Tech Note 727

Automating Object Configuration Tasks in the GRAccess Toolkit: Configuring Properties for Analog FieldAttribute

All Tech Notes, Tech Alerts and KBCD documents and software are provided "as is" without warranty of any kind. See the **Terms of Use** for more information.

Topic#: 002502
Created: October 2010

Introduction

The GRAccess Toolkit enables automating activities that engineers normally perform manually using the Industrial Application Server Integrated Development Environment (IDE).

This *Tech Note* describes configuring following properties for Analog Field Attribute – AF_001 of a $UserDefined template using a C# console application.

- Access Mode to **Output**
- Data Type to **Float**
- Description to **Adding description for AF_001 Field Attribute**

Application Versions

To execute the GRAccess sample application that is described in this document, you will need the following prerequisites:

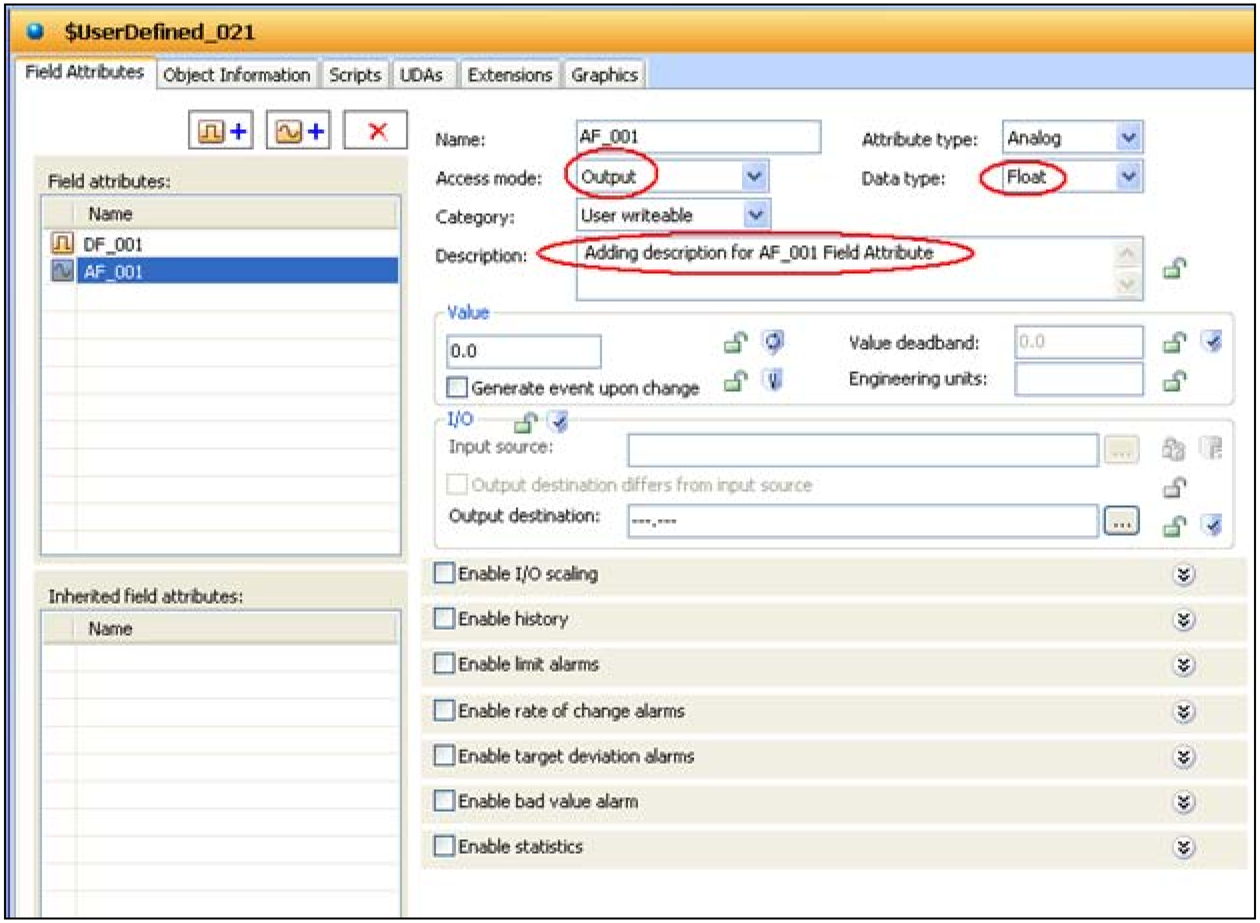
- Visual Studio 2005
- Industrial Application Server 3.0 or later
- GRAccess 3.0

Using the IDE to Configure Analog FieldAttributes Properties

Manual IDE configuration for this requirement are typically done by simply opening the **$UserDefined** template for editing, clicking on the Analog Field Attribute – AF_001 (Figure 1 below).

Then:

- Select **Output** from the Access mode list
- Select **Float** from the Data type list
- Type the specified text for your Description (Figure 1 below).



Name: AF_001

Attribute type: Analog

Access mode: Output

Data type: Float

Category: User writeable

Description: Adding description for AF_001 Field Attribute

Value: 0.0

□ Generate event upon change

Value deadband: 0.0

Engineering units:

Input source:

□ Output destination differs from input source

Output destination: ---.---

□ Enable I/O scaling

□ Enable history

□ Enable limit alarms

□ Enable rate of change alarms

□ Enable target deviation alarms

□ Enable bad value alarm

□ Enable statistics

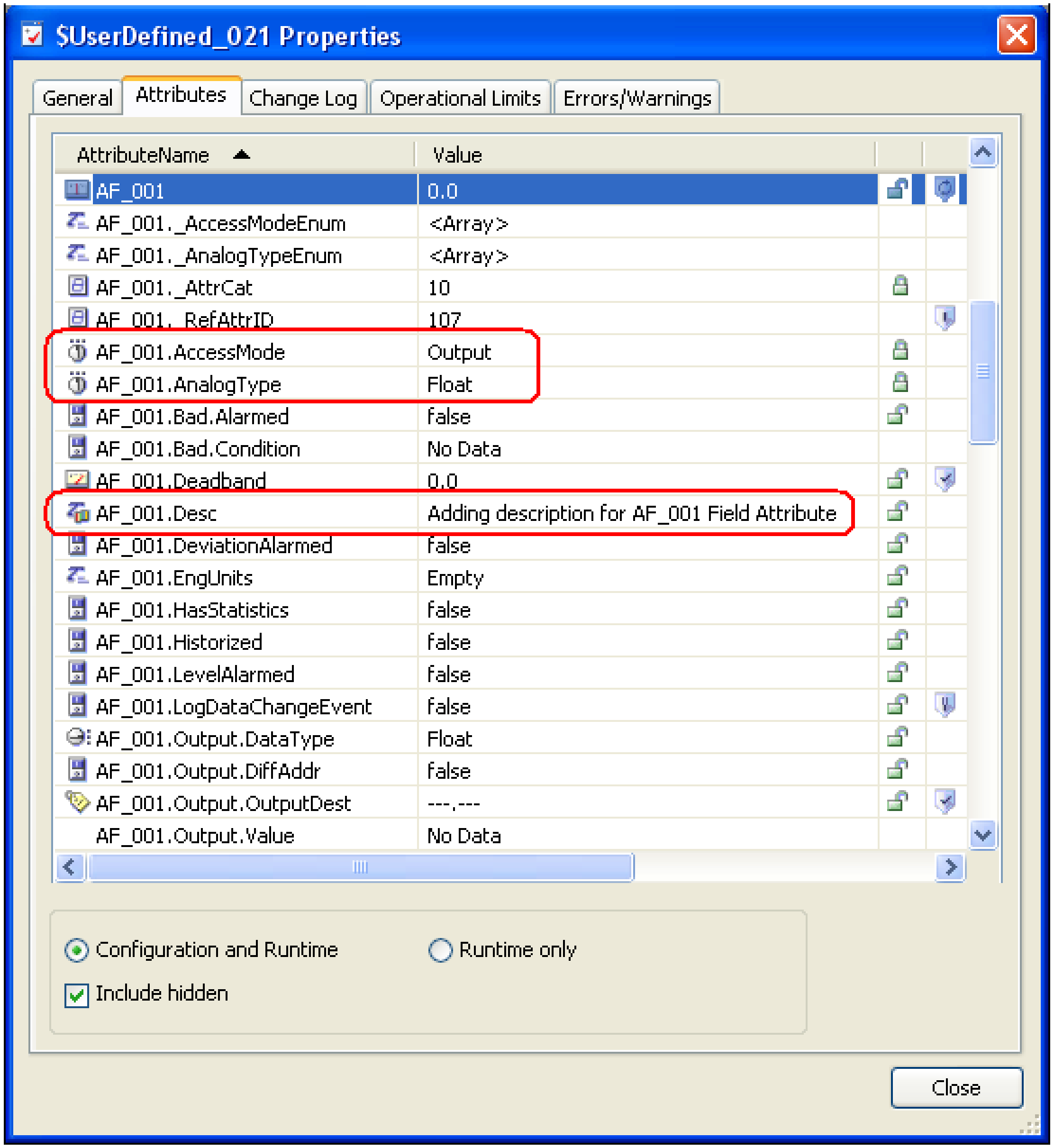
Figure 1: Analog FieldAttribute AF_001 Target Properties

To Automate this requirement using GRAccess API, you have to do some initial research. The following two tasks are done as part of the research.

Task 1

Your first task is to find out how to reference the desired property within the code – Eg: how do I reference Access mode property in the C# source code?

For this task, refer back to manual IDE configuration to look up the properties to set for Analog Field Attribute AF_001 (Figure2 below).



$UserDefined_021 Properties

AttributeName	Value
AF_001	0.0
AF_001._AccessModeEnum	<Array>
AF_001._AnalogTypeEnum	<Array>
AF_001._AttrCat	10
AF_001._RefAttrID	107
AF_001.AccessMode	Output
AF_001.AnalogType	Float
AF_001.Bad.Alarmed	false
AF_001.Bad.Condition	No Data
AF_001.Deadband	0.0
AF_001.Desc	Adding description for AF_001 Field Attribute
AF_001.DeviationAlarmed	false
AF_001.EngUnits	Empty
AF_001.HasStatistics	false
AF_001.Historized	false
AF_001.LevelAlarmed	false
AF_001.LogDataChangeEvent	false
AF_001.Output.DataType	Float
AF_001.Output.DiffAddr	false
AF_001.Output.OutputDest	---.---
AF_001.Output.Value	No Data

● Configuration and Runtime ○ Runtime only

☑ Include hidden

Figure 2: UserDefined Properties

You can conclude from the properties panel that in your C# code, you need to set the following:

- Access Mode: **AF_001.AccessMode**
- Data Type: **AF_001.AnalogType**
- Description: **AF_001.Desc**

Task 2

Your second task is to find out what are the possible values and Datatype to be used for this property. In this case the possible values for Access mode are **Input**, **InputOutput** and **Output**.

To determine the possible values

1. In Logviewer, enable the log flag **SetAttribute** for the **WWFsObject** component.
2. Manually configure the **Access Mode**, **Data Type** and **Description** properties in the ArchestrA IDE.
3. Then go back to Logviewer to examine the log messages.

   This process is described in detail in *Tech Note* **728 Using GRAccess Toolkit to Determine Values and Datatype for a Property**.

After examining the **Log Messages**, the **Values** and **Data Types** for the AccessMode, AnalogType and Desc properties you find the following:

Property	Value	Data Type
AF_001.AccessMode	3 (for Output)	MxInteger
AF_001.AnalogType	2 (for Float)	MxInteger
AF_001.Desc	"Adding description for AF_001 Field Attribute"	MxInternationalizedString

Implementing Code in C#

After doing your initial research, you can implement this requirement in C# as follows:

Declarations:

```
ICommandResult CR;
ITemplate UDATest;
IAttributes UDATestAttributes;
MxValue MXVal= new MxValueClass();
IAttribute AF_001;

UDATest.CheckOut()

// Now Configure attributes of Analog Field Attribute
// first configure AF_001.AccessMode - Input
// NOTE AF_001._AccessModeEnum - Input,InputOutput,Output
// Value 1 = Input
// Value 2 = InputOutput
// Value 3 = Output

UDATestAttributes = UDATest.ConfigurableAttributes;
AF_001 = UDATestAttributes["AF_001.AccessMode"];

//MxValue MxVal = new MxValueClass();
MxVal.PutInteger(3); // Set AccessMode to Output

AF_001.SetValue(MxVal);

if (AF_001 == null)
{
   CR = UDATest.CommandResult;
   MessageBox.Show("Attribute AF_001.AccessMode not found: " + CR.Text + " : " + CR.CustomMessage);
   UDATest.CheckIn("");
   return;
}
```

```
// next configure AF_001.AnalogType - Float
// NOTE AF_001.AnalogType - Integer, Float, Double
// Integer = 1
// Float = 2
// Double = 3

UDATestAttributes = UDATest.ConfigurableAttributes;
AF_001 = UDATestAttributes["AF_001.AnalogType"];
//MxValue MxVal = new MxValueClass();
MxVal.PutInteger(2); // set to Float

AF_001.SetValue(MxVal);

if (AF_001 == null)
{
   CR = UDATest.CommandResult;
   MessageBox.Show("Attribute AF_001.AccessMode not found: " + CR.Text + " : " + CR.CustomMessage);
   UDATest.CheckIn("");
   return;
}

// Now Configure Description attribute of Analog Field Attribute - AF_001
//"Adding description for AF_001 Field Attribute"

UDATestAttributes = UDATest.ConfigurableAttributes;
AF_001 = UDATestAttributes["AF_001.Desc"];
MxVal.PutInternationalString(1033,"Adding description for AF_001 Field Attribute");

AF_001.SetValue(MxVal);

if (AF_001 == null)
{
   CR = UDATest.CommandResult;
   MessageBox.Show("Attribute AF_001.Desc not found: " + CR.Text + " : " + CR.CustomMessage);
   UDATest.CheckIn("");
   return;
}

UDATest.Save();

UDATest.CheckIn("");
```

References

- **Tech Note 528 Automating Object Configuration Tasks Using GRAccess: Creating and Configuring Field Attributes**
- **Tech Note 728 Using GRAccess Toolkit to Determine Values and Datatype for a Property**

P. Kulkarni

Tech Notes are published occasionally by Wonderware Technical Support. Publisher: Invensys Systems, Inc., 26561 Rancho Parkway South, Lake Forest, CA 92630. There is also technical information on our software products at **Wonderware Technical Support.**

For technical support questions, send an e-mail to **support@wonderware.com**.

Back to top