the Free Edition includes elementary logic tracing capability along with the general display options.

The LLI Edition adds the basic Task Logic Inspector functionality, including predecessor and successor tables with user-selectable data columns and Jump buttons for logic-based navigation. This edition does not include advanced filter handling or logic analysis.

The Standard Edition adds the ability to permanently save Tracer results, to show only driving/driven logic paths (with constraint delays), and to use hourly schedules in all Tracer functions. In addition, the Standard Edition includes the pre-programmed Logic Quick Checker and the QuickTrace, Longest Path, and Local Network tracers/filters. Finally, the Standard Edition adds conventional and hierarchical driving logic analysis to the Task Logic Inspector windows, re-sorting and flagging driving relationships accordingly. The Standard Edition is aimed at the typical project manager, scheduling coordinator, or administrator whose duties may include schedule development and review (for simpler projects) or generating standard reports requiring these functions (for complex projects.)

The Professional Edition adds the more advanced functions described in this document, including relative float analysis (with sophisticated calendar handling), late-date and alternative-calendar relative float, cross-project links, bounded network analysis, driving path drag, resource leveling drivers, and Gantt bar highlighting. The Professional Edition is aimed at the professional project planner/scheduler or scheduling consultant who requires a much more comprehensive and in-depth understanding of the scheduling logic network.

6.0 SOFTWARE DEPLOYMENT AND SECURITY

BPC Logic Filter is securely published to a Microsoft Azure cloud server for deployment as a self-updating ClickOnce application. (ClickOnce is a Microsoft technology developed for this purpose.) The download/installation process is seamlessly integrated with Microsoft's Internet Explorer and Edge browsers, and downloading using other browsers is not recommended. By default, ClickOnce applications are installed as standalone additions to a Windows User account. Such installations normally do not require administrative privileges, but such limitations can be imposed by corporate security policy.

Right after installation, users will be prompted to accept the End User License Agreement and to validate or change a selected folder for storing logs and other files. Rejection of the EULA may lead to unreliable results.

During operation, the software requires internet connectivity for two purposes:

- At each startup it checks for updates at the deployment server.

- At various times during operation it communicates with a license service to verify permissions for the advanced functions.

Feature	Free Edition	LLI Edition	Std Edition	Pro Edition	Remarks
Fully-functioning trial version.	✓	✓	✓	✓	All three editions may be evaluated. After expiration of the trial, Free edition features will remain available.
MAIN FEATURES					
Task Logic Tracer	✓	✓	✓	✓	Builds a filter of logically-connected tasks.
Restore Last View	✓	✓	✓	✓	Remembers previous non-BPC view attributes.
Local Network Filter	Limited	Limited	✓	✓	A step-limited fragnet.
Task Logic Inspector		✓	✓	✓	Two tables of linked task details; with link-jump buttons. (Includes relative float and driving logic depending on Edition.)
Logic Quick Check			✓	✓	Expedited review of basic task logic issues.
QuickTrace			✓	✓	Fast and simple. Driving logic provided by MSP.
Longest Path Filter			✓	✓	Driving path to project completion.
Project Logic Checker				✓	Detailed review and dynamic filtering of logic issues.
Near-Longest Path Filter				✓	Driving and near-driving paths to project completion.
ANALYSIS / DISPLAY OPTIONS					
General tracer options					
Trace Predecessors, Successors, or Both	✓	✓	✓	✓	
Display Tracer results using Highlight OR Filter	✓	✓	✓	✓	
Show/Hide Summary & Completed Tasks	✓	✓	✓	✓	(In Filter)
Re-sort tasks to show paths	✓	✓	✓	✓	(In Filter)
Display an Output Form	✓	✓	✓	✓	Copy/paste image or log for documentation.
Trace Summary Logical Dependencies	✓	✓	✓	✓	
Trace Summary Parent/Child Dependencies	X	X	X	X	
Data persistence			✓	✓	If selected, saves results in custom fields for user analysis/reporting. Needed for dynamic filters in Checkers and for accelerated color-coding of bars.
Driving logic – standard definition			✓	✓	Used in Tracers. Computed from (early-date) relationship relative float.
- Override if successor is delayed by constraint			✓	✓	For complete longest path.
- Allow partial-day lags/slips/float			✓	✓	Adjusts for typical updating errors.
Driving logic – Hierarchical (Beta)			✓	✓	In Logic Inspector: Identifies sources and consequences of rolled-up and rolled-down schedule dates
Driving logic – late & bi-directional				✓	In Logic Inspector: Identifies forward & backward drivers
Relative float (Early/Late/Alt)				✓	For multiple float path analysis, and for enhanced displays on task logic inspector.
Resource logic analysis				✓	Identifies resource drivers (availability, leveling). Incorporates combined resource/task calendars – in Tracers, Checkers, and Inspector
Color-coded bars & links				✓	Autoformats bar chart view. Highlights resource drivers in Tracers. Clarifies inline logic displays.
Master/sub project links				✓	Traces and jumps logic links between projects.
Bounded network analysis				✓	Traces logic only between specified tasks, i.e. *selected* task(s) and *target* task.
Driving path drag				✓	

Table 1. BPC Logic Filter – Feature/Edition Comparison