Tech Note 1000

Wonderware Application Server Security Troubleshooting Essentials Part 3: Attribute Category

Topic#: 002830
Created: January 2014

Introduction

This Essentials Guide is the 3rd in a projected series.

This *Tech Note* discusses the IDE Attribute Category and its usage. In addition, we utilize an ArchestrA Object Toolkit (AOT) sample to demonstrate the assignment of additional Attribute Categories at the ArchestrA Object (AA Object) level. Included is a table listing all Attribute Category types available in ArchestrA.

Application Versions

- Wonderware Application Server 2012 and later

Attribute Category Review

The **Attribute Category** determines the behavior of *one* attribute. The defined attribute has the same behavior for every Galaxy logged-on user.

The category of an attribute is used to denote whether:

- This attribute is accessible during the configuration or runtime.
- This attribute can provide value propagation with object templates.
- This attribute's value will be retained during a failover.

All attributes existing in the ArchestrA provided by primitives or objects, such as WinPlatform, AppEngine, etc. have corresponding Category Types predefined. For example,

Primitive	Attribute	Category
WinPlatform	PlatformInfo	Calculated
WinPlatform	Host	SystemWriteable

AppEngine	ScanStatCmd	Writeable_US
AppEngine	ExecutionRelatedObject	Writeable_C_Lockable
…	…	…

The ArchestrA attribute can use many available Category Types. See the **end of this *Tech Note*** for details.

Define the following four types of the Category when creating an UDA (in the IDE):

- **Calculated**: Permits only scripts within the same object to write to the attribute. Calculated attributes **are not saved** across restarts.
- **Calculated Retentive**: Permits only scripts within the same object to write to the attribute. Calculated Retentive attributes **are saved** across restarts.
- **Object Writable**: Permits other objects to write to this attribute in addition to being set by scripts within this object. Object Writeable attributes are saved across restarts, and they are **Writeable_S**. This category is **not user-writeable**.
- **User Writeable**: Permits other users to write to this attribute in addition to being set by scripts and objects throughout the system.

  User writeable attributes **are saved across restarts**, they are **Writeable_USC_Lockable**, and they can be locked at configuration time. This category is **user writeable**.

We will explain each Category Type in the following section.

For other Category Types, you can use Wonderware AOT to create customized primitives or objects with the desired Category Type for each attribute. We will show this procedure in a later section of this *Tech Note*.

IDE Category Types Working Scenarios

Initialization

Among the IDE Category Types, the **Calculated** and **Calculated Retentive** cannot have a predefined initial value after it is deployed. At this moment, the attribute's Quality is showing **20:Initializing**.

AttributeReference	Value	Timestamp	Quality	Status
UDOCategoryTest.UDACalculated	0	12/15/2013 8:37:54.933 PM	20:Initializing	Ok
UDOCategoryTest.UDACalculatedRetentive	0	12/15/2013 8:37:54.933 PM	20:Initializing	Ok
UDOCategoryTest.UDAObjectWrite	0	12/15/2013 8:37:54.880 PM	C0:Good	Ok
UDOCategoryTest.UDAUserWrite	5	12/15/2013 8:37:54.880 PM	C0:Good	Ok

Figure 1: Initial Runtime Values of IDE Category Types

Writeable Restriction

Due to the definition of IDE Category Types, only the logged-on user can change value of User writeable Category Type. Otherwise, we will see an error:

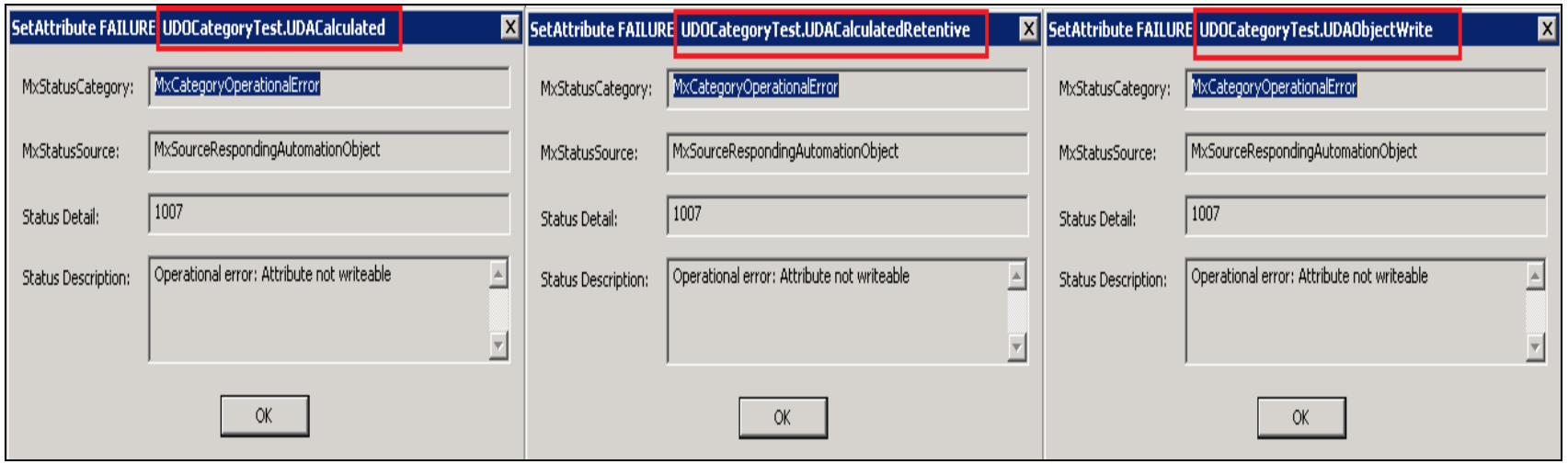


Figure 2: Logged-On User Cannot Change the Above Three Category Types' Values in Object Viewer or InTouch User Input

Writeable for Script

Script has the Writeable permission on all of the four IDE Category Types' attributes.

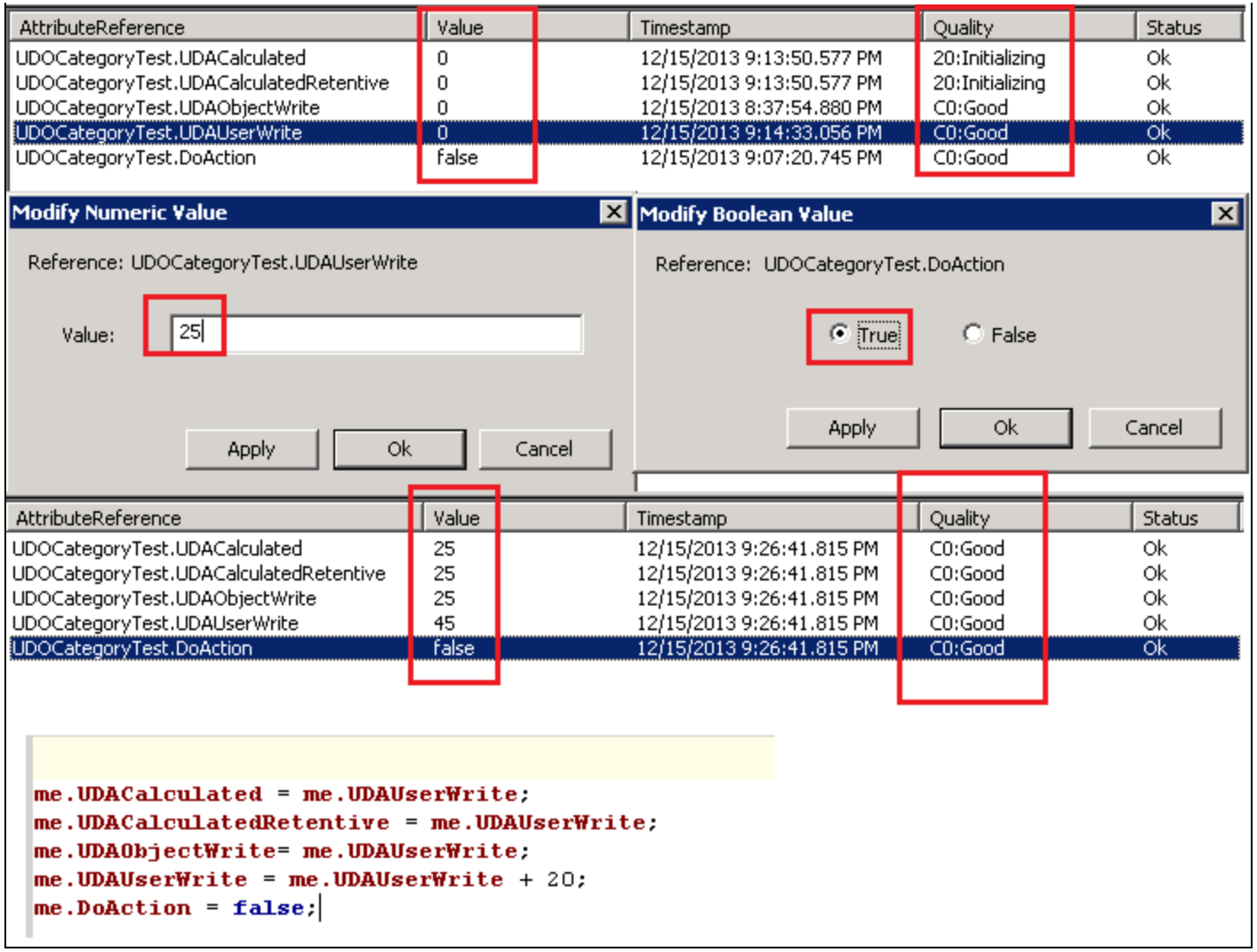


Figure 3: Script can Write Values to All IDE Category Types

Writeable for Crossing AA Objects

For the Calculated or Calculated Retentive type of attribute, Script only has the Write permission within the AA Object's boundary.

1. Set up an object called **UDOCategoryTest** (AA Object) with the following initial values:

FIGURE 4: AA TEST OBJECT INITIAL VALUES

2. Set up another object called **UDOCategoryOtherObj** (AA Object) with the following script:

```
UDOCategoryTest.UDACalculated = 200;
UDOCategoryTest.UDACalculated_Retentive = 200;
UDOCategoryTest.UDAObjectWrite= 200;
UDOCategoryTest.UDAUserWrite= 200;
me.CrossObject = false;
```

FIGURE 5: SCRIPT FOR UDOCATEGORYOTHEROBJ

3. To test, from the UDOCategoryOtherObj AA Object, set the command **me.CrossObject** to **True**.

   The values for the **Calculated** and **Calculated Retentive** type attributes do not change.

AttributeReference	Value	Timestamp	Quality	Status
UDOCategoryTest.UDACalculated	0	12/15/2013 9:50:44.915 PM	C0:Good	Ok
UDOCategoryTest.UDACalculatedRetentive	0	12/15/2013 9:50:44.915 PM	C0:Good	Ok
UDOCategoryTest.UDAObjectWrite	200	12/15/2013 10:01:20.961 PM	C0:Good	Ok
UDOCategoryTest.UDAUserWrite	200	12/15/2013 10:01:20.961 PM	C0:Good	Ok
UDOCategoryOtherObj.CrossObject	false	12/15/2013 10:01:20.961 PM	C0:Good	Ok

FIGURE 6: CALCULATED AND CALCULATED RETENTIVE ATTRIBUTE TYPES CANNOT CHANGE VALUE ACROSS THE AA OBJECT'S BOUNDARY

Attribute Value for Failover

Among the four IDE Category Types, only the **Calculated UDA** is not Checkpointed. This means the value of the UDA is *not* passed to the newly-started partner AppEngine after a Failover is finished.

AttributeReference	Value	Timestamp	Quality	Status
UDOCategoryTest.DoAction	false	12/16/2013 7:40:10.014 AM	C0:Good	Ok
RedundantTest.Redundancy.Identity	Backup	12/15/2013 7:09:06.177 PM	C0:Good	Ok
RedundantTest.Redundancy.ForceFailoverCmd	false	12/15/2013 7:09:06.177 PM	C0:Good	Ok
UDOCategoryTest.UDACalculated	0	12/16/2013 7:40:10.067 AM	20:Initializing	Ok
UDOCategoryTest.UDACalculatedRetentive	0	12/16/2013 7:40:10.067 AM	20:Initializing	Ok
UDOCategoryTest.UDAObjectWrite	0	12/16/2013 7:40:10.014 AM	C0:Good	Ok
UDOCategoryTest.UDAUserWrite	0	12/16/2013 7:40:10.014 AM	C0:Good	Ok
UDOCategoryOtherObj.CrossObject	false	12/15/2013 10:01:20.961 PM	C0:Good	Ok

Figure 7: Initial Settings for IDE Category UDAs

AttributeReference	Value	Timestamp	Quality	Status
UDOCategoryTest.DoAction	false	12/16/2013 7:43:43.323 AM	C0:Good	Ok
RedundantTest.Redundancy.Identity	Backup	12/15/2013 7:09:06.177 PM	C0:Good	Ok
RedundantTest.Redundancy.ForceFailoverCmd	false	12/15/2013 7:09:06.177 PM	C0:Good	Ok
UDOCategoryTest.UDACalculated	20	12/16/2013 7:43:43.323 AM	C0:Good	Ok
UDOCategoryTest.UDACalculatedRetentive	20	12/16/2013 7:43:43.323 AM	C0:Good	Ok
UDOCategoryTest.UDAObjectWrite	20	12/16/2013 7:43:43.323 AM	C0:Good	Ok
UDOCategoryTest.UDAUserWrite	40	12/16/2013 7:43:43.323 AM	C0:Good	Ok
UDOCategoryOtherObj.CrossObject	false	12/15/2013 10:01:20.961 PM	C0:Good	Ok

Figure 8: UDA Values Before Failover

AttributeReference	Value	Timestamp	Quality	Status
UDOCategoryTest.DoAction	false	12/16/2013 7:43:43.323 AM	C0:Good	Ok
RedundantTest.Redundancy.Identity	Primary	12/16/2013 7:46:48.187 AM	C0:Good	Ok
RedundantTest.Redundancy.ForceFailoverCmd	false	12/16/2013 7:46:48.187 AM	C0:Good	Ok
UDOCategoryTest.UDACalculated	0	12/16/2013 7:46:48.543 AM	20:Initializing	Ok
UDOCategoryTest.UDACalculatedRetentive	20	12/16/2013 7:43:43.323 AM	C0:Good	Ok
UDOCategoryTest.UDAObjectWrite	20	12/16/2013 7:43:43.323 AM	C0:Good	Ok
UDOCategoryTest.UDAUserWrite	40	12/16/2013 7:43:43.323 AM	C0:Good	Ok
UDOCategoryOtherObj.CrossObject	false	12/15/2013 10:01:20.961 PM	C0:Good	Ok

Figure 9: After Failover, Calculated Type UDA Does Not Have the Value Passed to the Newly-Started Partner AppEngine

Field Attributes

Each Field Attribute could have one Access mode selecting from **Input**, **Output** or **InputOutput** Extensions.

Unlike the UDA, within the Field Attribute, each Access Extension mode has different Category Types defined.

Access Extension	Category Types Selection
Input	NONE
Output	Calculated, Calculated Retentive, Object Writeable and User Writeable

InputOutput	Object Writeable and User Writeable

The following bullets give detailed explanations.

- **Input Access mode**: A Field Attribute (FA) with this mode means this FA cannot have its own value or quality. It only references the value or quality from other Attribute. There is **no Writeable Requirement**. Therefore, **Input Extension** does not need any **IDE Category**.

  The following screenshots show the principle of the Input Extension Access mode.

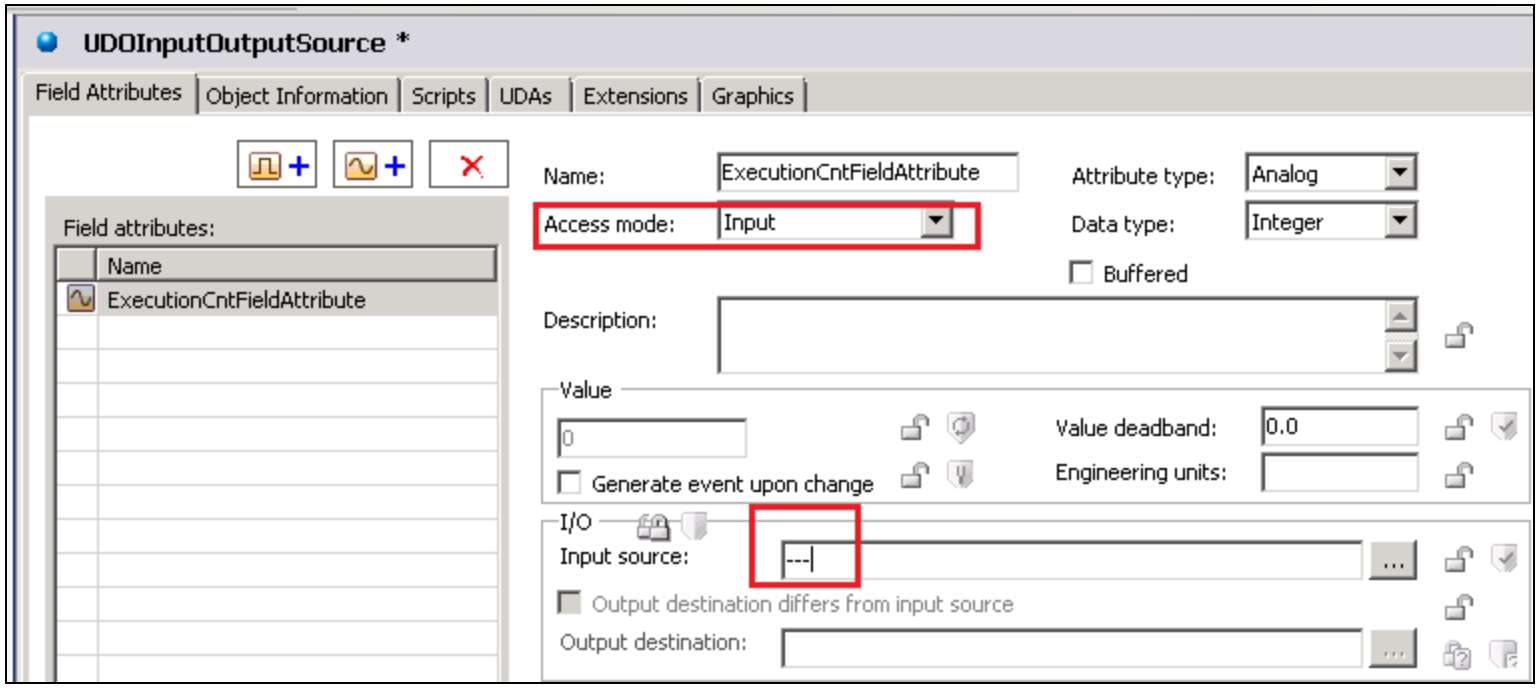


Figure 10: Define an Input Access Mode Field Attribute Without Any Input Source (Reference to Other Attribute)

- In the Object Viewer, the Input Access mode Field Attribute does not have any initial value and the Quality is bad. If setting a value to this attribute, we see the error.

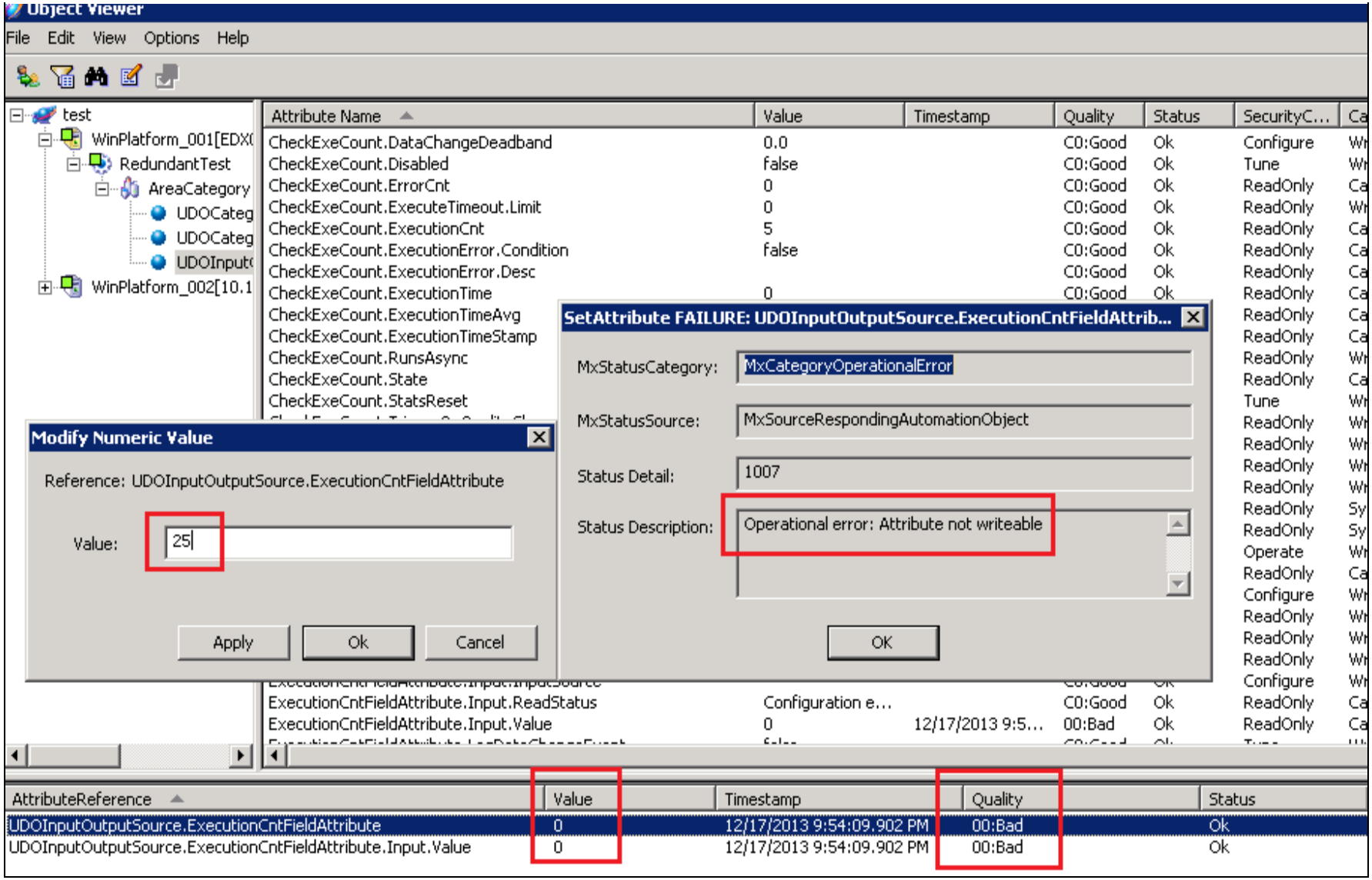


FIGURE 11: INPUT ACCESS MODE FIELD ATTRIBUTE CANNOT SET ANY VALUE BECAUSE IT IS FOR REFERENCING OTHER ATTRIBUTES

- Add a reference in the Input source Text box (Figure 12 below).

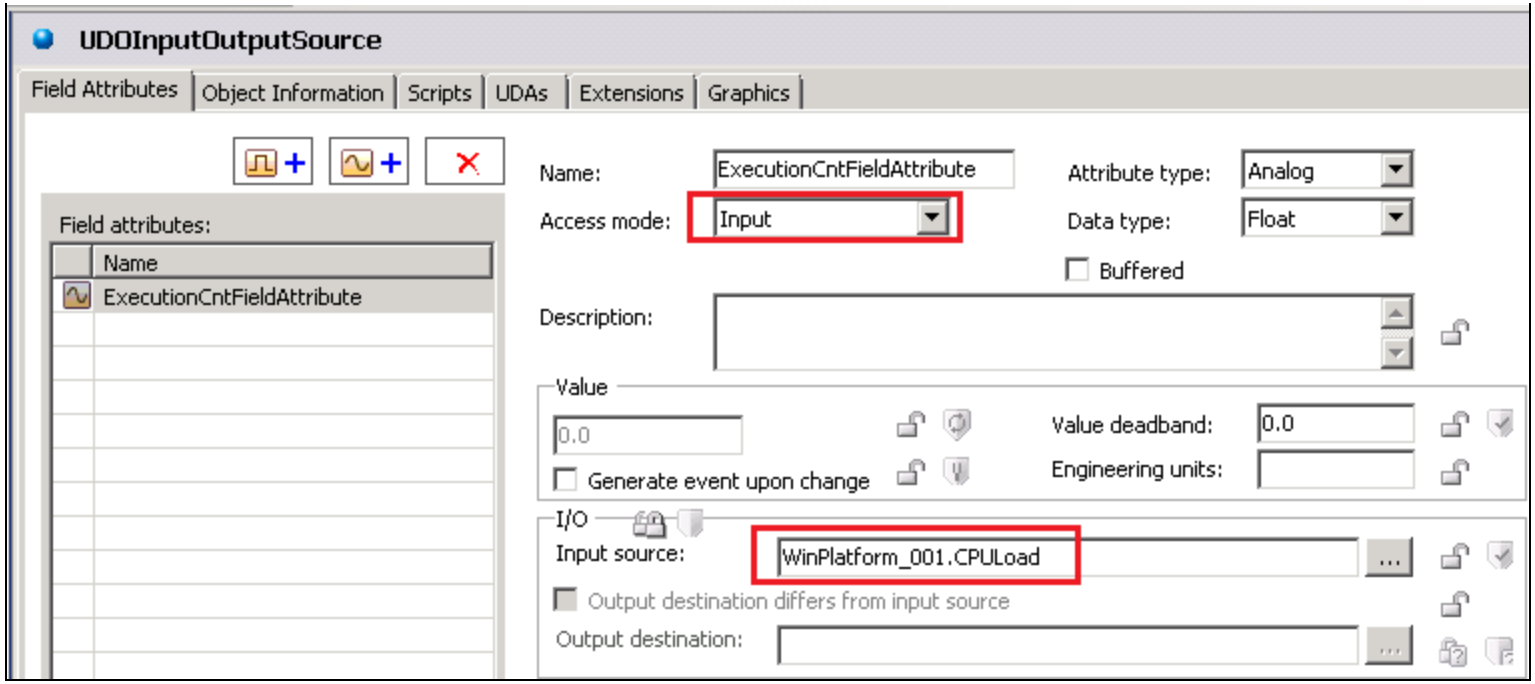


FIGURE 12: ADD A REFERENCE TO THE CPULOAD SYSTEM ATTRIBUTE IN THE INPUT SOURCE TEXT BOX

- The result is that the Field Attribute called **ExecuteCntFieldAttribute** is pointing to the **CPULoad** Attribute:

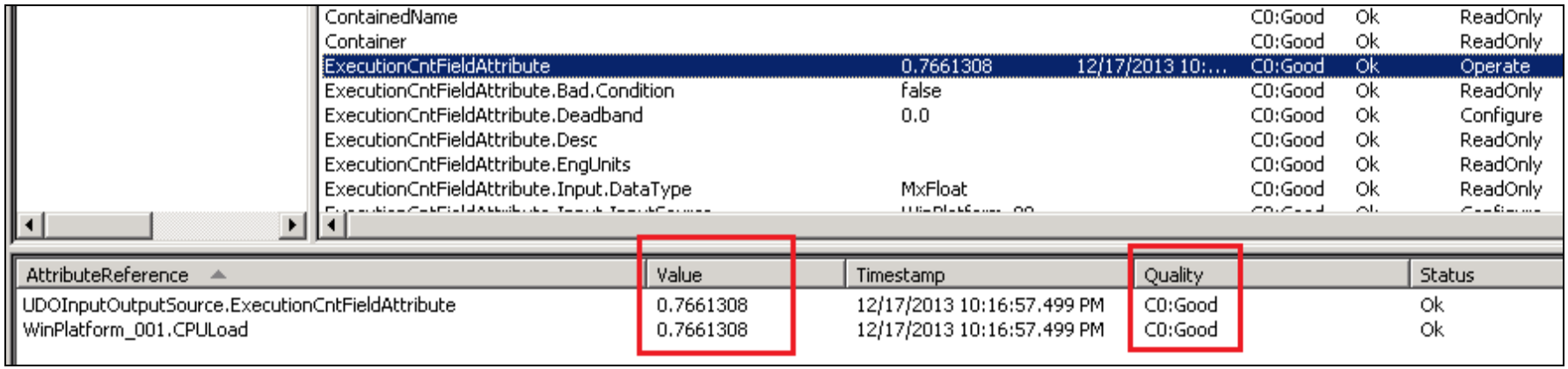


FIGURE 13: FIELD ATTRIBUTE POINTS TO CPULOAD ATTRIBUTE

Output Access Mode

Unlike the Input Extension, the Output Extension allows the Field Attribute's value to write to an **external reference destination**. In other words, when this Field Attribute's value is changed, it updates the value of the external attribute.

Because of the Writeable requirement, the Output Extension Access mode needs all four IDE's Category Types. The following screenshots

demo the principle of the Output Extension Access mode.

Setup

Item Definition	Value
Field Attribute	OutputExtensionMode
Access Mode	Output
Output Source	Me.GRPlatformCPULoad
UDA	GRPlatformCPULoad

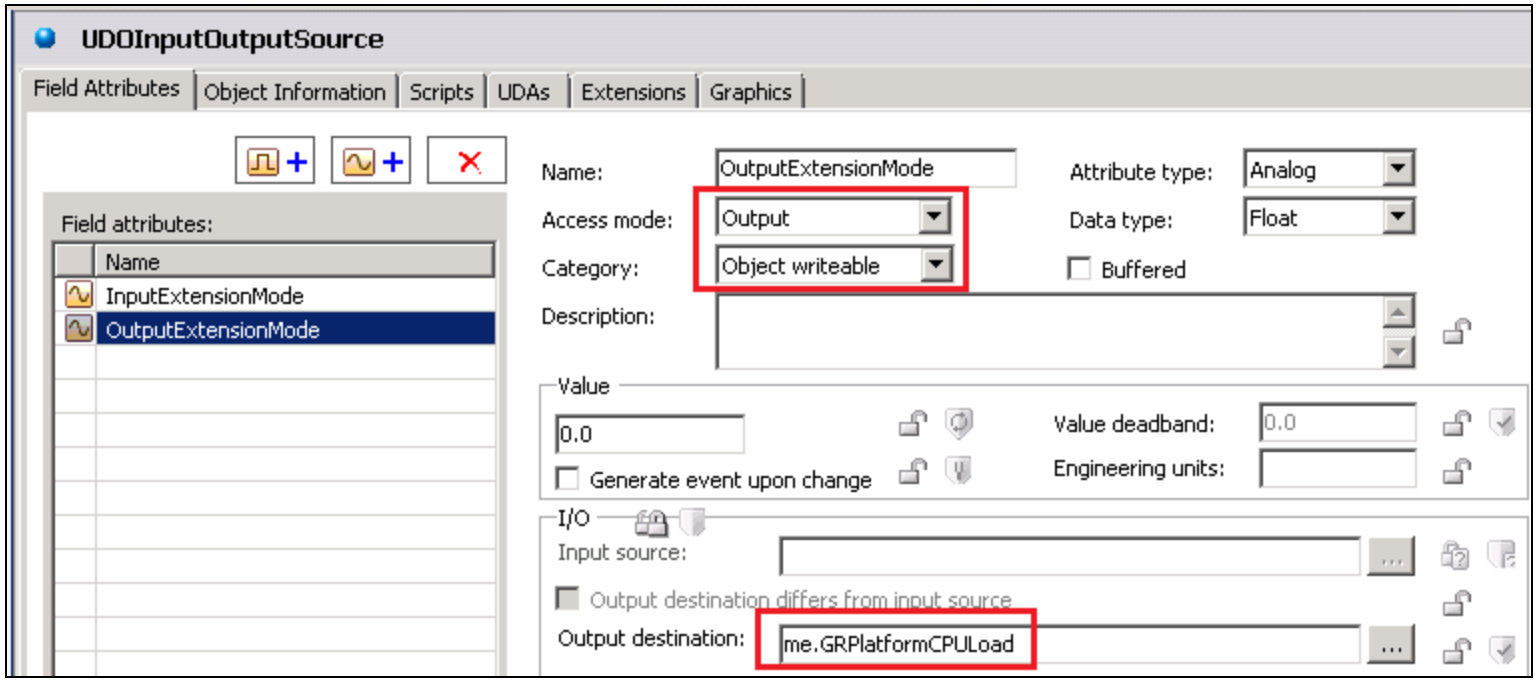


Figure 14: Value Modification on OutputExtensionMode Updates the Output Destination me.GRPlatformCPULoad

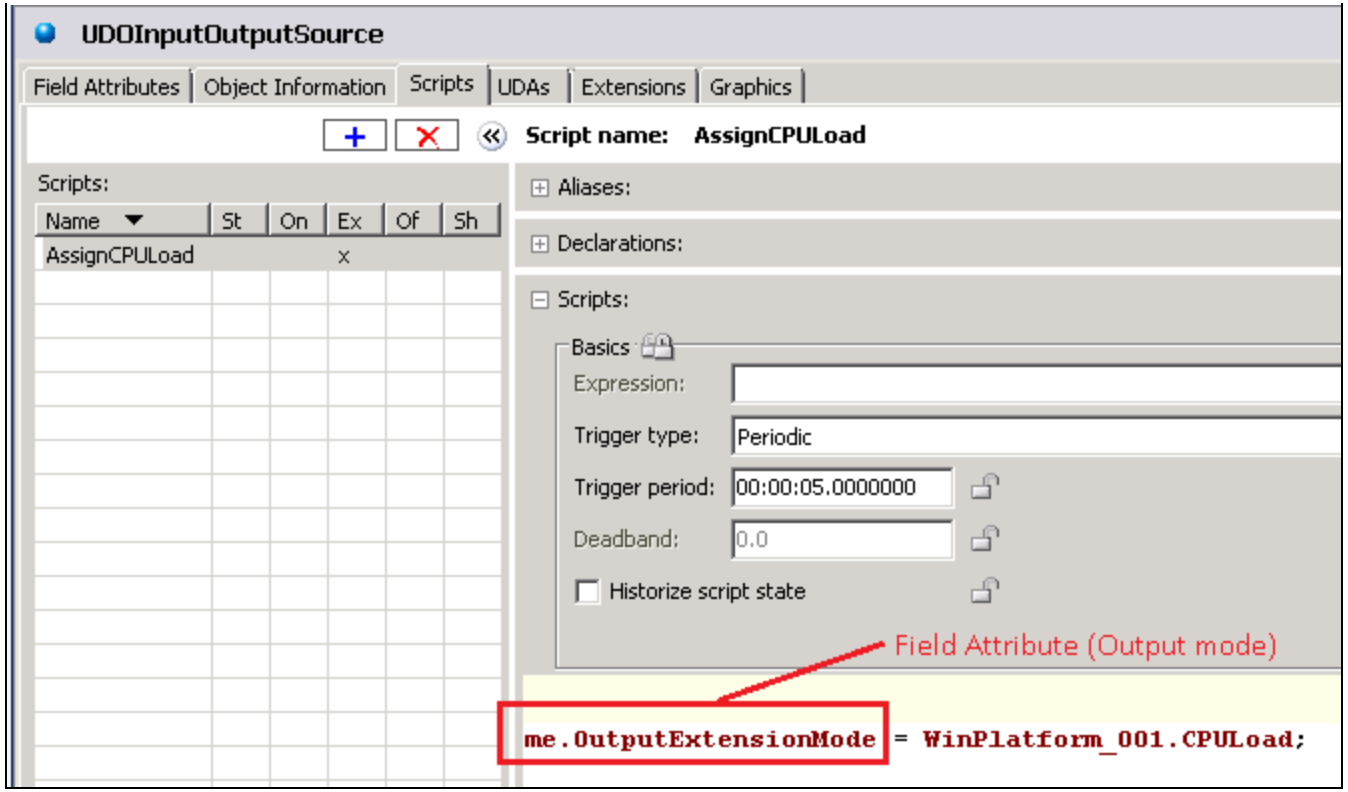


Figure 15: Modify the Value of the Field Attribute OutputExtensionMode with a Simple Script

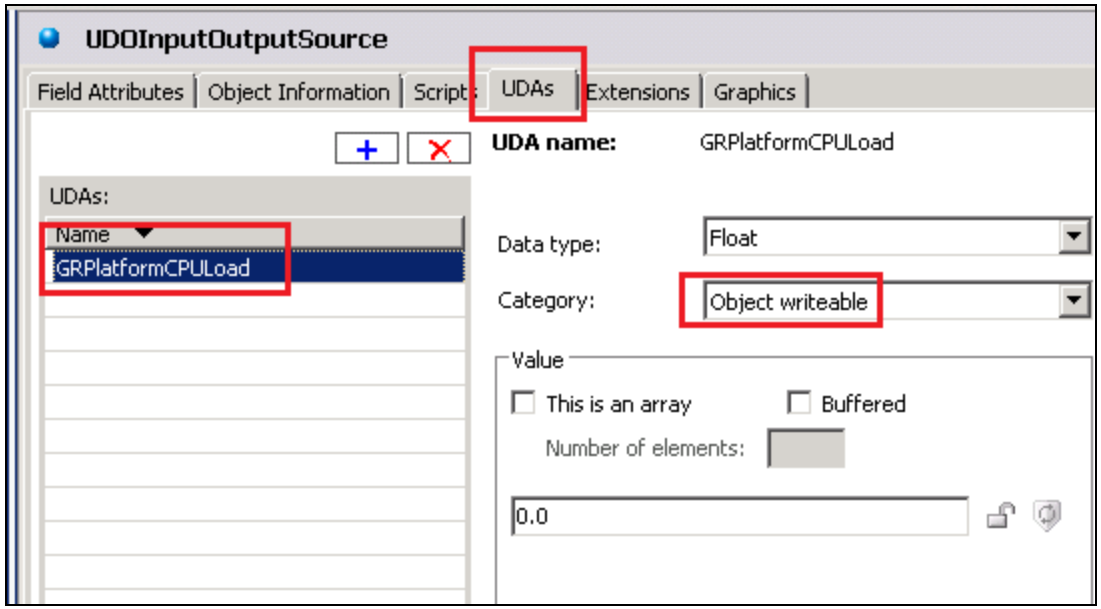
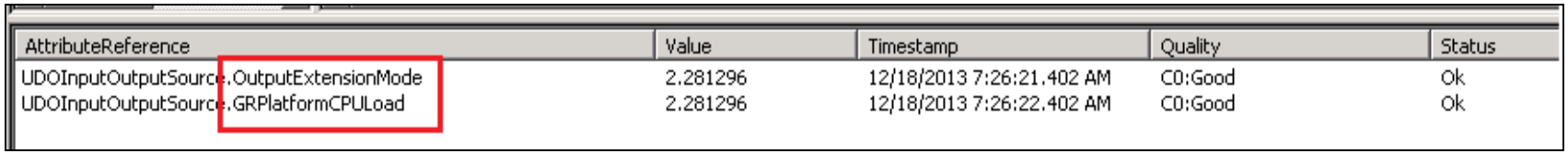


Figure 16: GRPlatformCPULoad UDA

After the deployment, you see the following result.



AttributeReference	Value	Timestamp	Quality	Status
UDOInputOutputSource.OutputExtensionMode	2.281296	12/18/2013 7:26:21.402 AM	C0:Good	Ok
UDOInputOutputSource.GRPlatformCPULoad	2.281296	12/18/2013 7:26:22.402 AM	C0:Good	Ok

FIGURE 17: RESULT OF THE OUTPUT ACCESS MODE

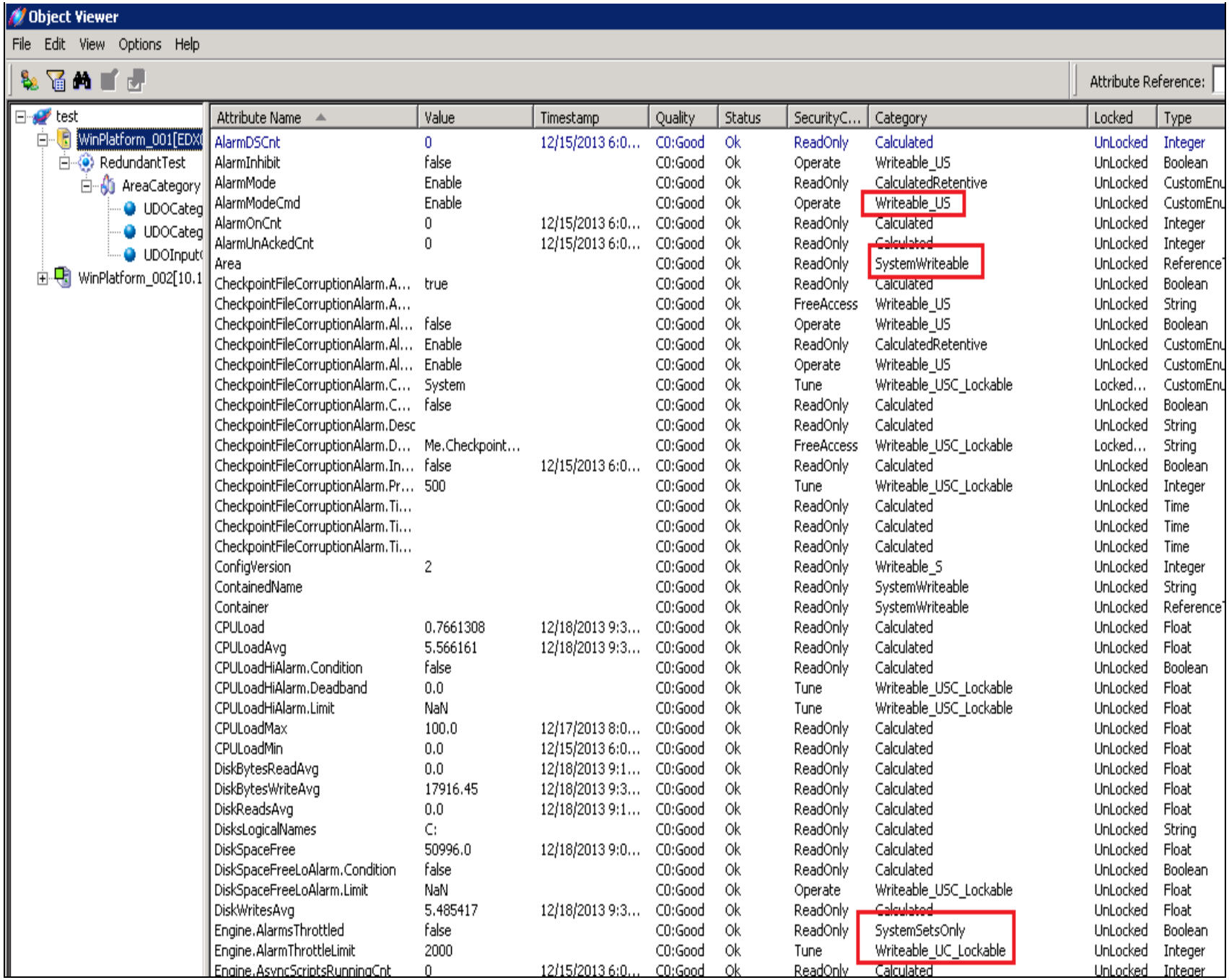
InputOutput Access Mode

The InputOutput Extension is to allow an attribute in AA Object to be configured so that its value is both read and written to an external reference destination. The primary job of the InputOutput Extension is to monitor the value/quality of an input and to send output upon change.

By design, **InputOutput Access** mode makes only the **Object writeable** and **User writeable** attributes available.

Use AOT to Create AA Object with Other Category Types

In addition to the four Category Types configurable via the IDE, more Category Types are available for certain System AA Objects such as **WinPlatform_001**. These category types are easily viewable with the Object Viewer.



Attribute Name	Value	Timestamp	Quality	Status	SecurityC...	Category	Locked	Type
AlarmDSCnt	0	12/15/2013 6:0...	C0:Good	Ok	ReadOnly	Calculated	UnLocked	Integer
AlarmInhibit	false		C0:Good	Ok	Operate	Writeable_US	UnLocked	Boolean
AlarmMode	Enable		C0:Good	Ok	ReadOnly	CalculatedRetentive	UnLocked	CustomEnu
AlarmModeCmd	Enable		C0:Good	Ok	Operate	Writeable_US	UnLocked	CustomEnu
AlarmOnCnt	0	12/15/2013 6:0...	C0:Good	Ok	ReadOnly	Calculated	UnLocked	Integer
AlarmUnAckedCnt	0	12/15/2013 6:0...	C0:Good	Ok	ReadOnly	Calculated	UnLocked	Integer
Area			C0:Good	Ok	ReadOnly	SystemWriteable	UnLocked	Reference
CheckpointFileCorruptionAlarm.A...	true		C0:Good	Ok	ReadOnly	Calculated	UnLocked	Boolean
CheckpointFileCorruptionAlarm.A...			C0:Good	Ok	FreeAccess	Writeable_US	UnLocked	String
CheckpointFileCorruptionAlarm.Al...	false		C0:Good	Ok	Operate	Writeable_US	UnLocked	Boolean
CheckpointFileCorruptionAlarm.Al...	Enable		C0:Good	Ok	ReadOnly	CalculatedRetentive	UnLocked	CustomEnu
CheckpointFileCorruptionAlarm.Al...	Enable		C0:Good	Ok	Operate	Writeable_US	UnLocked	CustomEnu
CheckpointFileCorruptionAlarm.C...	System		C0:Good	Ok	Tune	Writeable_USC_Lockable	Locked...	CustomEnu
CheckpointFileCorruptionAlarm.C...	false		C0:Good	Ok	ReadOnly	Calculated	UnLocked	Boolean
CheckpointFileCorruptionAlarm.Desc			C0:Good	Ok	ReadOnly	Calculated	UnLocked	String
CheckpointFileCorruptionAlarm.D...	Me.Checkpoint...		C0:Good	Ok	FreeAccess	Writeable_USC_Lockable	Locked...	String
CheckpointFileCorruptionAlarm.In...	false	12/15/2013 6:0...	C0:Good	Ok	ReadOnly	Calculated	UnLocked	Boolean
CheckpointFileCorruptionAlarm.Pr...	500		C0:Good	Ok	Tune	Writeable_USC_Lockable	UnLocked	Integer
CheckpointFileCorruptionAlarm.Ti...			C0:Good	Ok	ReadOnly	Calculated	UnLocked	Time
CheckpointFileCorruptionAlarm.Ti...			C0:Good	Ok	ReadOnly	Calculated	UnLocked	Time
CheckpointFileCorruptionAlarm.Ti...			C0:Good	Ok	ReadOnly	Calculated	UnLocked	Time
ConfigVersion	2		C0:Good	Ok	ReadOnly	Writeable_S	UnLocked	Integer
ContainedName			C0:Good	Ok	ReadOnly	SystemWriteable	UnLocked	String
Container			C0:Good	Ok	ReadOnly	SystemWriteable	UnLocked	Reference
CPULoad	0.7661308	12/18/2013 9:3...	C0:Good	Ok	ReadOnly	Calculated	UnLocked	Float
CPULoadAvg	5.566161	12/18/2013 9:3...	C0:Good	Ok	ReadOnly	Calculated	UnLocked	Float
CPULoadHiAlarm.Condition	false		C0:Good	Ok	ReadOnly	Calculated	UnLocked	Boolean
CPULoadHiAlarm.Deadband	0.0		C0:Good	Ok	Tune	Writeable_USC_Lockable	UnLocked	Float
CPULoadHiAlarm.Limit	NaN		C0:Good	Ok	Tune	Writeable_USC_Lockable	UnLocked	Float
CPULoadMax	100.0	12/17/2013 8:0...	C0:Good	Ok	ReadOnly	Calculated	UnLocked	Float
CPULoadMin	0.0	12/15/2013 6:0...	C0:Good	Ok	ReadOnly	Calculated	UnLocked	Float
DiskBytesReadAvg	0.0	12/18/2013 9:1...	C0:Good	Ok	ReadOnly	Calculated	UnLocked	Float
DiskBytesWriteAvg	17916.45	12/18/2013 9:3...	C0:Good	Ok	ReadOnly	Calculated	UnLocked	Float
DiskReadsAvg	0.0	12/18/2013 9:1...	C0:Good	Ok	ReadOnly	Calculated	UnLocked	Float
DisksLogicalNames	C:		C0:Good	Ok	ReadOnly	Calculated	UnLocked	String
DiskSpaceFree	50996.0	12/18/2013 9:0...	C0:Good	Ok	ReadOnly	Calculated	UnLocked	Float
DiskSpaceFreeLoAlarm.Condition	false		C0:Good	Ok	ReadOnly	Calculated	UnLocked	Boolean
DiskSpaceFreeLoAlarm.Limit	NaN		C0:Good	Ok	Operate	Writeable_USC_Lockable	UnLocked	Float
DiskWritesAvg	5.485417	12/18/2013 9:3...	C0:Good	Ok	ReadOnly	Calculated	UnLocked	Float
Engine.AlarmsThrottled	false		C0:Good	Ok	ReadOnly	SystemSetsOnly	UnLocked	Boolean
Engine.AlarmThrottleLimit	2000		C0:Good	Ok	Tune	Writeable_UC_Lockable	UnLocked	Integer
Engine.AsyncScriptsRunningCnt	0	12/15/2013 6:0...	C0:Good	Ok	ReadOnly	Calculated	UnLocked	Integer

FIGURE 18: CATEGORY TYPES IN RED ARE NOT IDES SELECTABLE CATEGORY TYPES

If your business logic calls for attributes with Category Types *not* found in the IDE, these additional attributes are configurable. In this case, Wonderware's ArchestrA Object Toolkit (AOT) is the bridge between your customized attributes and other Category Types.

In this section, we will give a brief view on the construction of this bridge.

Note: AOT has additional functionalities that allow you to build a full AA Object. That process is outside the scope of this *Tech Note*.

1. Build a single attribute AA Object using the **AOT**.

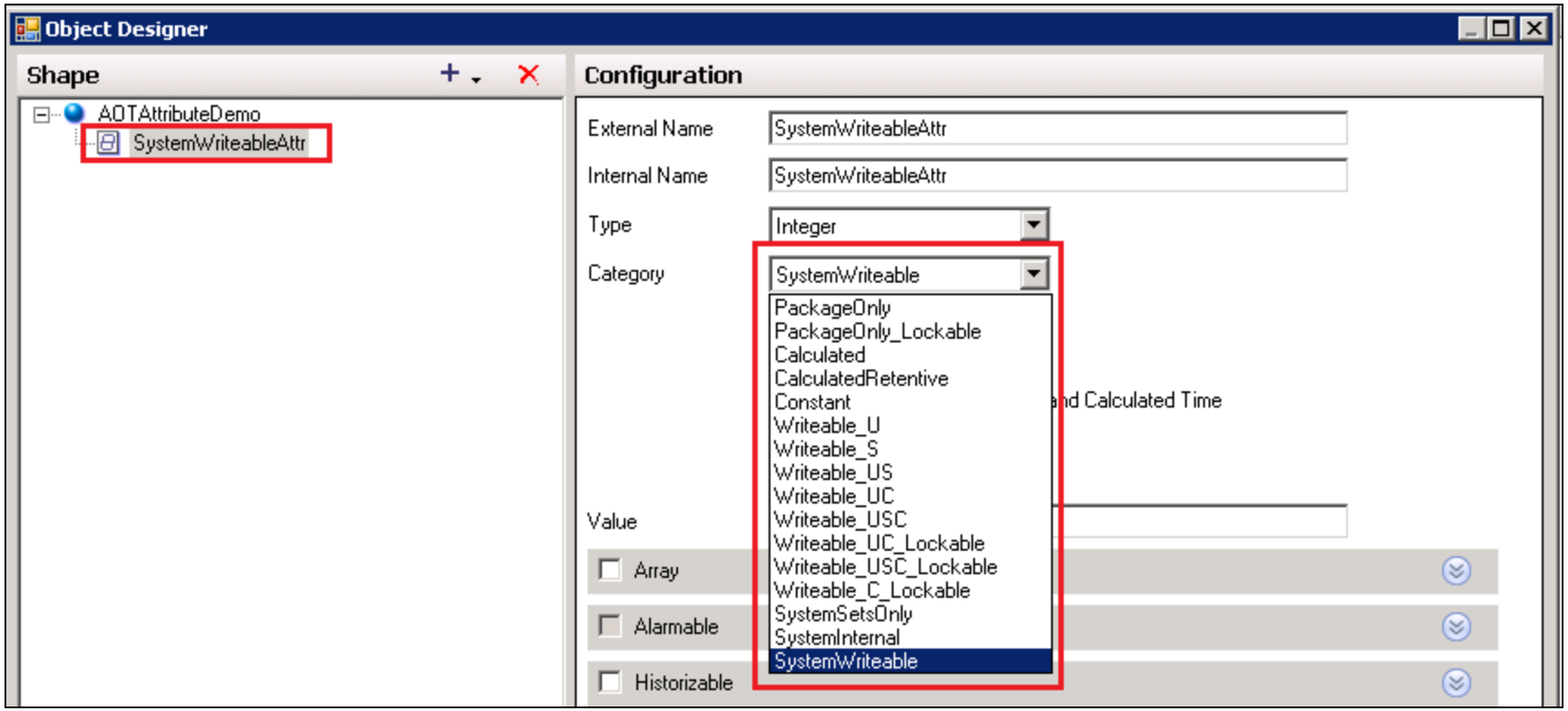


Figure 19: All Available ArchestrA Category Types

2. After making an AOT build, generate the aaPDF file. In this example, we have built **AOTAttributeDemo1.aaPDF**.
3. Import the **AOTAttributeDemo1.aaPDF** into the IDE and create an instance of the new imported template (Figure 20 below).

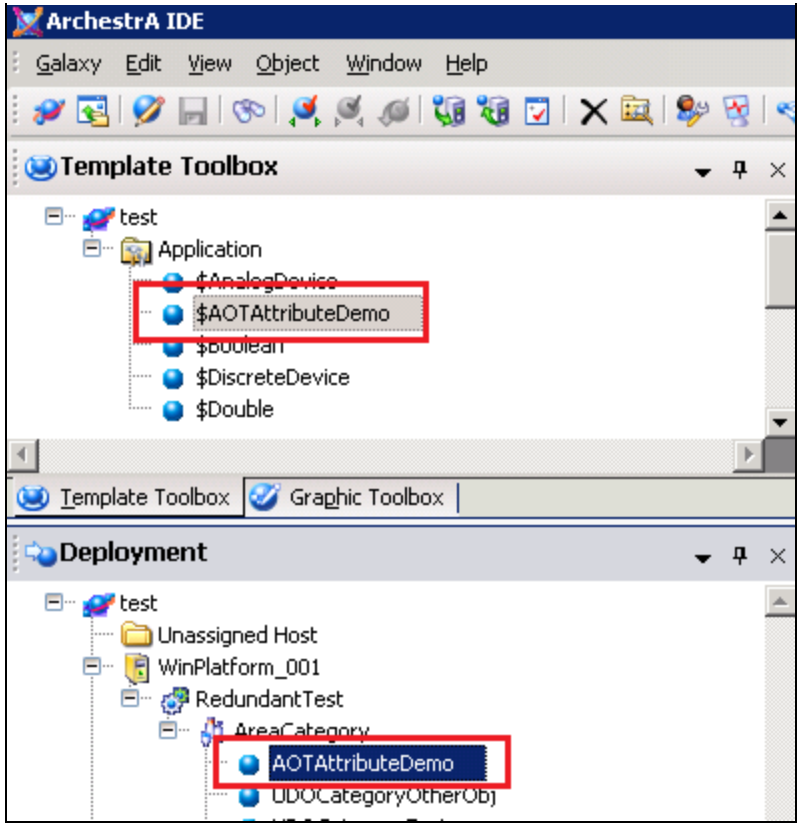


FIGURE 20: IMPORTED AOT TEMPLATE AND INSTANCE

4. View the new attribute created by AOT in Object Viewer (Figure 21 below).

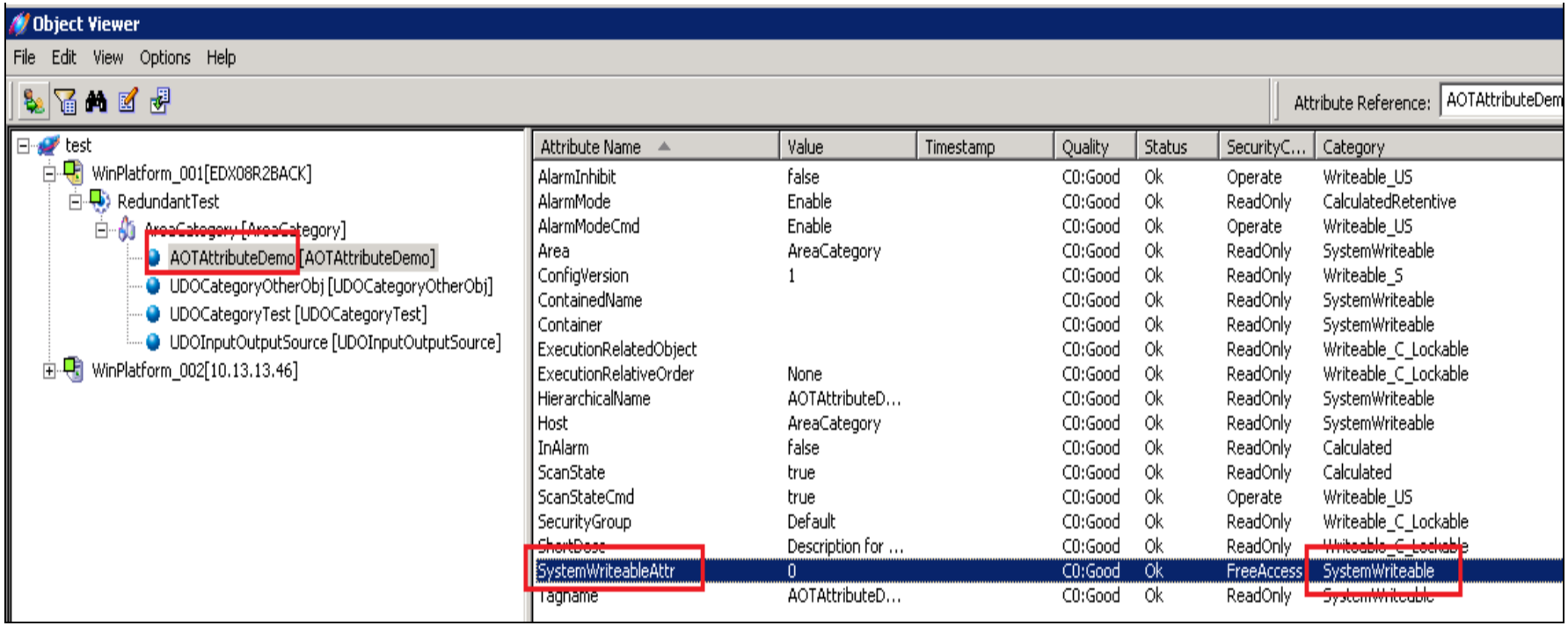


FIGURE 21: AOT USED TO CREATE NEW ATTRIBUTE SYSTEMWRITEABLEATTR, WITH THE SYSTEMWRITEABLE CATEGORY TYPE

AA Object Category Types

Category Name	Description	IDE Category Name
PackageOnly	Only exists at config time. Not deployed.	
PackageOnly_Lockable	Only exists at config time. Not deployed. Can be locked.	
Calculated	Only exists at run time. Not externally writeable by users. Run time changes are not persisted to disk by the AppEngine.	Calculated
Calculated_Retentive	Only exists at run time. Not externally writeable by users or other objects. Run time changes are persisted to disk by the AppEngine. Constant Defined by an object developer. Never changes. Exists at config time and run time.	Calculated_Retentive
Writeable_U	Exists at config time and run time, but only the Security Classification is configurable. Only externally writeable by users at run time.	
Writeable_S	Only exists at run time. Only externally writeable by other objects at run time.	Object Writable

Writeable_US	Exists at config time and run time, but only Security Classification is configurable. Externally writeable by users or other objects at run time.	
Writeable_UC	Exists at config time and run time. Only externally writeable by users at run time.	
Writeable_UC_Lockable	Exists at config time and run time. Only externally writeable by users at run time. Can be locked.	
Writeable_USC	Exists at config time and run time. Externally writeable by users or other objects at run time.	
Writeable_USC_Lockable	Exists at config time and run time. Externally writeable by users or other objects. Can be locked.	User Writeable
Writeable_C_Lockable	Exists at config time and run time. Not writeable at run time, even by the object itself. Can be locked.	
SystemSetsOnly	Designation for an Attribute Category means that the Attribute may only receive write requests and they may only originate from an object, including the same object containing the Attribute. This means that write requests from clients will not be processed.	
SystemInternal	An Attribute Category restricted to reading and writing of the value only within the object itself. The Attribute is not accessible by external objects or clients.	
SystemWriteable	Is an Attribute Category designation restricted to writing the value from within the object itself. The Attribute is not accessible by external objects or clients.	

References

- Wonderware Application Server 2012 R2 – IDE.PDF
- ArchestraObject_Toolkit_Development_Guide
- Wonderware FactorySuite A2 Deployment Guide

A. Rantos, E. Xu

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 **wwsupport@invensys.com**.

Back to top