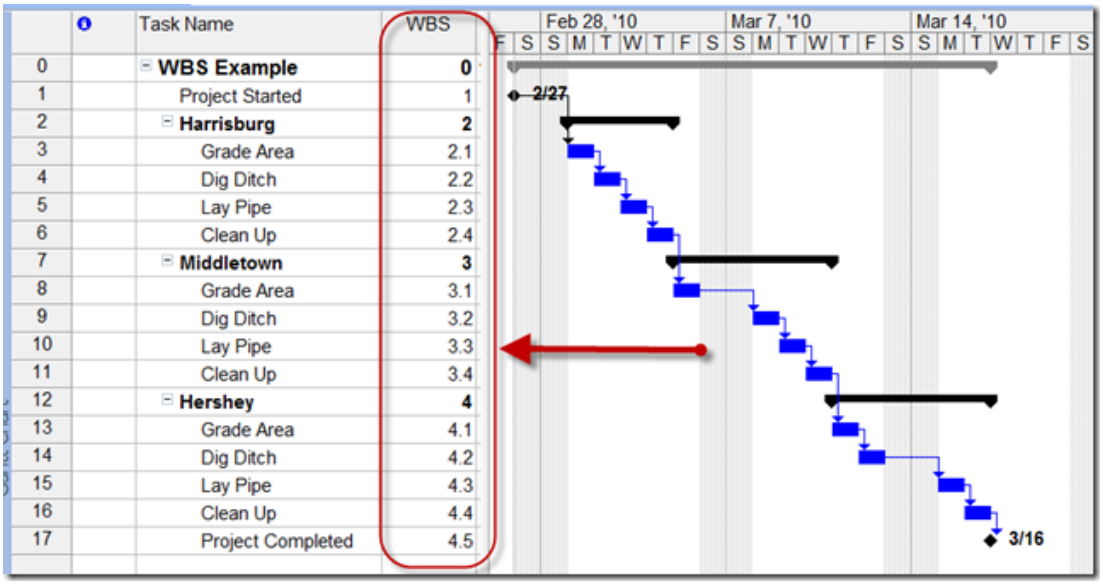
Using Intelligent WBS Codes in Microsoft Project



Andrew Lavinsky March 3, 2010

This is a question that's come up multiple times with folks I've worked with, and is something I consider best practice. The last time someone asked me about this, I figured I'd just blog it up with screenshots and send him the URL. So Daniel, this post is for you….

For years Project has had a function which remarkably few people know about, and even fewer people use – that is the intelligent WBS Code. The intelligent WBS code basically entails swapping out the default numerical outline (1.2.3) with a descriptive outline of the task (ABC.DEF.GHI). There are two main functions of intelligent WBS codes:

1) To clearly identify tasks that have the same or similar names. Often schedules are populated using fragments of templates. For instance if I am a professional gaspasser and building a pipeline with multiple compressor stations, I'll find an old template for compressor stations, then string the tasks into my schedule at various places. The result of this practice is that I'll have multiple tasks with pretty similar names. The intelligent WBS is an easy way to differentiate those tasks in an easy and intuitive fashion.

2) Many organizations need to slice and dice their data in a number of different ways. For instance, in the following example I am laying pipe in three locations: Harrisburg, Middletown, and Hershey. I have similar tasks (grading, ditching, laying, cleaning) for each of the areas. I want to sort by geographic area –or by

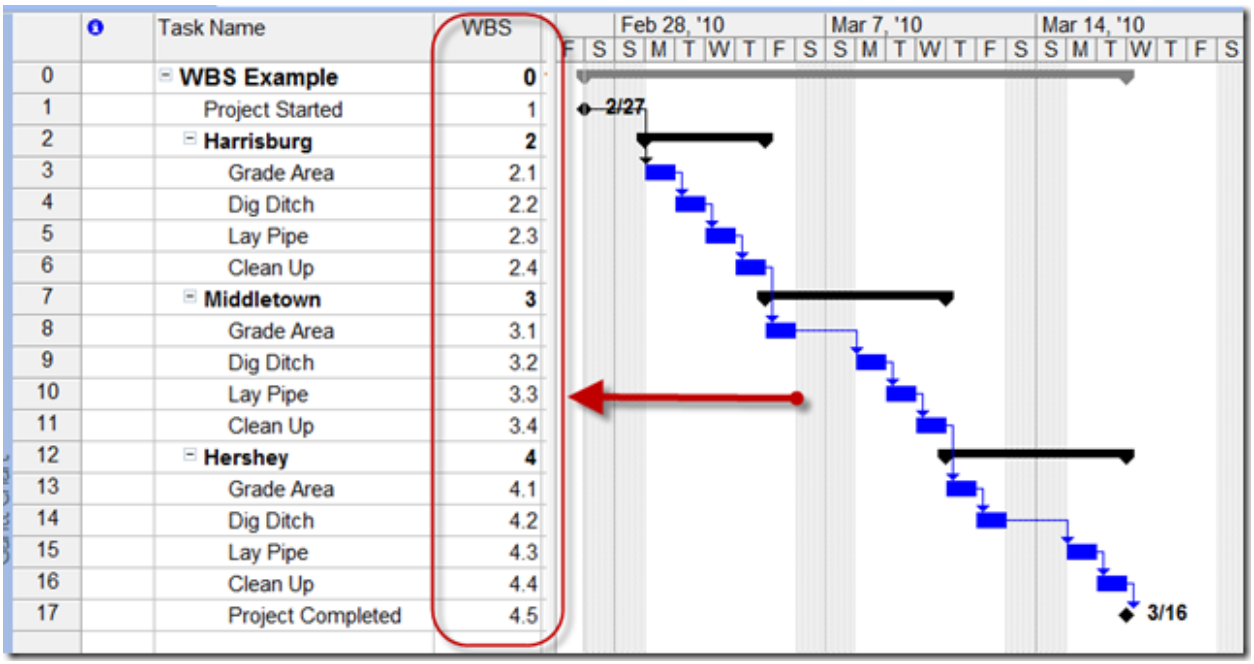
subsystem – or by subsystem and then geographic area. That is a pretty typical request from project schedulers that is enabled through the use of the intelligent WBS code.

A lot of schedulers end up creating custom fields to capture similar data and code each new activity with the appropriate metadata. That is definitely a good practice. From a tool perspective though, I've always felt the WBS code to be a slightly better option to do this as 1) it's embedded in the system out of the box and 2) the WBS Code allows the user to simply enter data on the summary task and have the data cascade down to the child tasks automatically. So from a desktop tool user perspective, the built-in WBS code provides a bit more ease of use.

There's plenty of documentation out there in the help screens to assist users in using the WBS Code. My plan with this post is to cover the basics, as well as including a couple of custom formulas to strip different levels out of the WBS Code, and then discuss implications of using the WBS Code in conjunction with Project Server.

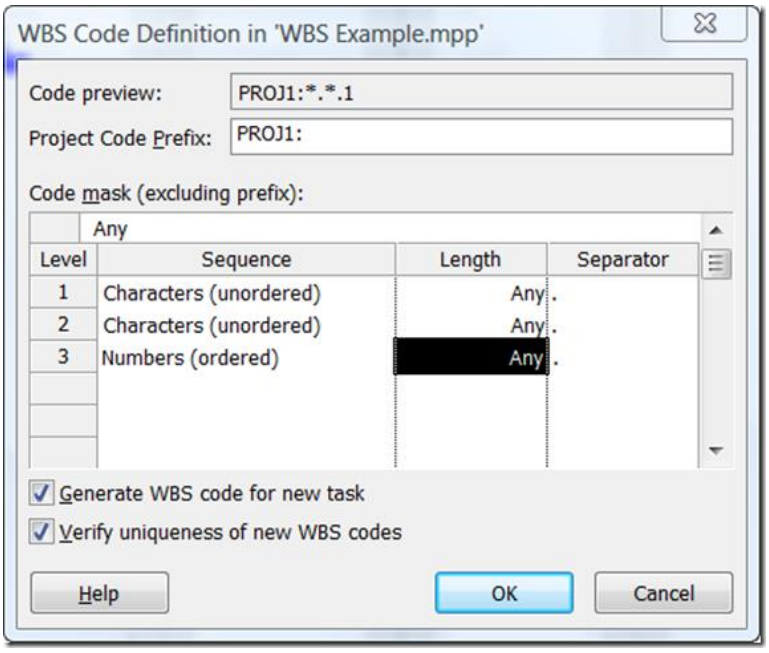
How To

This feature has been in the product as long as I've been using it. The default behavior of the WBS Code is to display an outline structure for the project.

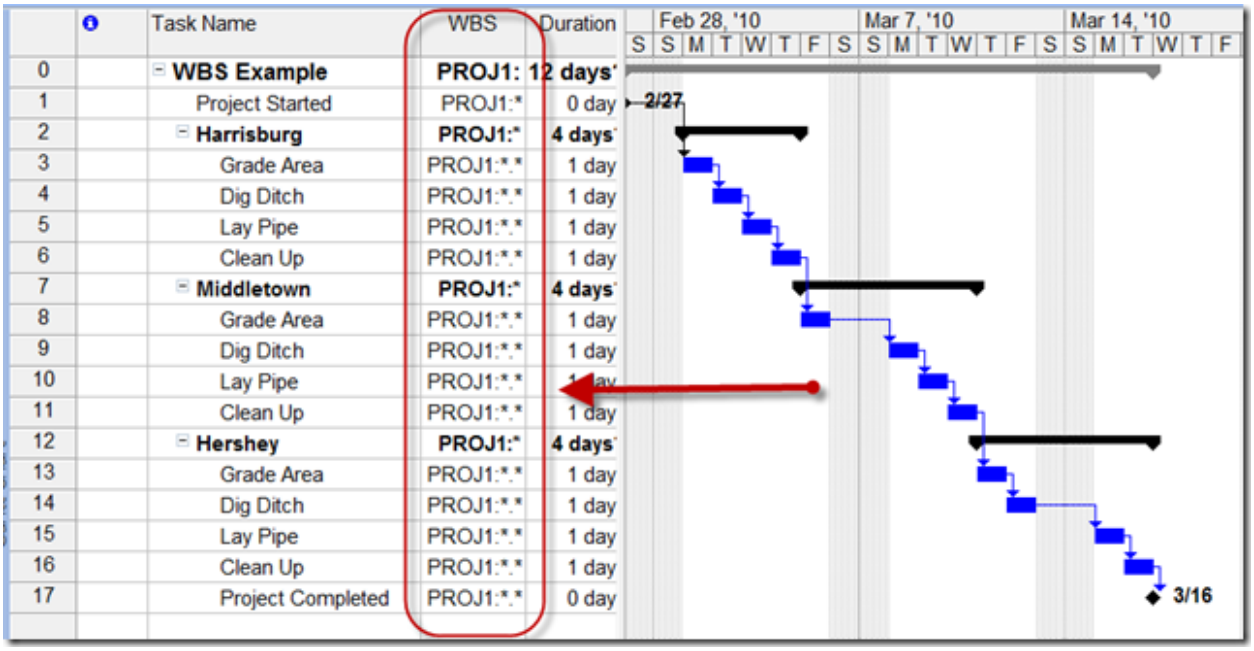


However, this behavior can be modified using the Define WBS Code Dialog Box found under the *Project > WBS > Define Code* menu structure. By using this dialog box, we can easily create intelligent WBS Codes using text that clearly defines each of the activities.

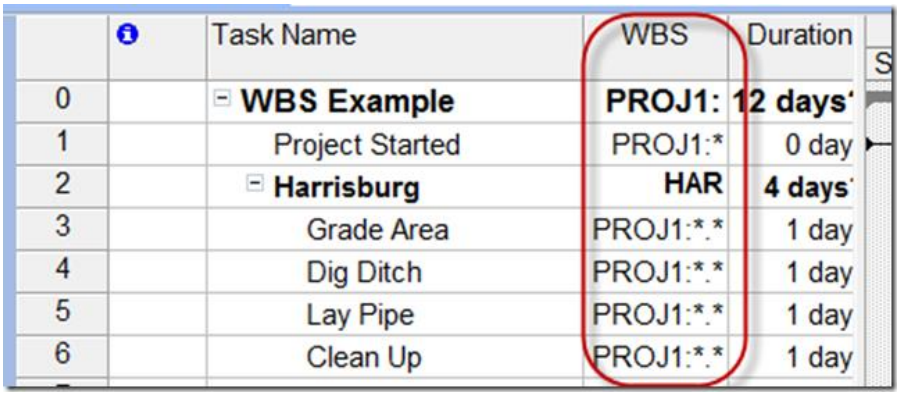
In this case, I configure the Define WBS Code Dialog Box as follows:



The result is that my project now looks like this:

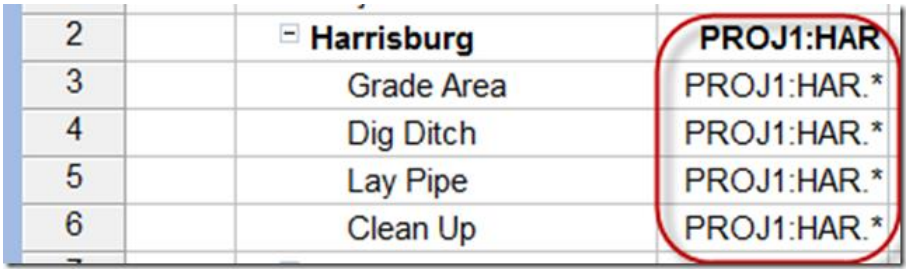


Now we can see the ease of using the WBS Code. All I have to do to populate this code is select the summary task, click on WBS, and simply type whatever value I feel is appropriate.



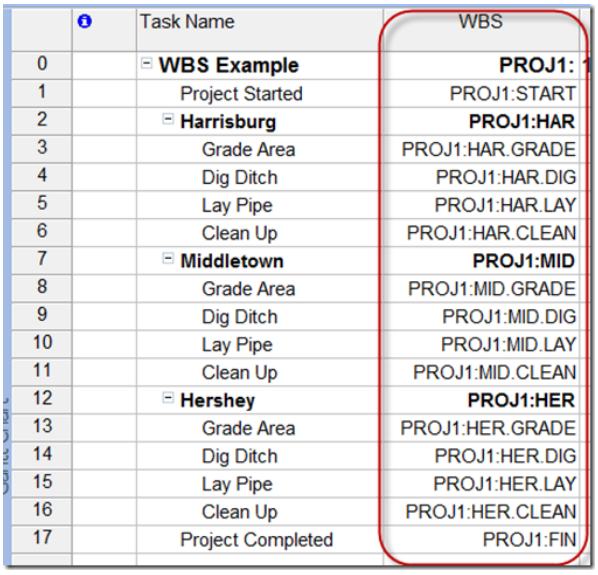
	Task Name	WBS	Duration
0	**WBS Example**	**PROJ1:**	**12 days**
1	Project Started	PROJ1:*	0 day
2	**Harrisburg**	**HAR**	**4 days**
3	Grade Area	PROJ1:*.*	1 day
4	Dig Ditch	PROJ1:*.*	1 day
5	Lay Pipe	PROJ1:*.*	1 day
6	Clean Up	PROJ1:*.*	1 day

There's no need to position the cursor in the Input Bar, select text or anything. Just click on the box and type the code for that summary task. The program takes care of the rest. All child tasks automatically take on the same value.



2	**Harrisburg**	**PROJ1:HAR**
3	Grade Area	PROJ1:HAR.*
4	Dig Ditch	PROJ1:HAR.*
5	Lay Pipe	PROJ1:HAR.*
6	Clean Up	PROJ1:HAR.*

Now, I'll go ahead and add the other codes to the project:



	Task Name	WBS
0	**WBS Example**	**PROJ1:**
1	Project Started	PROJ1:START
2	**Harrisburg**	**PROJ1:HAR**
3	Grade Area	PROJ1:HAR.GRADE
4	Dig Ditch	PROJ1:HAR.DIG
5	Lay Pipe	PROJ1:HAR.LAY
6	Clean Up	PROJ1:HAR.CLEAN
7	**Middletown**	**PROJ1:MID**
8	Grade Area	PROJ1:MID.GRADE
9	Dig Ditch	PROJ1:MID.DIG
10	Lay Pipe	PROJ1:MID.LAY
11	Clean Up	PROJ1:MID.CLEAN
12	**Hershey**	**PROJ1:HER**
13	Grade Area	PROJ1:HER.GRADE
14	Dig Ditch	PROJ1:HER.DIG
15	Lay Pipe	PROJ1:HER.LAY
16	Clean Up	PROJ1:HER.CLEAN
17	Project Completed	PROJ1:FIN

Again, all I have to is select the WBS cell for the task and type in the appropriate code.

WBS Formulas

So now the question is what to do with this WBS code. Well, almost on an annual basis, someone asks me about formulas to pull different levels of the WBS from the WBS code. Here're a couple of formulas to get you started. Note that these formulas are predicated on using WBS Codes separated with a period and with a Project Number prefix including a colon.

For example, "PROJ1:ABC.DEF.GHI"

To extract each level, insert the following formulas into a spare Text field. Make sure to configure the fields to use the formula to calculate summary row values.

Level	Formula
1	IIf([Outline Level]=0,"",IIf([Outline Level]=1,Right([WBS],Len([WBS])-INSTR([WBS],":")),Mid([WBS],INSTR([WBS],":")+1,INSTR([WBS],".")-INSTR([WBS],":")-1)))
2	IIf([Outline Level]=2,RIGHT([WBS],LEN([WBS])-INSTR([WBS],".")),IIf([Outline Level]>2,Left(RIGHT([WBS],LEN([WBS])-INSTR([WBS],".")),INSTR(RIGHT([WBS],LEN([WBS])-INSTR([WBS],".")-1),".")),""))
3	IIf([Outline Level]>3,Mid([WBS],INSTR(INSTR([WBS],".")+INSTR(RIght([WBS],Len([WBS])-INSTR([WBS],":")),"."),[WBS],".")+1,INSTR(RIGHT([WBS],Len([WBS])-INSTR(INSTR([WBS],".")+INSTR(RIght([WBS],Len([WBS])-INSTR([WBS],":")),"."),[WBS],".")),".")-1),IIf([Outline Level]=3,Right([WBS],Len([WBS])-(INSTR(INSTR([WBS],".")+INSTR(RIght([WBS],Len([WBS])-INSTR([WBS],":")),"."),[WBS],"."))),""))

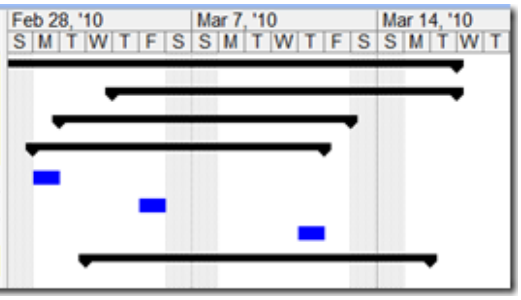
As you can see, the formulas got more and more complicated with each level. I am sure that there are more elegant methods to achieve the same results, but these seem to get the job done.

I added those formulas to my example from above and got the following results:

	Task Name	WBS	Level 1	Level 2
0	**WBS Example**	**PROJ1:**		
1	Project Started	PROJ1:START	START	
2	**Harrisburg**	**PROJ1:HAR**	**HAR**	
3	Grade Area	PROJ1:HAR.GRADE	HAR	GRADE
4	Dig Ditch	PROJ1:HAR.DIG	HAR	DIG
5	Lay Pipe	PROJ1:HAR.LAY	HAR	LAY
6	Clean Up	PROJ1:HAR.CLEAN	HAR	CLEAN
7	**Middletown**	**PROJ1:MID**	**MID**	
8	Grade Area	PROJ1:MID.GRADE	MID	GRADE
9	Dig Ditch	PROJ1:MID.DIG	MID	DIG
10	Lay Pipe	PROJ1:MID.LAY	MID	LAY
11	Clean Up	PROJ1:MID.CLEAN	MID	CLEAN
12	**Hershey**	**PROJ1:HER**	**HER**	
13	Grade Area	PROJ1:HER.GRADE	HER	GRADE
14	Dig Ditch	PROJ1:HER.DIG	HER	DIG
15	Lay Pipe	PROJ1:HER.LAY	HER	LAY
16	Clean Up	PROJ1:HER.CLEAN	HER	CLEAN
17	Project Completed	PROJ1:FIN	FIN	

Which now allows me to group on those fields using a custom Group developed in the *Project > Group By* menu. I can analyze the work, cost, or schedule of the digging activities vs. the cleaning activities – or I can see how they work out on a schedule for resource allocation review.

	Task Name	WBS	Level 1	Level 2
	Level 2: No Value			
	Level 2: CLEAN			
	Level 2: DIG			
	Level 2: GRADE			
3	Grade Area	PROJ1:HAR.GRADE	HAR	GRADE
8	Grade Area	PROJ1:MID.GRADE	MID	GRADE
13	Grade Area	PROJ1:HER.GRADE	HER	GRADE
	Level 2: LAY			



Other Benefits

So once we have these intelligent codes, what can we do out of the box?

We can trace logic using the WBS Predecessor and WBS Successor fields. If we're analyzing dependencies created to support a resource critical path, i.e. that we have one pipe laying team, and built dependencies to enforce resource allocation rules, this would be an easy way to identify that logic in the network.

We can also review the tasks at the Assignment level, as the WBS Code extends to the Assignment data for each task:

		Resource Name	WBS
		Unassigned	
		Project Started	*PROJ1:START*
		Project Completec	*PROJ1:FIN*
1		Digging Team	
		Dig Ditch	*PROJ1:HAR.DIG*
		Dig Ditch	*PROJ1:MID.DIG*
		Dig Ditch	*PROJ1:HER.DIG*
2		Grading Team	
		Grade Area	*PROJ1:HAR.GRADE*
		Grade Area	*PROJ1:MID.GRADE*
		Grade Area	*PROJ1:HER.GRADE*
3		Pipelaying Team	
		Lay Pipe	*PROJ1:HAR.LAY*
		Lay Pipe	*PROJ1:MID.LAY*
		Lay Pipe	*PROJ1:HER.LAY*
4		Clean Up Team	
		Clean Up	*PROJ1:HAR.CLEAN*
		Clean Up	*PROJ1:MID.CLEAN*
		Clean Up	*PROJ1:HER.CLEAN*

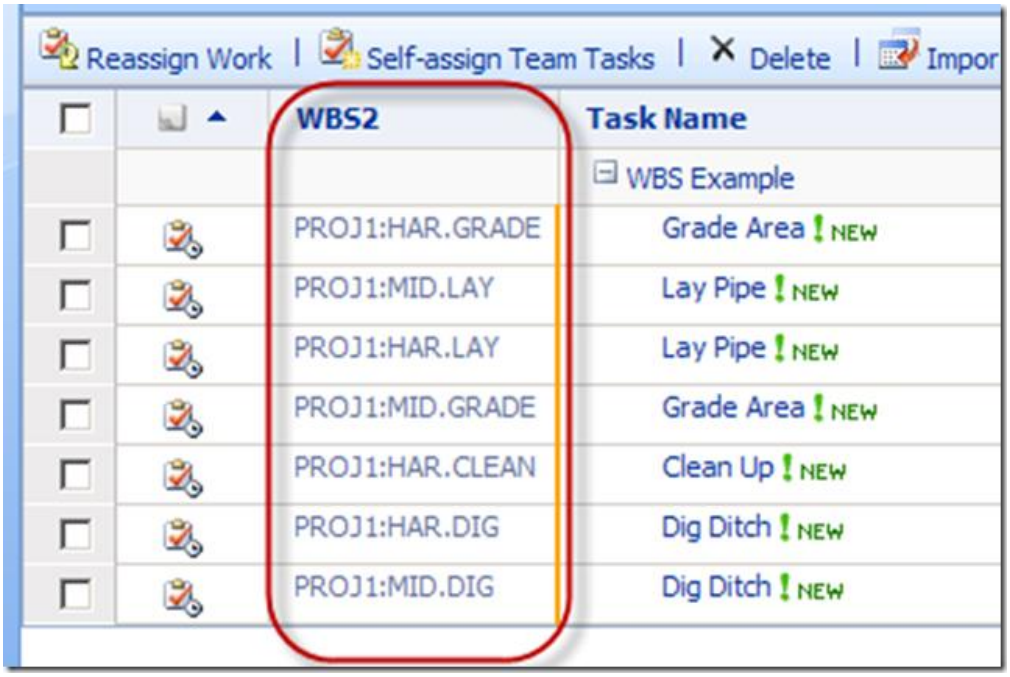
This allows us to easily differentiate tasks with similar or identical names.

Server-Side Considerations

So the question is now that we have this wonderful feature in the desktop tool, can we use Project Server to surface this information? Well, that's a bit of a different challenge.

As a colleague and I found out a couple of years ago, the WBS field can be a bit tricky to use in the Server views. The crux of the issue appears to be that WBS is published as a Task level field, and the Assignment level data sometimes is difficult to surface in some enterprise views. Thus, the WBS code is not available in the My Tasks view of Project Server.

Never fail however. There's always a workaround. The easiest method is to just create an enterprise custom field, which I called WBS2, and set the formula to "=[WBS]." Then set the field to roll down to the assignment level, add to the My Tasks views, and you're off to the races.



Similarly, those formulas I listed above work just fine as enterprise custom fields. This will allow you to create grouped and filtered enterprise views just as you would on the desktop. Following is a Project Detail view including the custom WBS fields.