OpenPlant PowerPID

Inline Component Tag from Pipeline Tag

Version 1.0

Initial Release
April 11, 2014

Trademarks

Copyrights

Table of Contents

Document Version History

Version	Date	Author	Comments
1.0	4/10/2015	Tony DeRosa	Initial release – Thanks to help from Justinas Lipnickas

Reviewed By	Date	Approved By	Date

Software Versions

Application Name	Version
Prerequisites for Bentley Desktop Applications	08.11.07.03 and Higher
OpenPlant PowerPID V8*i* SELECTseries 5	08.11.10.XX and Higher – performed in 08.11.10.316
Process & Instrumentation V8*i* SELECTseries 3R	08.11.08.151 and 08.11.09.140

Modified and New Files

Modified

File Name
OpenPlant_PID
OpenPlant_PID_Supplemental_Imperial
OpenPlant_Supplemental_Tagging

New

File Name
No new files

Tools used to perform edit

File Name
Bentley Class Editor V8*i*
Regex Designer

Inline Component Tag from Pipeline Tag

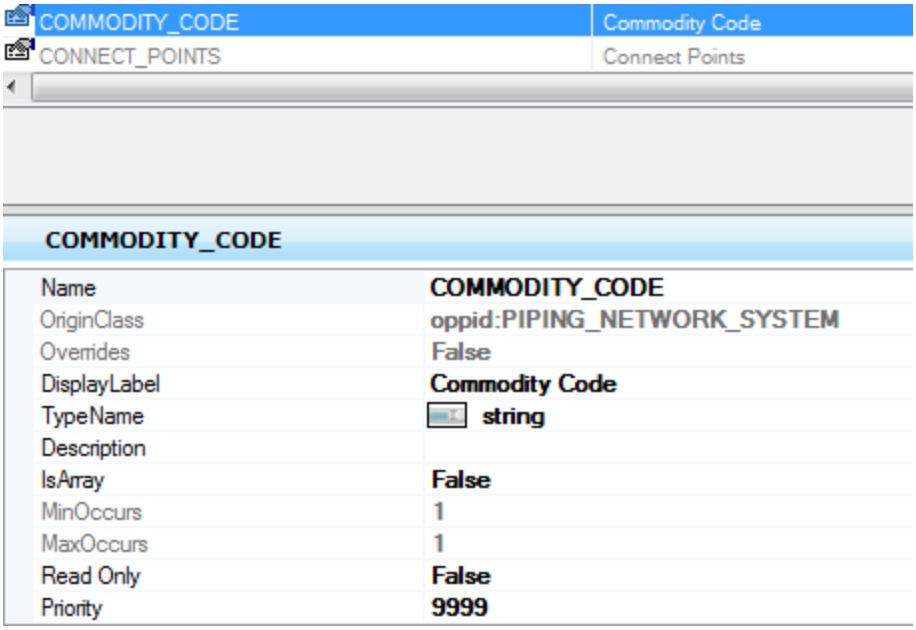
This document defines and example of creating a valve tag format that uses two properties from the pipeline in which the valve is inserted. You will add a new property to Pipeline and then pass this property to the Pipe Run along with the existing number property. From this the two properties mapped over to the run will be used in the tag format for a valve. Additionally a new tag format will be created for Pipeline and when modified will pass on to the run and ultimately onto the inline valves. Tag formats in the project database will also be addressed.
It is assumed you know how to edit schemas using the Bentley Class Editor.

Schema Changes – Part 1

Editing the OpenPlant_PID schema for Pipeline – Adding New Property

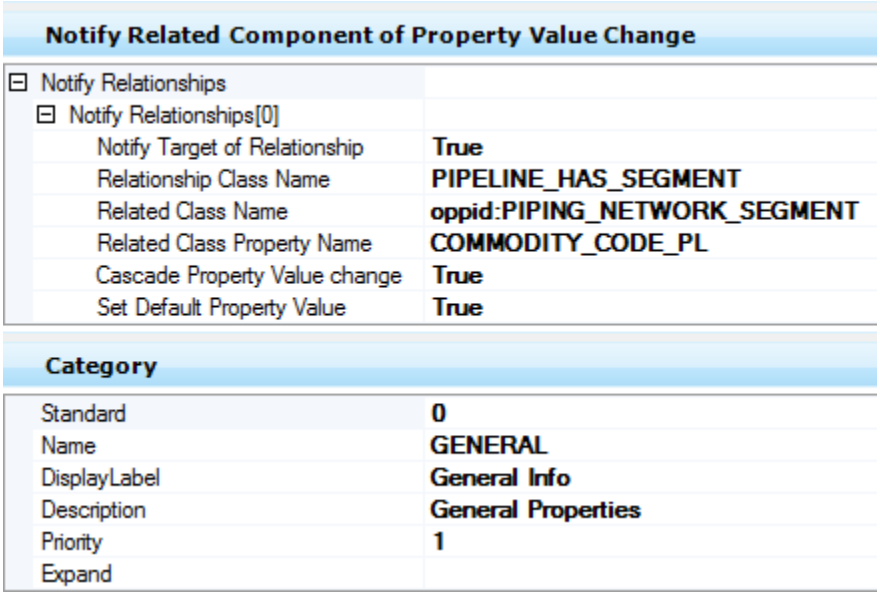
You will add the new property that is to be used in the tag for Pipeline as well as the tag for the valve. You will also configure this property to pass to the Pipe Run so it can be used in the tag for valve.

1. Load the Bentley **Class Editor**.
2. From within the class editor load the **OpenPlant_PID** schema located in workspace\project\OPPowerPID_Metric\dataset\schemas directory.
3. Right click on the schema and from the context menu pick **Supplement Schema…**
4. Supplement the schema with **the OpenPlant_PID_Supplemental_Imperial** schema.
5. Expand the tree in the left hand pane and pick the **PIPING_NETWORK_SYSTEM** class.
6. In the right hand pain select the **Properties** tab.
7. From the Class editor right hand pane select the **Add…** button and then select **New** from the context menu.
8. Add the new property **COMMODITY_CODE** as shown below.



9. After adding the new property, pick it and then select the **Custom Attributes…** button and from the context menu select **Add/Remove…**

10. Add a **Notify Related Component of Property Value Change** and a **Category** Custom Attribute. Fill these in as shown below.



Notify Related Component of Property Value Change	
⊟ Notify Relationships	
⊟ Notify Relationships[0]	
Notify Target of Relationship	**True**
Relationship Class Name	**PIPELINE_HAS_SEGMENT**
Related Class Name	**oppid:PIPING_NETWORK_SEGMENT**
Related Class Property Name	**COMMODITY_CODE_PL**
Cascade Property Value change	**True**
Set Default Property Value	**True**

Category	
Standard	**0**
Name	**GENERAL**
DisplayLabel	**General Info**
Description	**General Properties**
Priority	**1**
Expand	

11. Details on the Notify Related Component of Property Value Change custom attribute are as defined below.

FIELD	VALUE	DESCRIPTION
Notify Target of Relationship	True	Sets the notify to “on”
Relationship Class Name	PIPELINE_HAS_SEGMENT	This defines the relationship used to transfer the property value from this class to the other class.
Related Class Name	oppid:PIPING_NETWORK_SEGMENT	Defines the target class for this property value. It is prefixed with the schema namespace of the schema where the property is defined.
Related Class Property Name	COMMODITY_CODE_PL	Name of the property on the target class.
Cascade Property Value Change	True	Changes made to the property on source will be cascaded to the property on the target.

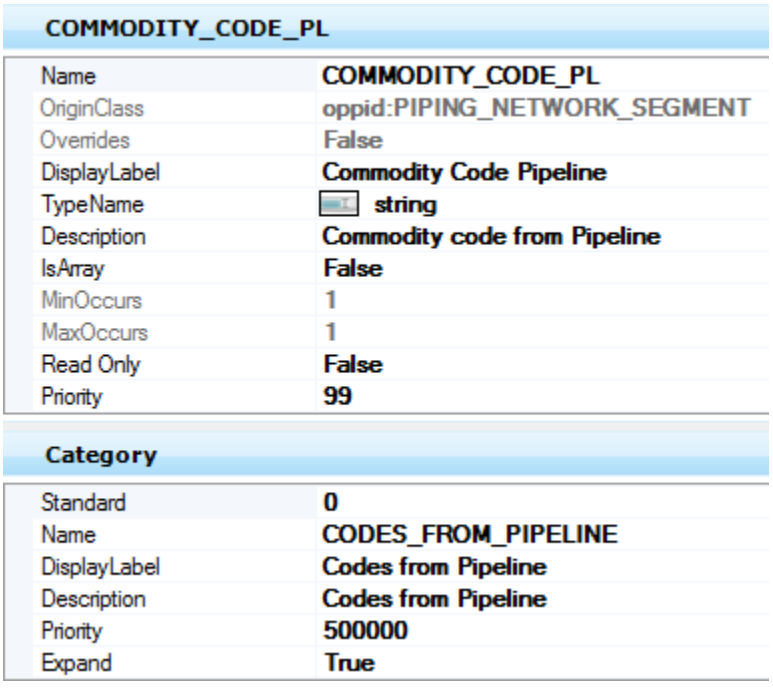
Set Default Property Value	True	When a value is entered in the property on the source class it will also set the property as the value on the property on the target class. If set to False, after placement the COMMODITY_CODE_PL property would not have a value until a change was made to the property on the source.

12. **Save** the Schema.

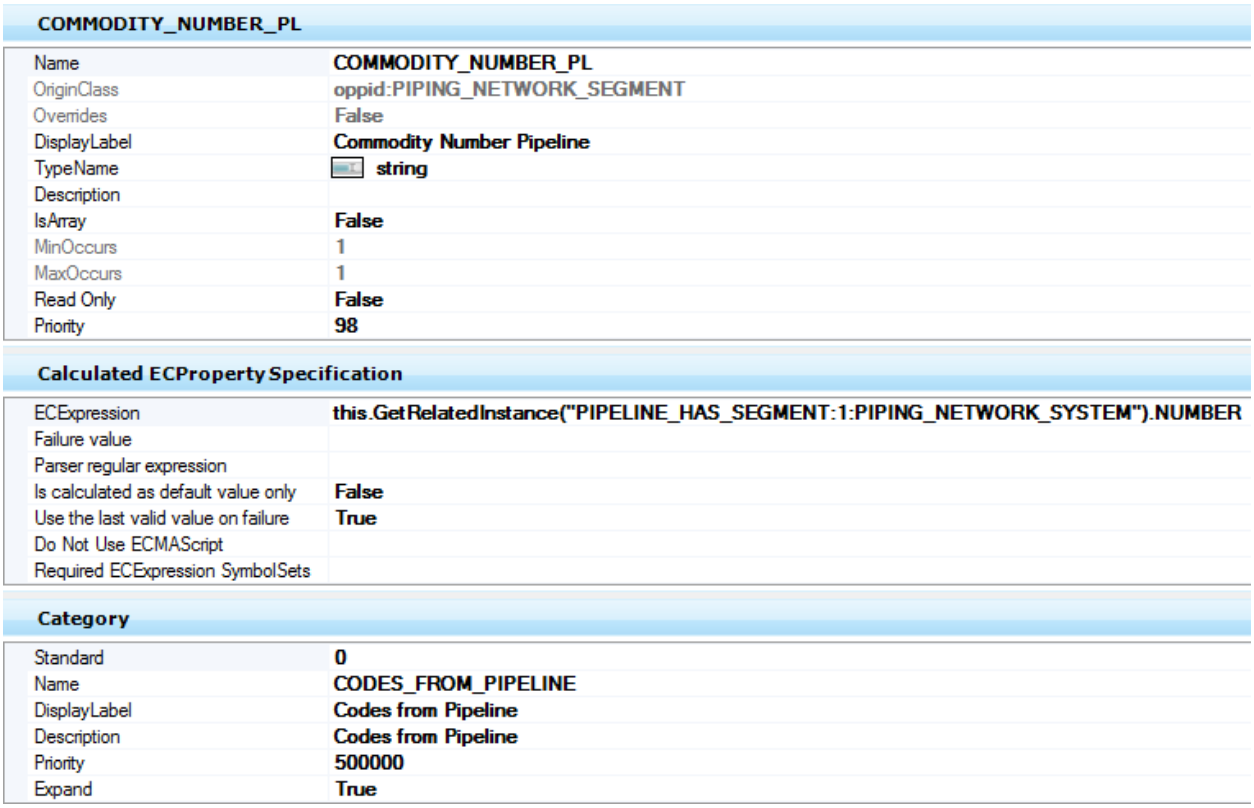
Editing the OpenPlant_PID schema for Pipe Run – Adding New Properties

Now you will modify the PIPING_NETWORK_SEGMENT class and add the new properties to hold the Commodity Code and Number values passed to Pipe Run from Pipeline.

1. Continuing in the Bentley **Class Editor** and the supplemented **OpenPlant_PID** schema, locate the **PIPING_NETWORK_SEGMENT** class.
2. Switch to the **Properties** tab.
3. Add the new **COMMODITY_CODE_PL** property. For this example we are adding this property to its own Category for simplicity sake and to verify our changes when we progress through testing. See below for details.



4. Repeat step 3 above and add the **COMMODITY_NUMBER_PL** property. For this property add the Category custom attribute and a **Calculated ECProperty Specification** custom attribute. See below for details.
   The ECExpression is used so that the run will read the number property from the pipeline without requiring the Pipeline Number to be edited first. If you use a Notify Component of Property Value Change custom attribute on the NUMBER on PIPELINE and the number value is not changed when initial placed the number will not pass to run. This is the case only when the default number on line (0001) is not changed.



COMMODITY_NUMBER_PL	
Name	COMMODITY_NUMBER_PL
OriginClass	oppid:PIPING_NETWORK_SEGMENT
Overrides	False
DisplayLabel	Commodity Number Pipeline
TypeName	string
Description	
IsArray	False
MinOccurs	1
MaxOccurs	1
Read Only	False
Priority	98

Calculated ECProperty Specification	
ECExpression	this.GetRelatedInstance("PIPELINE_HAS_SEGMENT:1:PIPING_NETWORK_SYSTEM").NUMBER
Failure value	
Parser regular expression	
Is calculated as default value only	False
Use the last valid value on failure	True
Do Not Use ECMAScript	
Required ECExpression SymbolSets	

Category	
Standard	0
Name	CODES_FROM_PIPELINE
DisplayLabel	Codes from Pipeline
Description	Codes from Pipeline
Priority	500000
Expand	True

5. **Save** the schema.

Editing the OpenPlant_PID schema for Valve – Adding New Properties

Now you will modify the VALVE class and add a new property to hold the Commodity Code and Number values passed to Pipe Run from Pipeline. You will need this value later when defining the Valve Tag Type in the database. This property does not need to be displayed as it only used for writing a calculated value to the valve database table.

1. Continuing in the Bentley **Class Editor** and the supplemented **OpenPlant_PID** schema, locate the **VALVE** class.
2. Switch to the **Properties** tab.
3. Pick the **Add…** button and from the menu select **New**.
4. Add a new property for the **Pipeline Commodity Code** and **Number** combined string. Name your property **PL_COMMCODE_AND_NUMBER** as shown below.

PL_COMMCODE_AND_NUMBER

Name	PL_COMMCODE_AND_NUMBER
OriginClass	oppid:VALVE
Overrides	False
DisplayLabel	Pipeline Commodity Code and Number
TypeName	string
Description	Combined stinrg of Commodity Code and the NUmber from Pipeline for tagging in database
IsArray	False
MinOccurs	1
MaxOccurs	1
Read Only	False
Priority	

5. Add **Category** and **Calculated ECProperty Specification** custom attributes. See below. NOTE – The Category custom attribute is for reference only and so we can verify our schema changes during the testing stage. It can be removed once this is complete. The ECExpression for the Calculated ECProperty Specification custom attribute is as follows:

Category

Standard	0
Name	COMMCODE_NUMBER
DisplayLabel	Commodity Code and Number
Description	Commoditye Code and Number
Priority	500000
Expand	True

Calculated ECProperty Specification

ECExpression	this.GetRelatedInstance("SEGMENT_HAS_PIPING_COMPONENT:1:PIPING
Failure value	None
Parser regular expression	
Is calculated as default value only	False
Use the last valid value on failure	True
Do Not Use ECMAScript	
Required ECExpression SymbolSets	

SEE THE TABLE FOR THE COMPLETE ECExpression

Field	Value	Description
ECExpression	this.GetRelatedInstance("SEGMENT_HAS_PIPING_COMPONENT:1:PIPING_NETWORK_SEGMENT").COMMODITY_CODE_PL & this.GetRelatedInstance("SEGMENT_HAS_PIPING_COMPONENT:1:PIPING_NETWORK_SEGMENT").COMMODITY_NUMBER_PL	This sets the value for the property to be the Commodity Code and sequential Number from the Pipeline. The Pipeline Commodity Code and Pipeline Number are called from Pipe Run using the SEGMENT_HAS_PIPING_COMPONENT relationship. This is possible because you

		previously passed these properties from Pipeline to Pipe Run.

6. **Save** the schema.

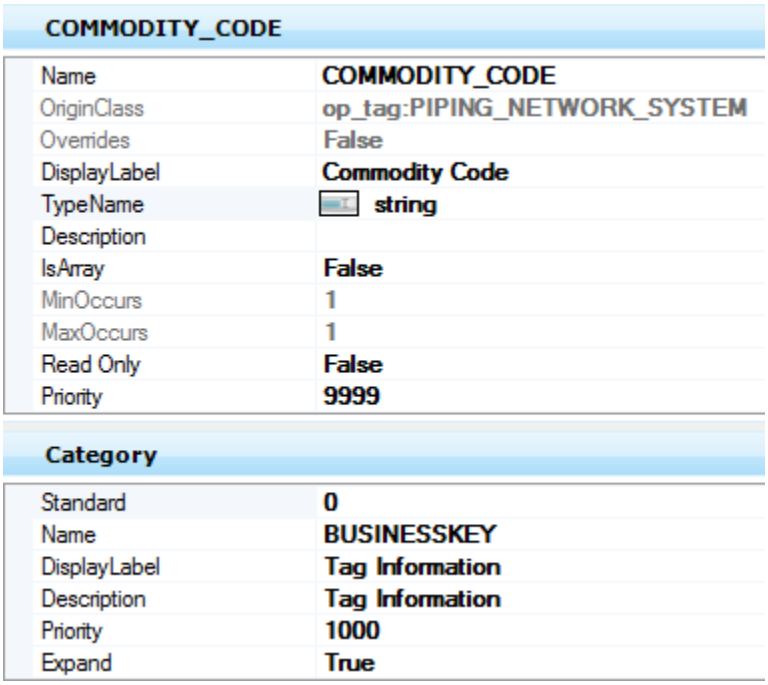
Editing the OpenPlant_Supplemental_Tagging schema – Modify Tag Format for Pipeline and Valve

Now you will load and modify the OpenPlant_Supplemental_Tagging schema and modify the tag formats for Pipeline and Valve. Pipeline will be changed to have the new Commodity Code property in the tag and Valve will have the Pipelines Commodity Code and Number included in its tag.

1. Continuing in the Bentley **Class Editor**, load the **OpenPlant_Supplemental_Mapping** schema located in workspace\project\OPPowerPID_Metric\dataset\schemas directory.
2. Locate the **Piping Network System** class in the tree. Switch to the **Properties** tab.
3. Pick on the **NAME** property. Edit the **Calculated ECProperty Specification** custom attribute as shown below. You will edit the **ECExpression** and the **Parser regular expression**.

Field	Value	Description
ECExpression	this.GetRelatedInstance("PLANT_AREA_HAS_NAMED_ITEM:1:PLANT_AREA").NAME & "-" & this.COMMODITY_CODE & this.NUMBER	This changes the name (tag) for Pipeline to be the Plant Area followed by a dash and then the Commodity Code and sequential number. An example tag is A100-CC1234. Note no dash between Commodity Code and Number.
Parser regular expression	^[?\w]*-(?<COMMODITY_CODE>[?\D]*)(?<NUMBER>[?\d]*)|(?<NUMBER>.*)	The parser expression starts by stripping out the Plant Area as this is an Associated Item. Then it parses the Commodity Code looking for non-numeric characters and then lastly assigns the balance of the numeric characters to the Number property. The last part after the bar symbol is a failure value to NUMBER if the first part of the expression does not evaluate to TRUE.

4. In order to get the **COMMODITY_CODE** property to show up in the Tag Information category, use the **Add…** button in the top panel and add the **COMMODITY_CODE** property.
5. Once this is added add a **Category** custom attribute and fill it in as shown below.



6. **Save** the Schema.
7. Continuing in the Bentley **Class Editor** and the **OpenPlant_Supplemental_Tagging** schema locate the **Valve** (FLUID_REGULATOR) class.
8. Pick on the **NAME** property. Edit the **Calculated ECProperty Specification** custom attribute as shown below. You will edit the **ECExpression** and the **Parser regular expression**.

Field	Value	Description
ECExpression	this.GetRelatedInstance("PLANT_AREA_HAS_NAMED_ITEM:1:PLANT_AREA").NAME & "-" & this.GetRelatedInstance("SEGMENT_HAS_PIPING_COMPONENT:1:PIPING_NETWORK_SEGMENT").COMMODITY_CODE_PL & this.GetRelatedInstance("SEGMENT_HAS_PIPING_COMPONENT:1:PIPING_NETWORK_SEGMENT").COMMODITY_NUMBER_PL & "-" & this.DEVICE_TYPE_CODE & this.NUMBER	This changes the name (tag) for Valve to be the Plant Area followed by a dash and then the Commodity Code and sequential number from the Pipeline, followed by a dash and then the Valve Device Type Code and Valve Number. An example tag is A100-CC1234-HV222. Note no dash between Commodity Code and Number as well as no dash between Device Type

		Code and Number. The Pipeline Commodity Code and Pipeline Number are called from Pipe Run using the SEGMENT_HAS_PIPING_COMPONENT relationship. This is possible because you previously passed these properties from Pipeline to Pipe Run.
Parser regular expression	^[?\w]*-[?\w]*-(?<DEVICE_TYPE_CODE>[?\D]*)(?<NUMBER>[?\d]*)|(?<NUMBER>.*)	The parser expression starts by stripping out the Plant Area as this is an Associated Item. Then it parses out both the Pipeline Commodity Code and Pipeline Number. Lastly it assigns all NON_NUMERIC character to Device Type Code and the balance of the numeric characters to Number. The last part after the bar symbol is a failure value to NUMBER if the first part of the expression does not evaluate to TRUE.

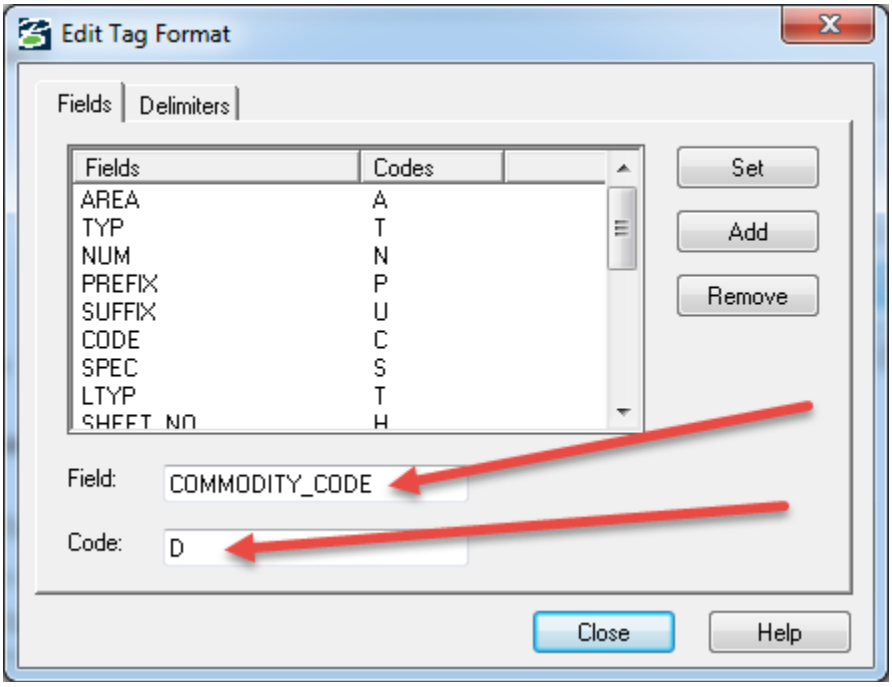
9. **Save** the schema.

Plant Project Database Changes

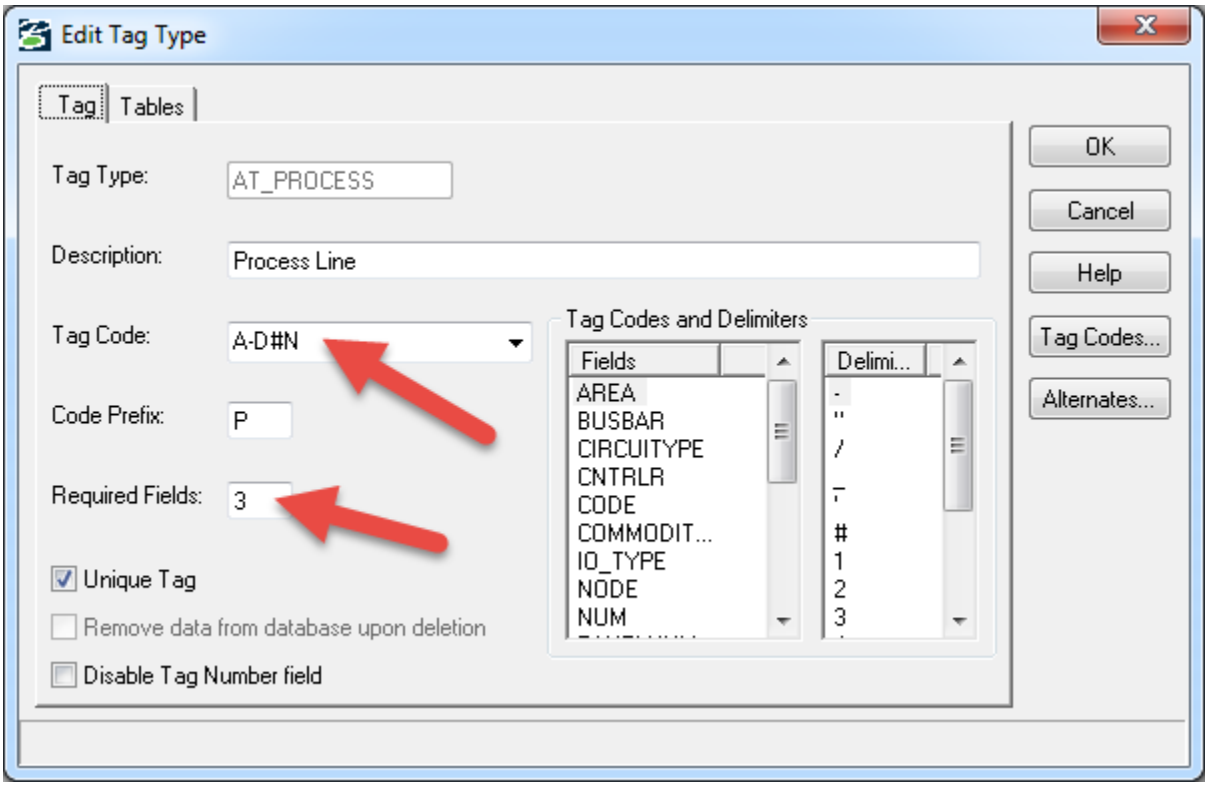
Editing Tag Format for Process Lines and Hand Valves

You will modify the Tag Format for the AT_PROCESS and AT_HVALVE tag types in the Plant Project Database. This is performed with the Project Administrator. This will require the addition of new tag codes.

1. Load the Bentley **Project Administrator**.
2. Browse to your project in the tree and to the **[PROJECT NAME]** > **Database** > **Tag Types** node.
3. Locate the Tag Type **AT_PROCESS** and pick the **Edit…** button.
4. The default tag format is “Number” you need to change this to be Area-<Commodity Code><Number>. To do this you must first create a new Tag Code for the Commodity Code field.
5. In the **Edit Tag Type** dialog pick the **Tag Codes…** button.
6. In the **Edit Tag Format** dialog in the “**Field**” section enter **COMMODITY_CODE**. In the “**Code**” section enter the first available letter – by default this is “**D**”. See below for details.



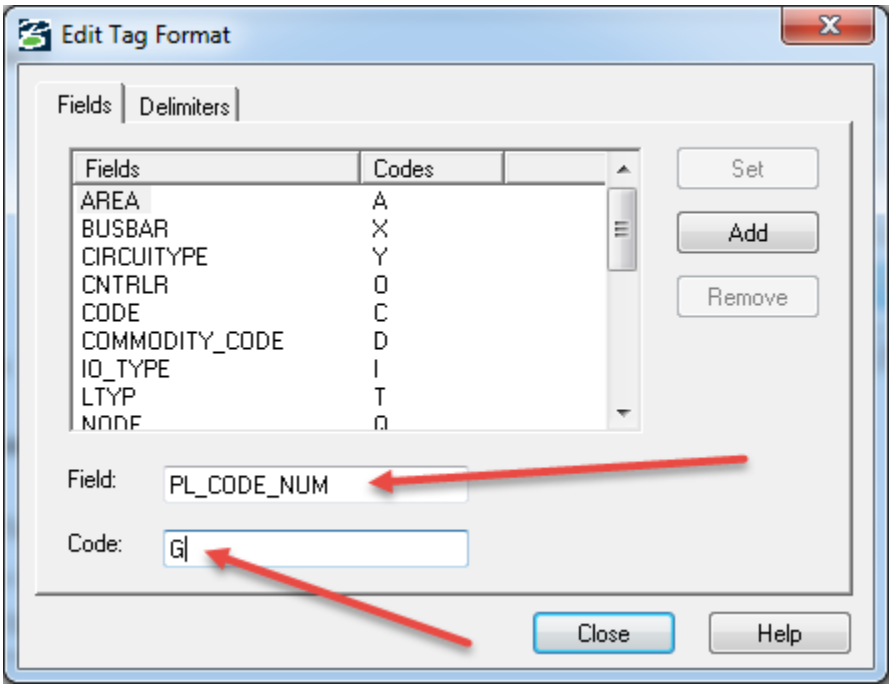
7. Pick **Add**. Pick **Close** to return to the **Edit Tag Type** dialog.
8. Now you need to adjust the **Tag Code** to match the new **Tag Format**. Using the **Tag Codes and Delimiters** enter the letters and proper delimiters to make up your **Tag Code**. Backspace over the current **N**. Select **A**, followed by the **dash**. Then pick the new **D** followed by the **#** sign delimiter and then pick an **N**.
9. Select in the **Required Fields** section and change this to **3**. See below for the finished Tag Type.



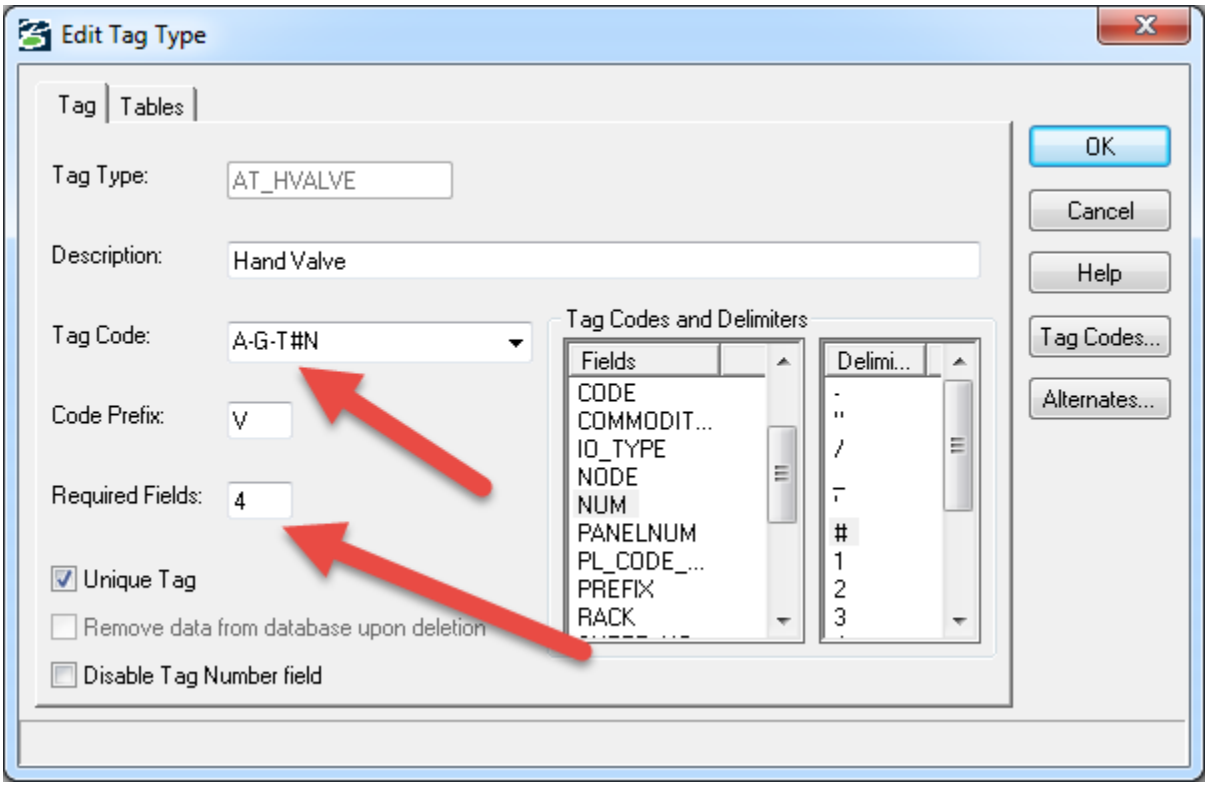
10. Pick **OK**. When prompted with a **Tag Number Fields** dialog informing you that the PCOMMODITY_CODE field was not found in the database pick the **Yes** button – you will remedy this warning by adding the fields to the database tables later.

NOTE: The # sign is a special delimiter. This is used when two fields are required in a tag and there is not a delimiter between them. Specifically in this case the Commodity Code is an Alpha character and the Number is a Numeric character. The # symbol is used and informs the system that the first part of the tag after the preceding dash is ALPHA characters and that this field continues to be populated with characters until it reaches a NUMERIC character. At that point it assigns the NUMERIC characters after the # to the next field. In this case it is to the end of tag but it could continue until another delimiter is found.

11. Continuing in the **Project Administrator** and the **Tag Types** page – locate the **AT_HVALVE** Tag Type and then pick the **Edit…** button.
12. The default tag format is "Code-Number" you need to change this to be Area-<Commodity Code & Number from pipeline>-<Device Type Code><Number>. To do this you must first create a new Tag Code for the Commodity Code/Pipeline number combined field.
13. In the **Edit Tag Type** dialog pick the **Tag Codes…** button.
14. In the **Edit Tag Format** dialog in the "**Field**" section enter **PL_CODE_NUM**. In the "**Code**" section enter the first available letter – by default this is "**G**". See below for details.



15. Pick **Add**. Pick **Close** to return to the **Edit Tag Type** dialog.
16. Now you need to adjust the **Tag Code** to match the new **Tag Format**. Using the **Tag Codes and Delimiters** enter the letters and proper delimiters to make up your **Tag Code**. Backspace over the current **C-N**. Select **A**, followed by the **dash**. Then pick the new **G** followed by the **dash** delimiter and then pick a **T** followed by the **#** sign and then pick **N**.
17. Select in the **Required Fields** section and change this to **4**. See below for the finished Tag Type.



18. Pick **OK**. When prompted with a **Tag Number Fields** dialog informing you that the VPL_CODE_NUM field was not found in the database pick the **Yes** button – you will remedy this warning by adding the fields to the database tables later.

Editing the Database and Adding New Fields

You will modify the Plant Project Database and add the new fields for Commodity Code in the PROCESS table and the combined Pipeline Commodity Code/Pipeline Number field to the Valve table. Instructions defined here are for a SQL Server format database.

1. Continuing in the **Project Administrator** locate the [**Project Name**] > **Database** > **Databases** node.
2. Select the **PROJDATA** database and then the **Edit Tables…** button.
3. In the **Tables in PROJDATA** dialog select the **PROCESS** table. Pick the **Edit…** button.
4. In the **Edit Table [PROCESS]** dialog uncheck the **Protected** property and then pick the **Table Design...** button.
5. At the bottom of the **Column Name** column enter in the name of the new commodity code field. In this case the name will be **PCOMMODITY_CODE**. NOTE **THIS MUST BE PREFIXED** with a **P**. This is a requirement. Any field that will be used in the makeup of the TAG must have the table prefix added. For the **PROCESS** table the prefix is the letter **P**.
6. Set the **Data Type** to **varchar** and the **Length** to **4**.
7. **Save** and exit Access.
8. Check the **Protected** property back on and pick **OK** to return to the **Tables in PROJDATA** dialog.
9. Select the VALVE table and then the Edit… button.
10. In the Edit Table [VALVE] dialog uncheck the Protected property and then the Table Design… button.
11. At the bottom of the **Column Name** column enter in the name of the new commodity code field. In this case the name will be **VPL_CODE_NUM**. NOTE **THIS MUST BE PREFIXED** with a **V**. This is a requirement.

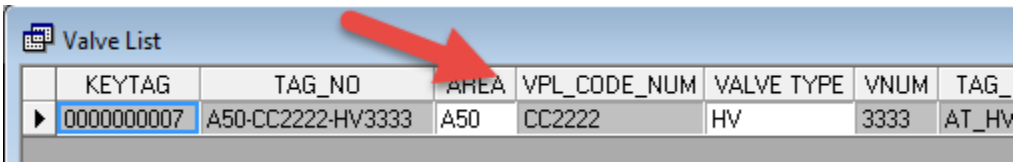
Any field that will be used in the makeup of the TAG must have the table prefix added. For the **VALVE** table the prefix is the letter **V**.

12. Set the **Data Type** to **varchar** and the **Length** to **10**.
13. **Save** and exit Access.
14. Check the **Protected** property back on and pick **OK** to return to the **Tables in PROJDATA** dialog.
15. Pick **Close** in the **Tables in PROJDATA** dialog.
16. **Exit** Project Administrator.

Editing the DataManager Views to Show the New Fields

The last part of the database configuration is to modify the Line List and Valve List views. You will modify the views to display the Commodity Code and related Commodity Code/Number fields.

1. Load Bentley **DataManager**.
2. Load the **Line List** View. Note that the **Commodity Code** filed is not in the view.
3. Right click on the **Line List** view and from the context menu select **Edit**.
4. In the **Edit View** dialog pick the **Design…** button.
5. In the **Design View – Line List** dialog select the **PROCESS** table in the **Available Fields** – **Source Tables:** column.
6. In the **Available Fields** – **Fields:** column, locate the **PCOMMODITY_CODE** field and double click it to move it to the **Selected Fields** – **Fields:** column.
7. Use the up and down buttons in the **Field Position** section to move the field to the desired location in the view. Pick **OK**.
8. **DO NOT LOAD THE VIEW YET**.
9. Locate the **Valve List** View. Right click on the view and from the context menu pick **Edit**.
10. In the **Edit View** dialog pick the **Design…** button.
11. In the Design View – Line List dialog select the **VALVE** table in the **Available Fields** – **Source Tables:** column.
12. In the **Available Fields** – **Fields:** column, locate the **VPL_CODE_NUM** field and double click it to move it to the **Selected Fields** – **Fields:** column.
13. Use the up and down buttons in the **Field Position** section to move the field to the desired location in the view.
14. This value is inherited from the pipeline so it would not likely that this value would be editable on the valve in DataManager. To make this field un-editable select the **VPL_CODE_NUM** field in the **Selected Fields** – **Fields:** column and then in the **Field Properties** section check on **Read Only**.
15. Pick **OK**.
16. The views have new fields added so you must update the link tables. To do this – from the menu select **Tools** > **Database Tools…**
17. In the **Database Tools** dialog pick on the **Check and Update Link Tables** radio button.
18. Pick **OK**.
19. When prompted that you have chosen to update link tables pick **Yes**.
20. If prompted to re-start DataManager pick **OK**.
21. Restart DataManager and check the two views to verify the new fields are added and in the Valve List view make sure the VPL_CODE_NUM field is read-only. Below you can see the read-only status on the VPL_CODE_NUM field.

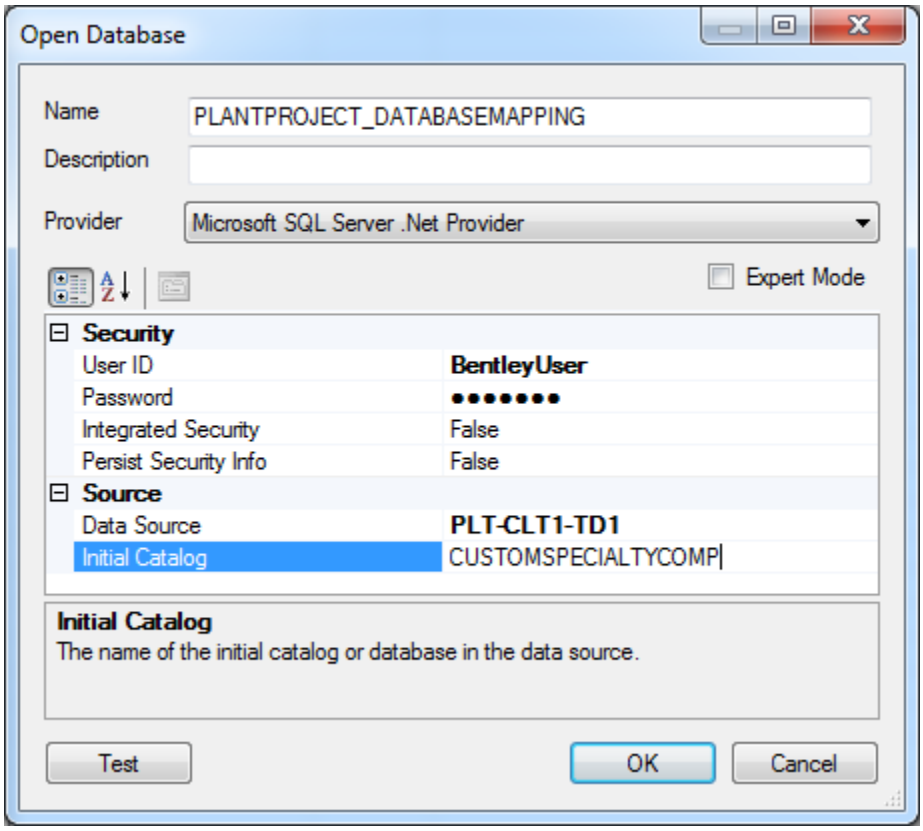


Schema Changes – Part 2

Editing the OpenPlant_PID schema and Mappings to Database

Now that the schemas and database have related properties and fields and tag types are set in the database you will map the schemas changes to the database tables.

1. Load the Bentley **Class Editor**.
2. From within the class editor load the **OpenPlant_PID** schema located in workspace\project\OPPowerPID_Metric\dataset\schemas directory.
3. From the menu select **Database Mapping** > **Open…**
4. In the **Open Mapping** dialog pick the **Browse** button.
5. In the **Custom Schema and Mapping Locations** dialog select the current path and then pick the **Remove** button. Pick the **Add** button.
6. In the **Browse For Folder** dialog browse to your project workspace schema directory. This is located in the "…workspace\Projects\<YOUR PROJECT NAME>\Dataset\schemas" directory. Pick the **directory** and not a file.
7. Pick **OK** and **OK** again in the **Custom Schema and Mapping Locations** dialog. Lastly pick the **OK** button in the **Open Mapping** dialog. If prompted for the **OpenPlant_ProjectProperties** schema browse to your projects **…dataset\schemas** directory and select the file. Pick **Open**.
8. Now that the mapping schema is loaded you must disconnect the database and the reconnect it again so it includes the new properties that you added to the **PROCESS** and **VALVE** tables.
9. In the **Summary** tab pick the **Edit Database Structure...** button.
10. In the **Database Structure Editor** dialog pick the **Remove** button.
11. Pick the **Load** button. Fill in the **Open Database** dialog as shown below. NOTE this will need to be filled in with your database type, user, server name and database instance.



12. Hit the **Kill** button in the upper right hand corner to return to the **Summary** tab.
13. Browse to and select the **PIPING_NETWORK_SYSTEM** class.
14. Switch to the **Properties** tab.
15. Locate the **COMMODITY_CODE** property.
16. In the lower right hand panel of **Class Mapping** tab, select **Edit…** button.
17. In the **Edit Property Mapping** dialog select the **PROCESS** table and the **PCOMMODITY_CODE** field in the **Columns** section.
18. Pick **OK**.
19. Browse to and select the **VALVE** class.
20. In the **Properties** tab, select the **PL_COMCODE_AND_NUMBER** property.
21. In the lower right hand panel of the **Class Mapping** tab select the **Edit…** button.
22. In the **Edit Property Mapping** dialog select the **VALVE** table and the **VPL_CODE_NUM** field in the **Columns** section.
23. Pick **OK**.
24. From the menu pick **Database Mapping** > **Save**.
25. **Exit** the Class Editor.

Testing the New Tagging Scheme

1. Load **OpenPlant PowerPID**. Create a new file or load an existing file.
2. In the **Settings>Drawing** dialog on the **Plant** tab add a new area – IE. A123.
3. Draw a **Major Pipeline**. Set the **Commodity Code** property to **CC**. Set the Number to **2222**.
4. Verify the tag format is correct – AREA-<COMMODITY_CODE><NUMBER>. The line tag should be **A123-CC2222**.
5. Edit the line and change the **Commodity Code** to **DD** – check properties of the line and verify the **Commodity Code** is updated.
6. Insert a **Gate Valve** in the line. Take the default tag number.
7. Look at the properties of the Valve. The tag should contain the Pipelines Commodity Code and sequence Number. Tag should be **A123-DD2222-HV0001**.
8. Change the Pipeline Number and verify that the Valve tag updates accordingly. Change the Commodity Code on the Pipeline and verify that the Valve tag updates accordingly.
9. Execute a Synchronize to the database and verify the Sync completes successfully and that the tags appear correct in the database.
10. Continue testing by changing a property on the pipeline in DataManager - change the number. Synchronize Database to DGN and verify that the line and valve tags update.

Notes