Tech Note 840
Configuring and Using Secured\Verified Write With the SignedWrite() Function

Topic#: 002632
Created: March 2012

Introduction

The **SignedWrite()** function performs a write to an AutomationObject attribute that has a Secured Write or Verified Write security classification in the Galaxy. The SignedWrite() function can be used only in ArchestrA client scripts, not in Application Object scripts, and only on Attributes that have been configured for Secured Write or Verified Write.

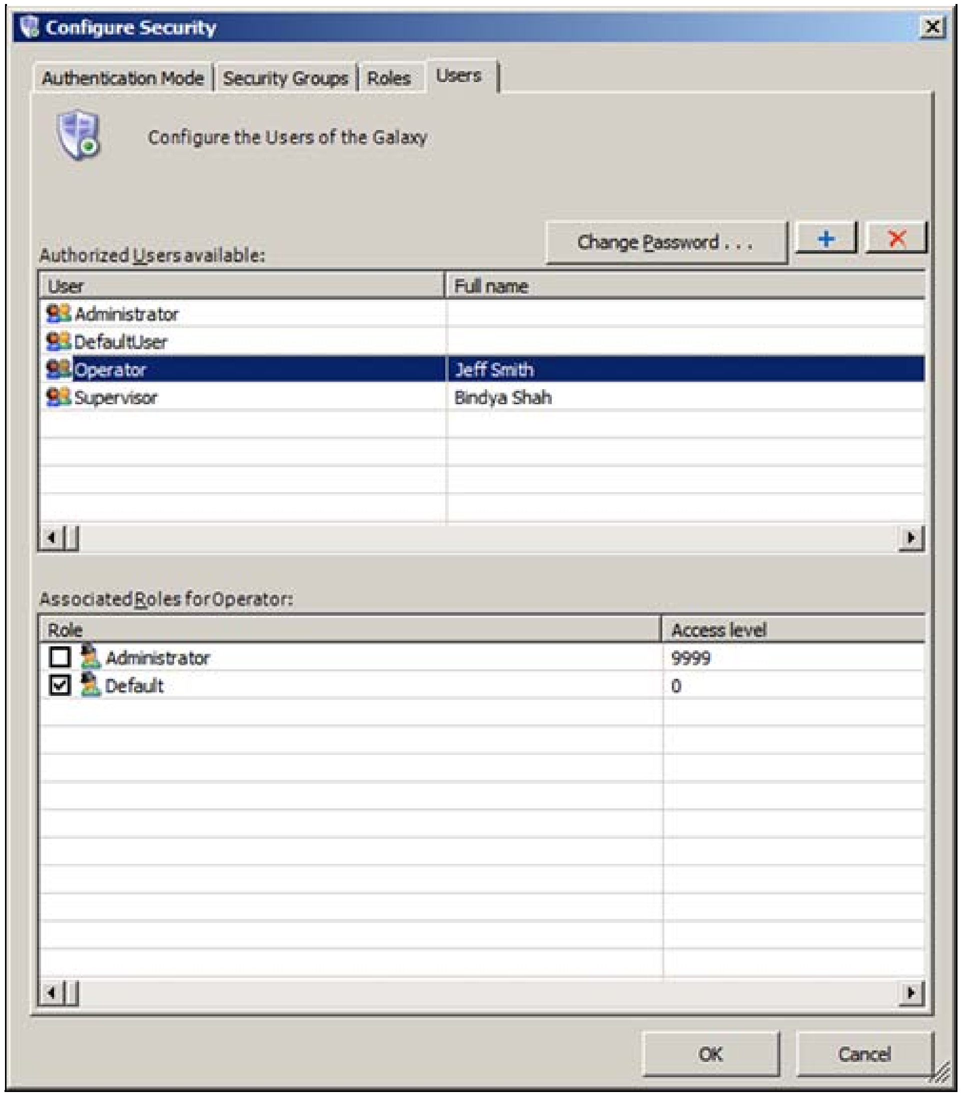
Application Versions

- Wonderware InTouch® 10.5 and later
- Wonderware Application Server 3.5 and later

For all the below examples please setup your security in the Galaxy.

To set up Galaxy Security

1. Create a new Galaxy.
2. On the main menu, click **Galaxy-> Configure-> Security**.
3. Select **Galaxy Security**.
4. Login with the following credentials:
   - Login Name: Administrator
   - Password: (blank)
5. Repeat Step 3 and click the **Users** tab.
6. Add two users. One user is an Operator (Jeff Smith) and the other is a Supervisor (Bindya Shah).



Authorized Users available:

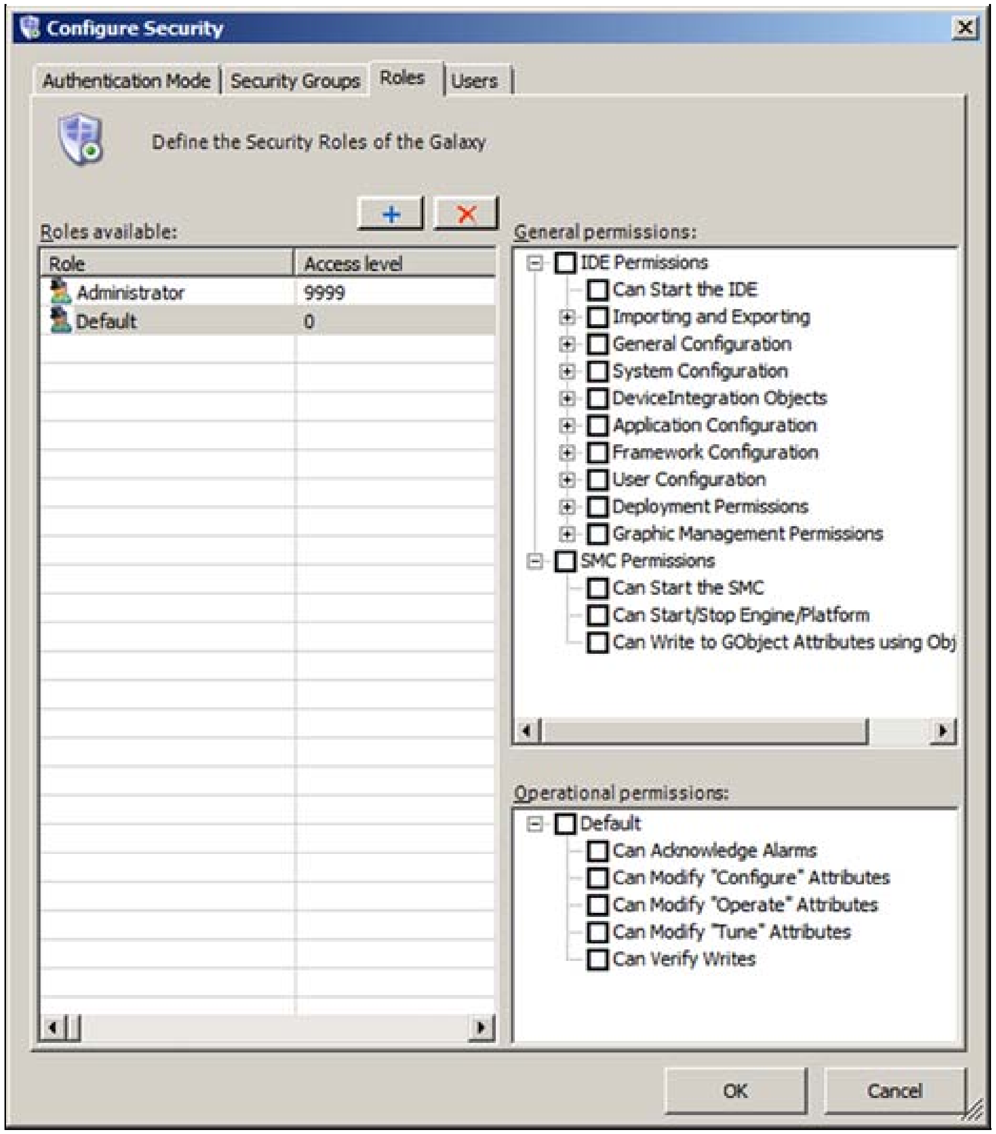
User	Full name
Administrator	
DefaultUser	
Operator	Jeff Smith
Supervisor	Bindya Shah

Associated Roles for Operator:

Role	Access level
☐ Administrator	9999
☑ Default	0

FIGURE 1: ADD OPERATOR AND SUPERVISOR USERS

7. Highlight the Operator and click **Change Password**.
8. The old password is (blank). Set a new password, for example, **operator**.
9. Highlight the Supervisor and click the **Change Password** button.
10. The old password is (blank). Set a new password, for example, **supervisor**.
11. Click the **Roles** tab.
12. Select **Default** and uncheck all options (Figure 1 below).



Role	Access level
Administrator	9999
Default	0

General permissions:

- ☐ IDE Permissions
  - ☐ Can Start the IDE
  - ☐ Importing and Exporting
  - ☐ General Configuration
  - ☐ System Configuration
  - ☐ DeviceIntegration Objects
  - ☐ Application Configuration
  - ☐ Framework Configuration
  - ☐ User Configuration
  - ☐ Deployment Permissions
  - ☐ Graphic Management Permissions
- ☐ SMC Permissions
  - ☐ Can Start the SMC
  - ☐ Can Start/Stop Engine/Platform
  - ☐ Can Write to GObject Attributes using Obj

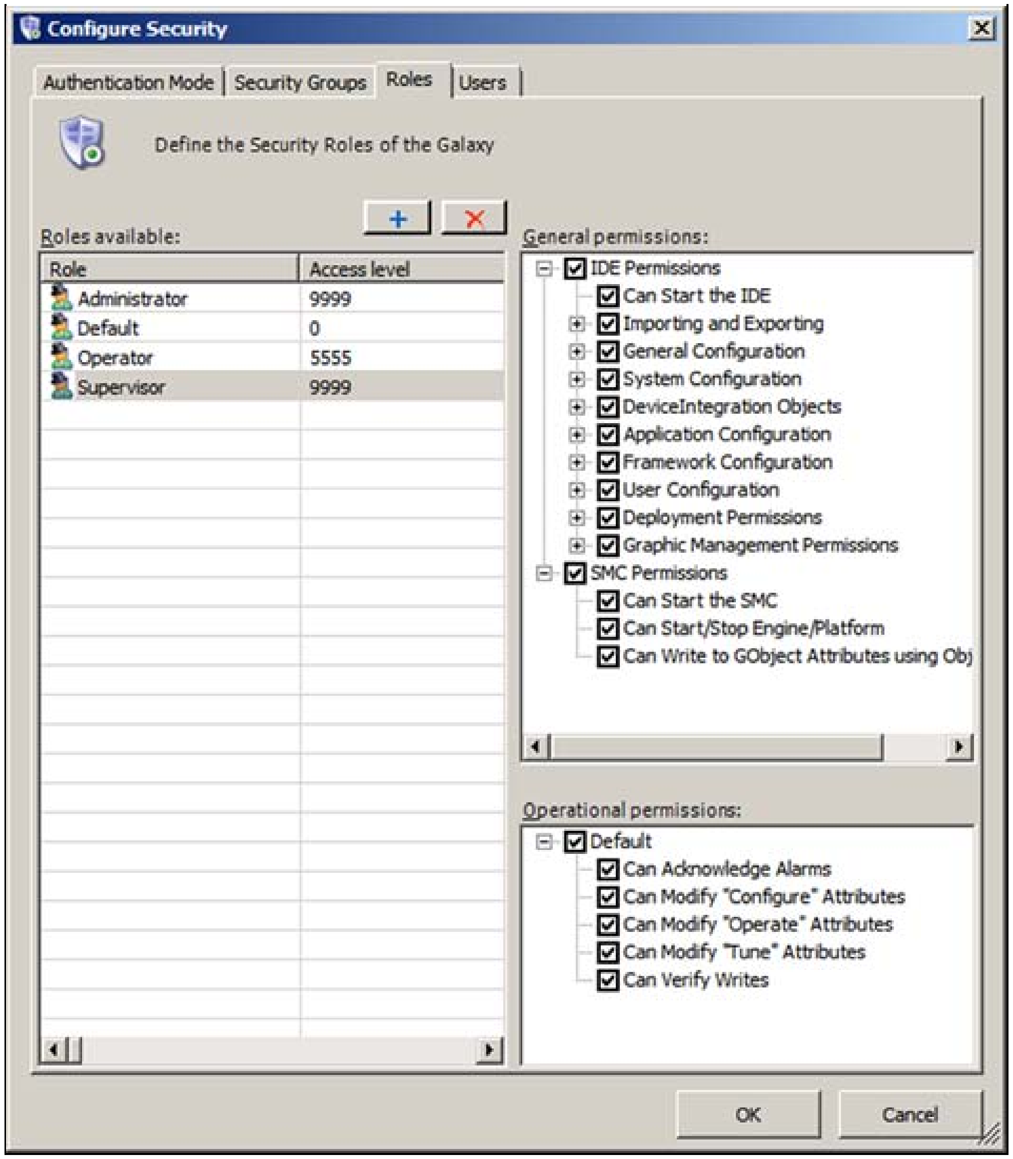
Operational permissions:

- ☐ Default
  - ☐ Can Acknowledge Alarms
  - ☐ Can Modify "Configure" Attributes
  - ☐ Can Modify "Operate" Attributes
  - ☐ Can Modify "Tune" Attributes
  - ☐ Can Verify Writes

FIGURE 2: UNCHECK ALL DEFAULT SECURITY OPTIONS

13. Add the **Supervisor** and **Operator** roles with Access Levels **9999** and **5555** respectively.

14. For the **Supervisor** role, check all the permissions shown in Figure 3 (below).



Define the Security Roles of the Galaxy

Roles available:

Role	Access level
Administrator	9999
Default	0
Operator	5555
Supervisor	9999

General permissions:

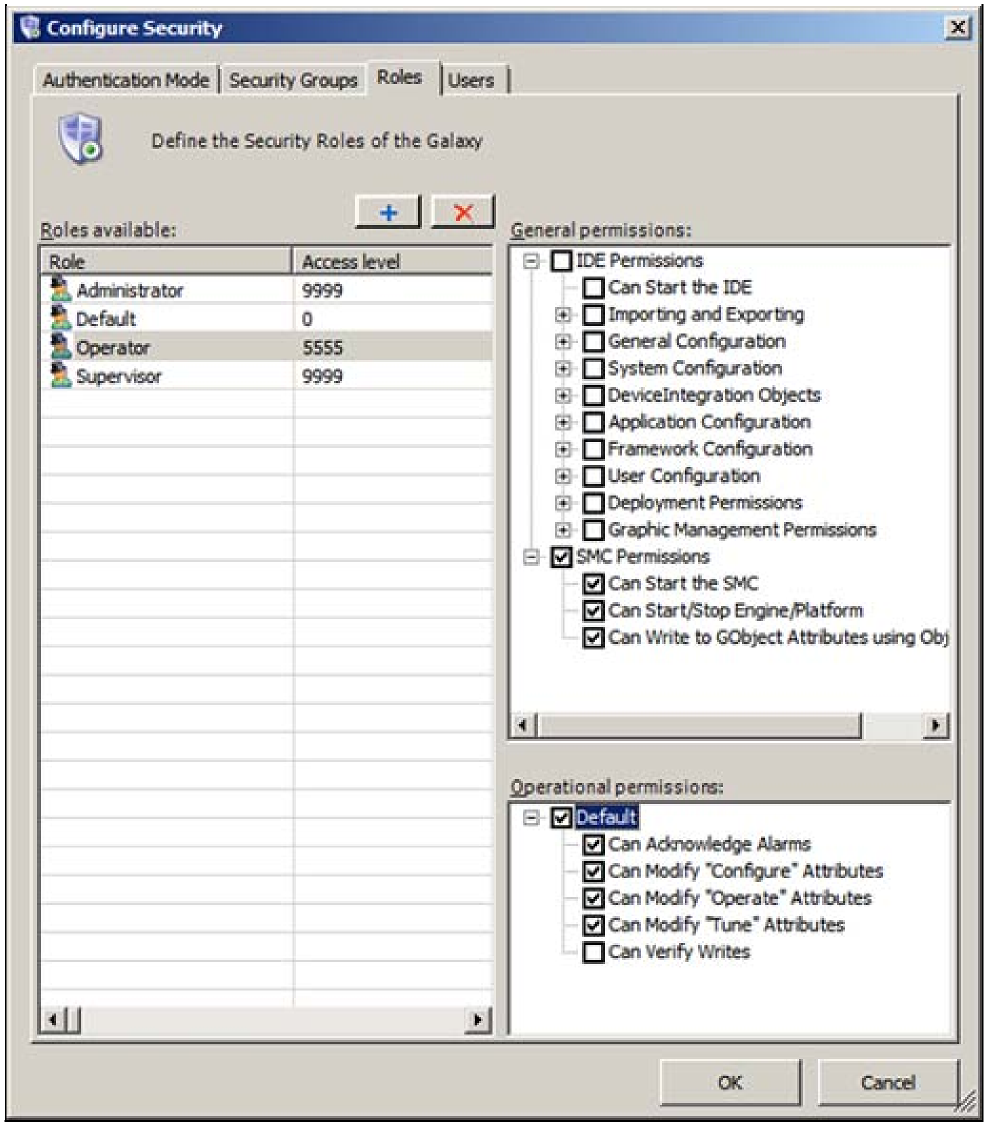
- ☑ IDE Permissions
  - ☑ Can Start the IDE
  - ☑ Importing and Exporting
  - ☑ General Configuration
  - ☑ System Configuration
  - ☑ DeviceIntegration Objects
  - ☑ Application Configuration
  - ☑ Framework Configuration
  - ☑ User Configuration
  - ☑ Deployment Permissions
  - ☑ Graphic Management Permissions
- ☑ SMC Permissions
  - ☑ Can Start the SMC
  - ☑ Can Start/Stop Engine/Platform
  - ☑ Can Write to GObject Attributes using Obj

Operational permissions:

- ☑ Default
  - ☑ Can Acknowledge Alarms
  - ☑ Can Modify "Configure" Attributes
  - ☑ Can Modify "Operate" Attributes
  - ☑ Can Modify "Tune" Attributes
  - ☑ Can Verify Writes

FIGURE 3: CHECK PERMISSIONS FOR SUPERVISOR

15. For the Operator role just check SMC permissions and everything under Default *except for* **Can Verify Writes** (Figure 4 below).



Define the Security Roles of the Galaxy

Roles available:

Role	Access level
Administrator	9999
Default	0
Operator	5555
Supervisor	9999

General permissions:

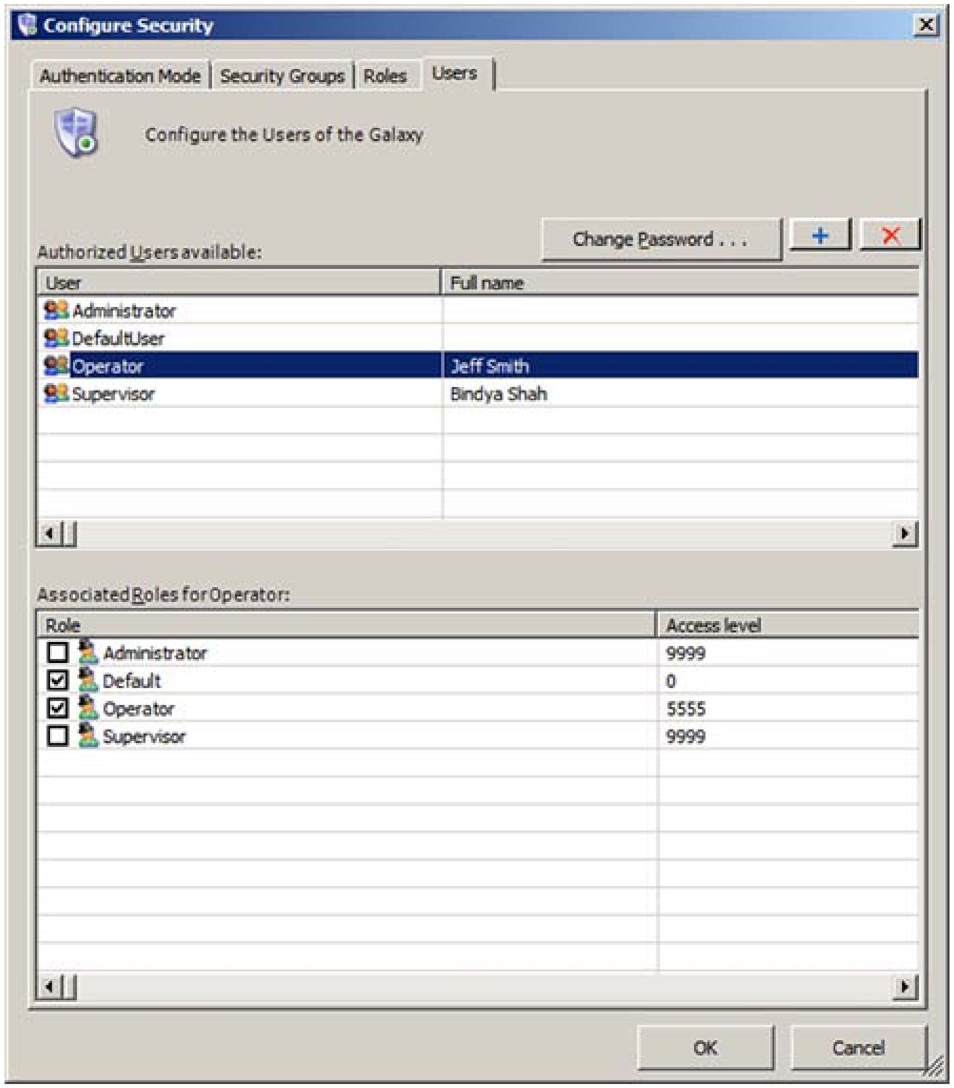
- [ ] IDE Permissions
  - [ ] Can Start the IDE
  - [ ] Importing and Exporting
  - [ ] General Configuration
  - [ ] System Configuration
  - [ ] DeviceIntegration Objects
  - [ ] Application Configuration
  - [ ] Framework Configuration
  - [ ] User Configuration
  - [ ] Deployment Permissions
  - [ ] Graphic Management Permissions
- [x] SMC Permissions
  - [x] Can Start the SMC
  - [x] Can Start/Stop Engine/Platform
  - [x] Can Write to GObject Attributes using Obj

Operational permissions:

- [x] Default
  - [x] Can Acknowledge Alarms
  - [x] Can Modify "Configure" Attributes
  - [x] Can Modify "Operate" Attributes
  - [x] Can Modify "Tune" Attributes
  - [ ] Can Verify Writes

FIGURE 4: OPERATOR PERMISSIONS

16. Click the **Users** tab and make sure you have selected the associated role for each authorized user. For example, **Operator** is associated with the Operator role (Figure 5 below).



Authorized Users available:

User	Full name
Administrator	
DefaultUser	
Operator	Jeff Smith
Supervisor	Bindya Shah

Associated Roles for Operator:

Role	Access level
☐ Administrator	9999
☑ Default	0
☑ Operator	5555
☐ Supervisor	9999

Figure 5: Associated Users and Roles

Examples

This Tech Note includes the following examples. Each section contains script samples you can copy/paste into your Objects.

- **Configuring and using the Secured Write AutomationObject**
- **Configuring and using the Verified Write AutomationObject**

Configuring and using the Secured Write AutomationObject

This example shows how to configure and use the Secured Write AutomationObject.

1. Create a new Galaxy and call it **SecurityApp**.
2. Open the IDE and create the following object instances:
   • $WinPlatform instance called **$WinPlatform_001**.
   • $AppEngine instance called **$AppEngine_001**.
   • $Area instance called **$Area_001**.
   • $UserDefined object instance called **$SecVerUDO**.
3. Create a UDA called **SecUDA** which is an **Integer** data type and make sure that the security type is **Secured Write**.
4. Create an ArchestrA Graphic on the $SecVerUDO called **SecuredWriteGraphic**.
5. Create a Text object and type in **Operator Name** and next to it add another text object **###**.
6. Create a Text object and type in **Secured Write Value**. Next to it add another text object **###** (Figure 6 below).

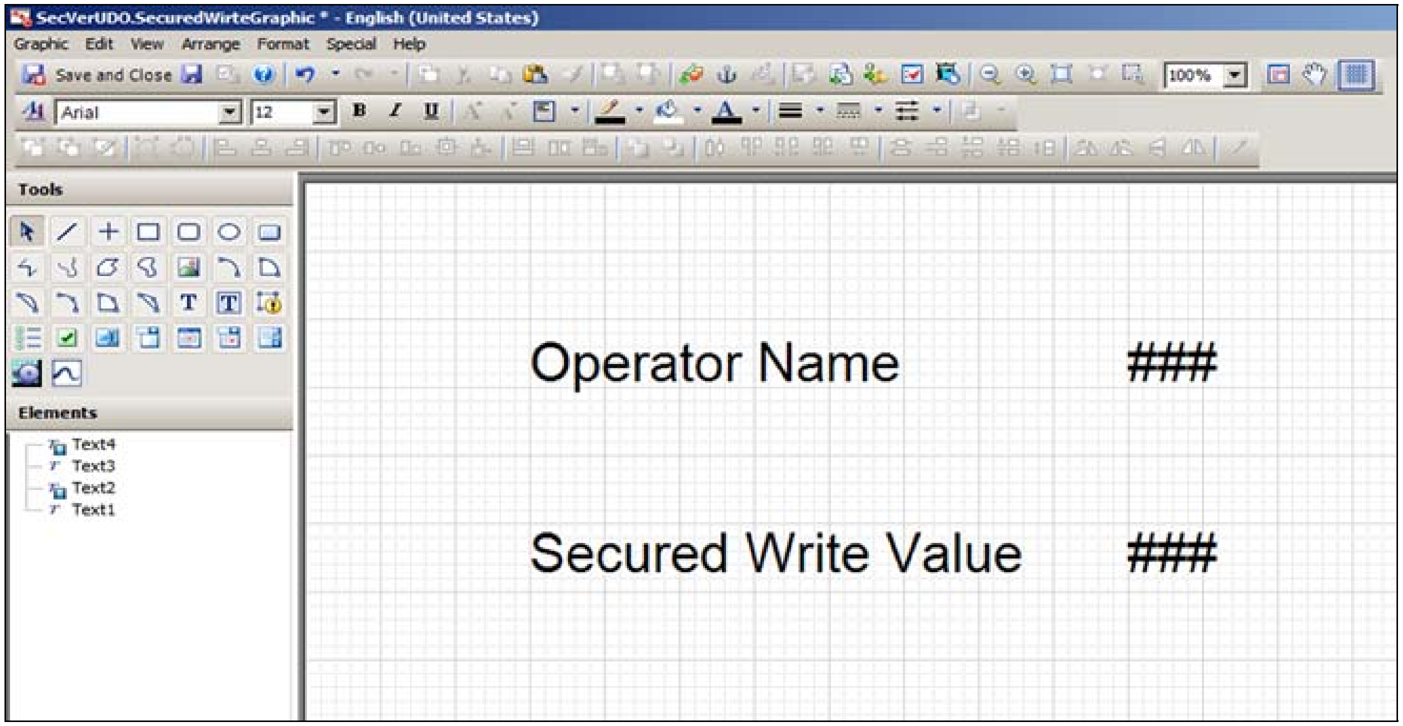
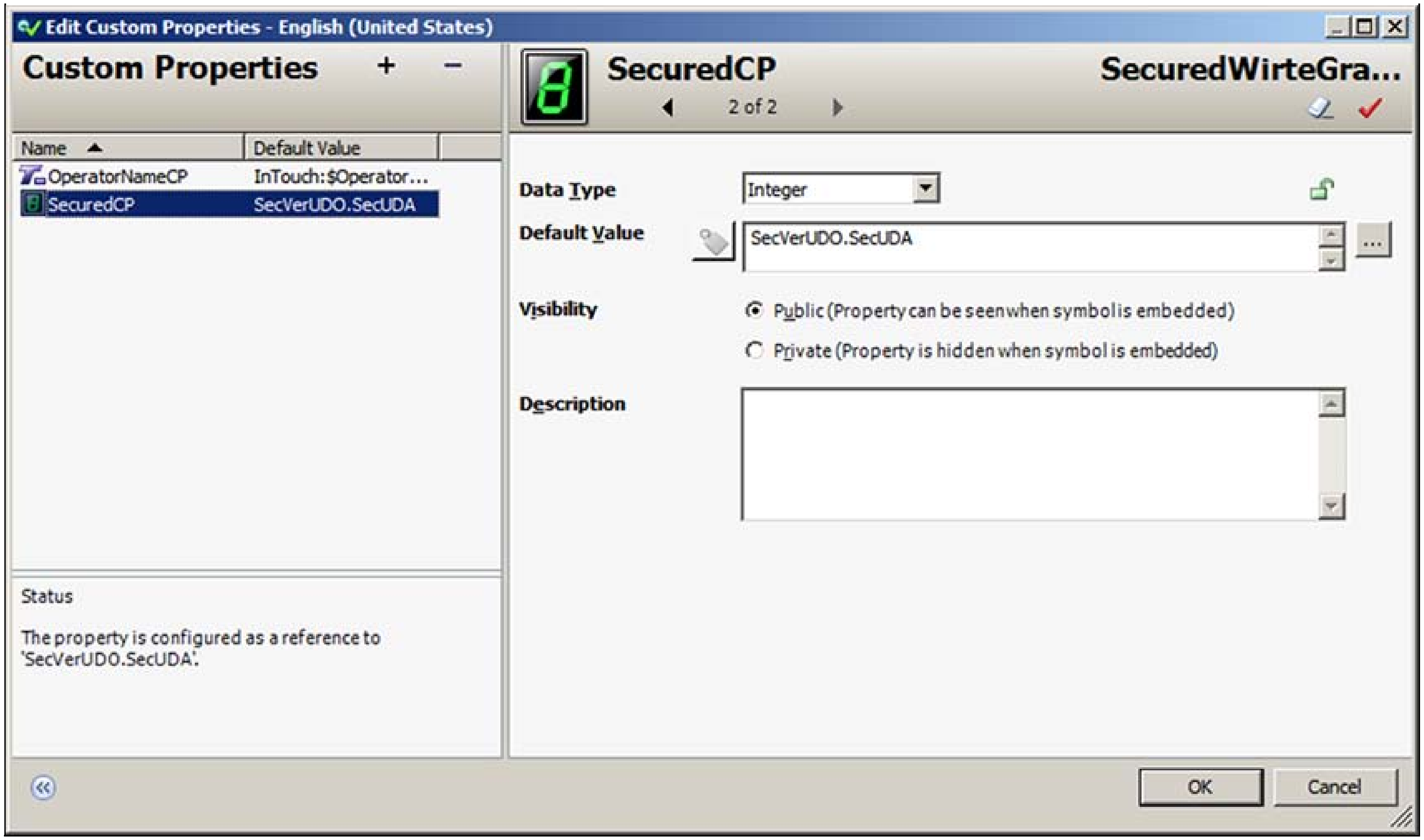


FIGURE 6: TEXT OBJECTS IN SECUREDWRITEGRAPHIC

7. Create the following **Custom Properties**:
   • **OperatorNameCP** which is a **String** data type.
   • **SecuredCP** which is an **Integer** data type.
8. Assign the **OperatorNameCP** to **InTouch:$OperatorName**.
9. Assign the **SecuredCP** to **SecVerUDO.SecUDA** (Figure 7 below).



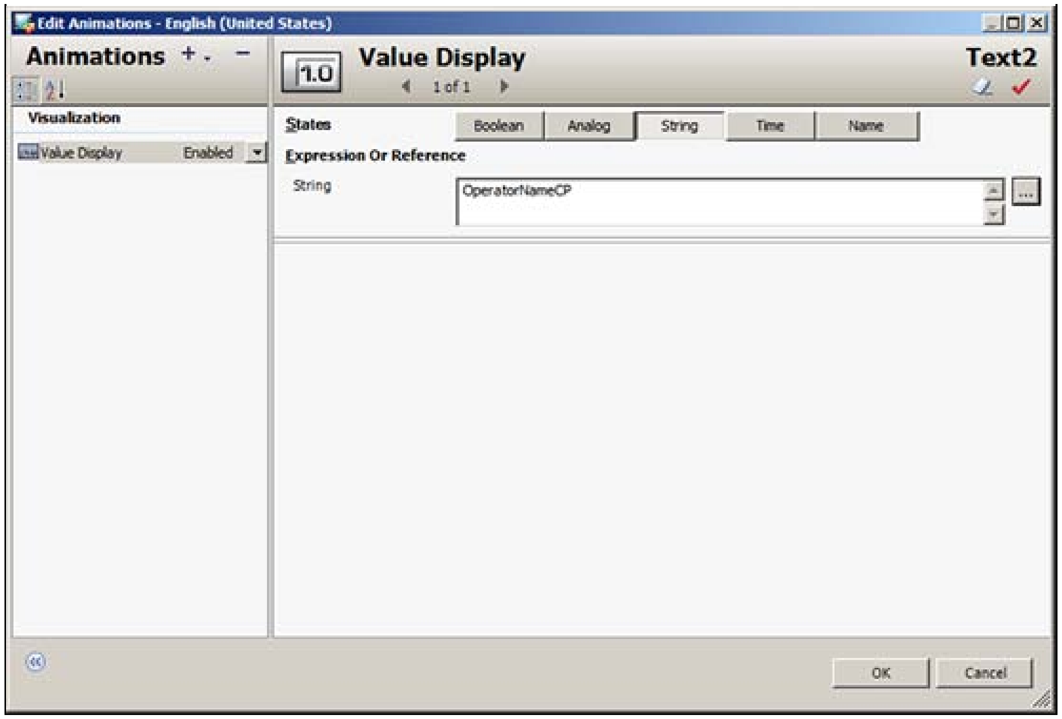
Name	Default Value
OperatorNameCP	InTouch:$Operator...
SecuredCP	SecVerUDO.SecUDA

Data Type: Integer
Default Value: SecVerUDO.SecUDA
Visibility: Public (Property can be seen when symbol is embedded) [✓] / Private (Property is hidden when symbol is embedded)
Description

Status
The property is configured as a reference to 'SecVerUDO.SecUDA'.

Figure 7: Assign Custom Properties

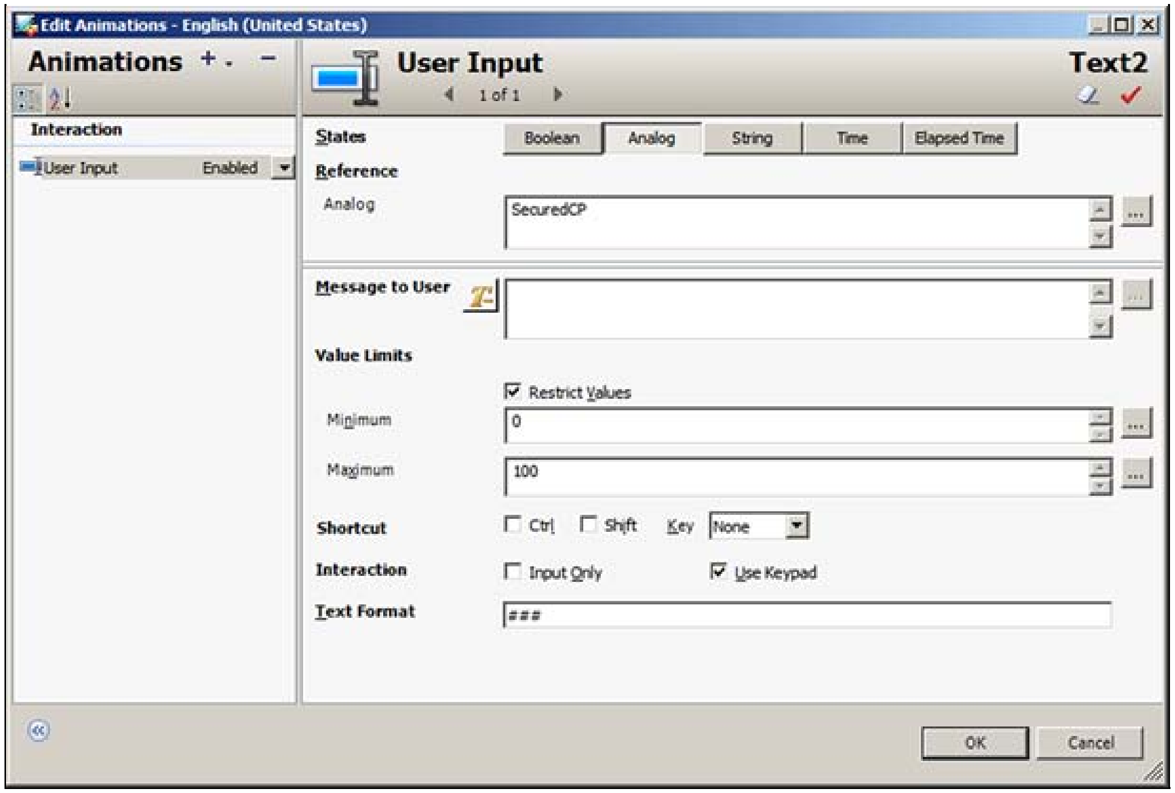
10. Assign a Value Display animation to the **###** next to **Operator Name** text to **OperatorNameCP** (Figure 8 below).



OperatorNameCP

FIGURE 8: VALUE ANIMATION TO OPERATORNAMECP

11. Assign a User Input Analog animation to **###** next to Secured Write Value and reference it to **SecuredCP** (Figure 9 below).



States: Analog

Reference

Analog: SecuredCP

Message to User

Value Limits

☑ Restrict Values

Minimum: 0

Maximum: 100

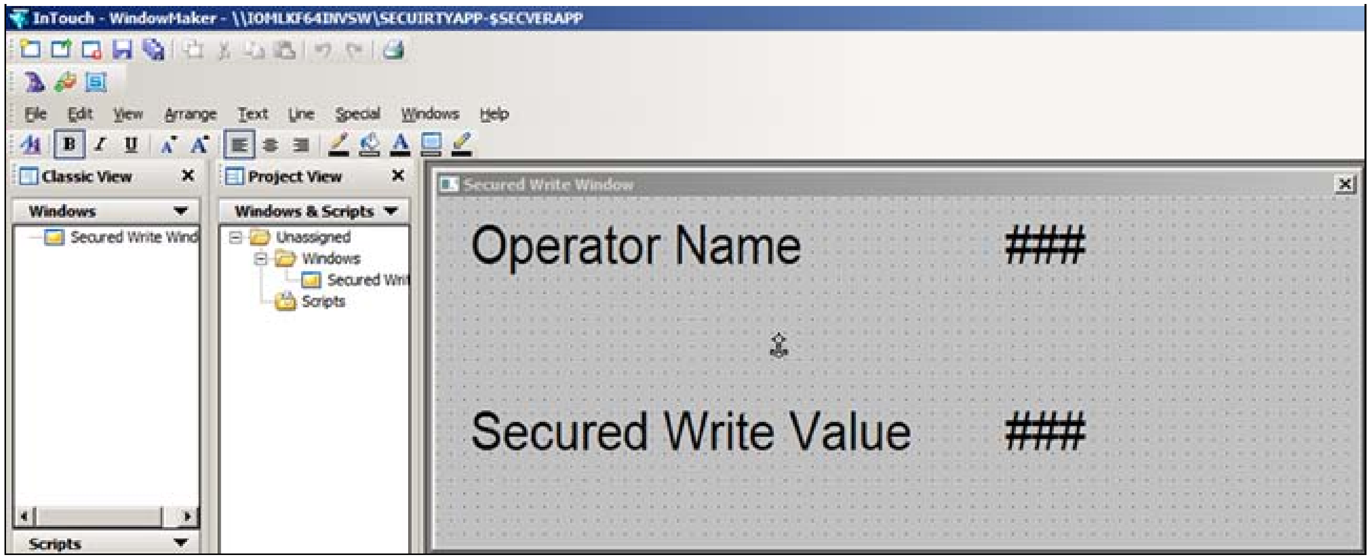
Shortcut: ☐ Ctrl ☐ Shift Key: None

Interaction: ☐ Input Only ☑ Use Keypad

Text Format: ###

Figure 9: User Input Animation to SecuredCP

12. Save and check in the **$SecVerUDO** object.
13. Create a new derived InTouchViewApp called **SecVerApp**.
14. Create an InTouch Window and call it **Secured Write Window**.
15. Embed the **SecuredWriteGraphic** on the InTouch Window (Figure 10 below).



Operator Name ###

Secured Write Value ###

FIGURE 10: EMBEDDED SECUREDWRITEGRAPHIC

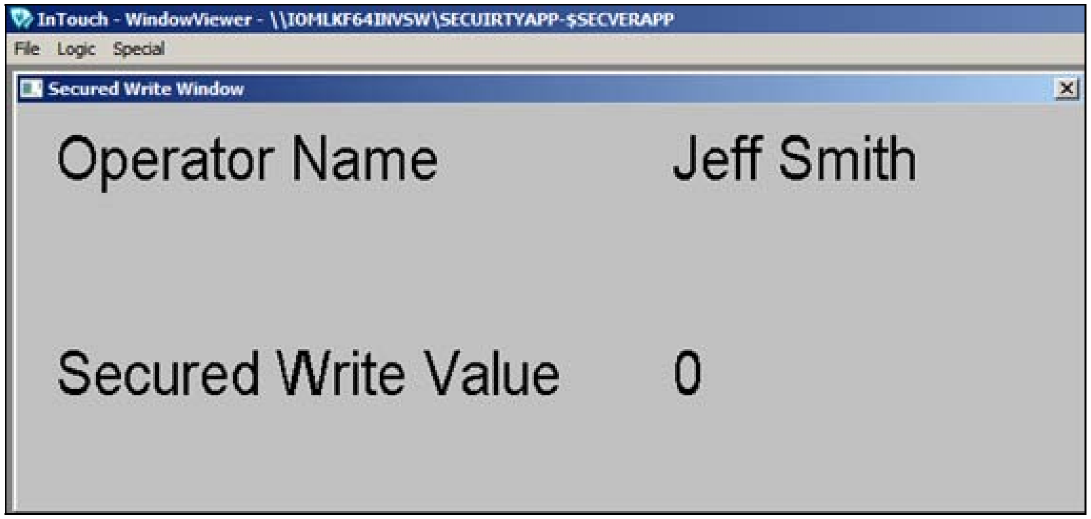
16. Go to **Special-> Security** and click **Select Security Type** as **ArchestrA**.

17. Deploy all the objects (**WinPlatform_001**, **AppEngine_001**, **Area_001**, **SecVerUDO**).

18. Switch to Runtime mode.

19. Click **Special-> Security-> Log on**.

20. Login as **Operator** with password **operator**. Notice if you login as Operator, the **OperatorName** will be Jeff Smith since that is what is configured in Galaxy security.

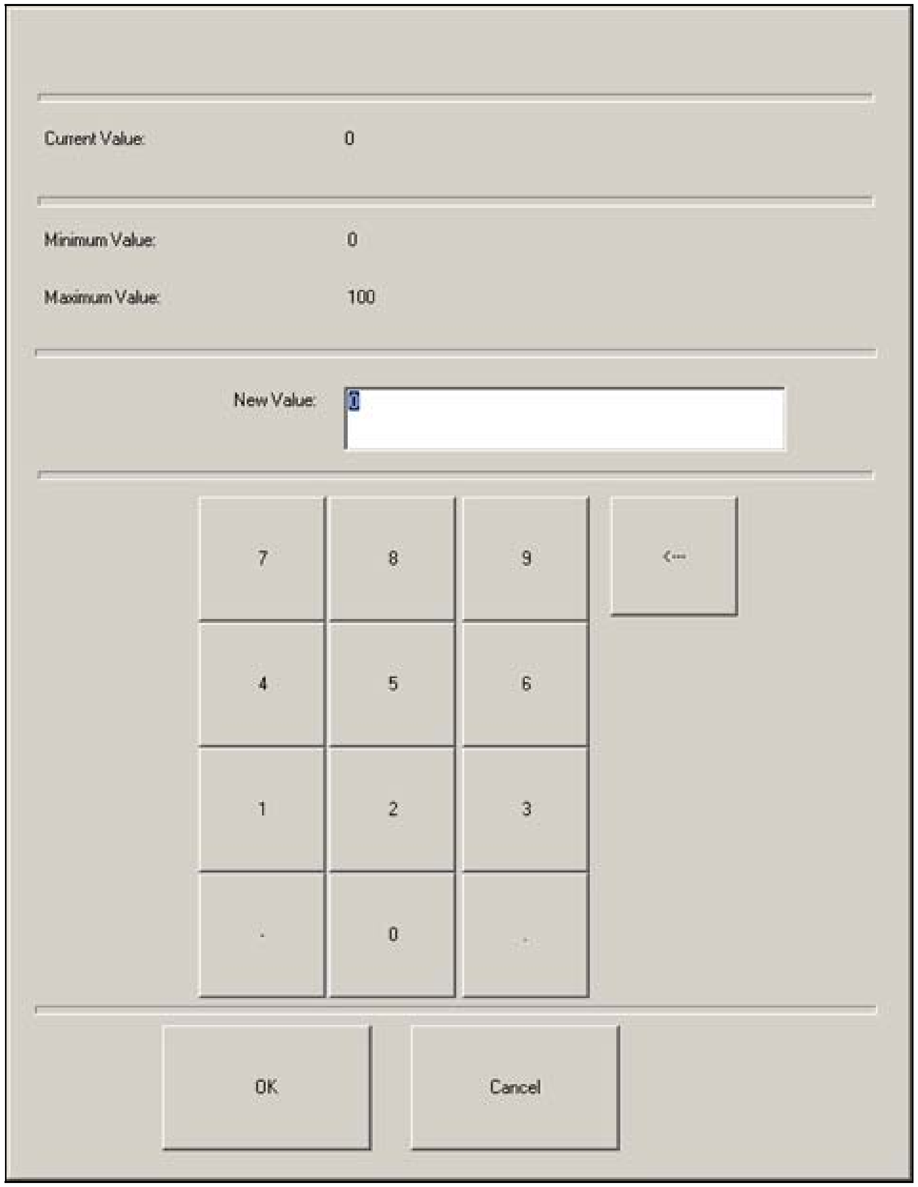

InTouch - WindowViewer - \\IOMLKF64INVSW\SECUIRTYAPP-$SECVERAPP
File Logic Special
Secured Write Window

Operator Name Jeff Smith

Secured Write Value 0

FIGURE 11: SECURED WRITE WINDOW IN RUNTIME

21. Click on the **0**. A keyboard dialog appears.


Current Value: 0

Minimum Value: 0

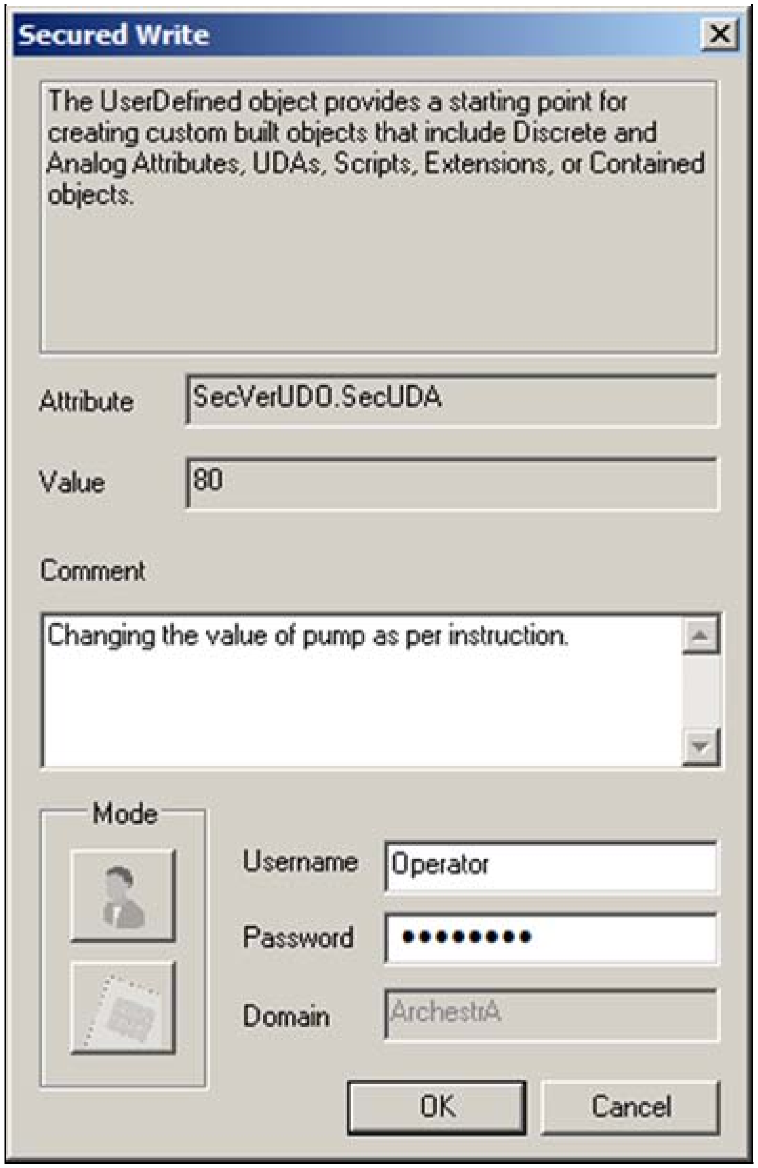
Maximum Value: 100

New Value: 0

FIGURE 12: KEYBOARD INPUT

22. Type a different value and click **OK**. In this example the value is **80**.
23. A **Secured Write** dialog box appears which requires a signature.

    Since we have enabled (in the above security model) Operator changes, add the password for the operator and also a comment about changing the value (Figure 13 below).



The UserDefined object provides a starting point for creating custom built objects that include Discrete and Analog Attributes, UDAs, Scripts, Extensions, or Contained objects.

Attribute: SecVerUDO.SecUDA

Value: 80

Comment: Changing the value of pump as per instruction.

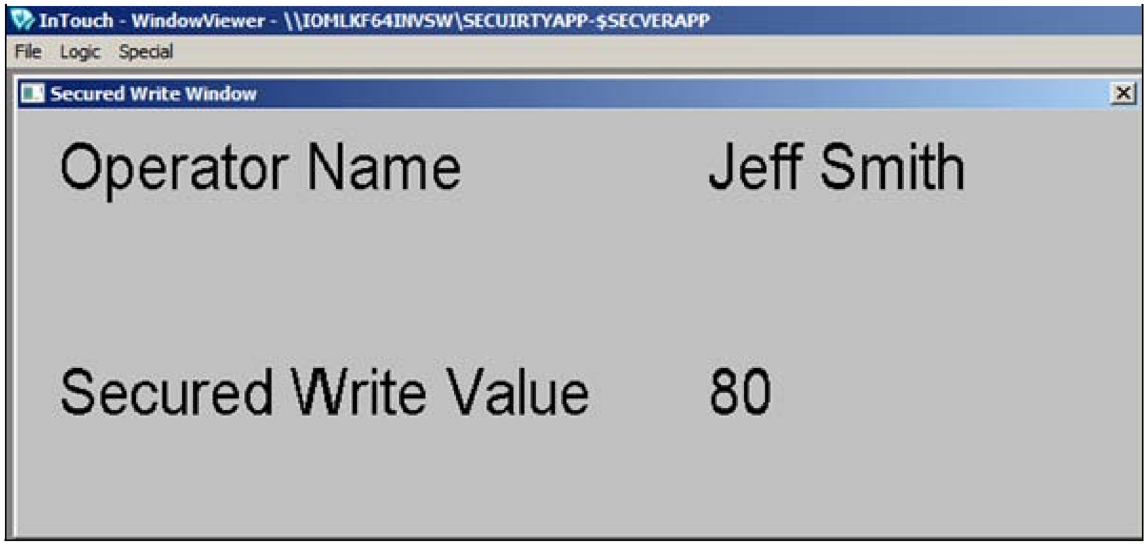
Username: Operator

Password: ●●●●●●●●

Domain: ArchestrA

Figure 13: Add Password and Comment

24. In WindowViewer you notice the change (Figure 14 below).


InTouch - WindowViewer - \\IOMLKF64INVSW\SECUIRTYAPP-$SECVERAPP
File Logic Special
Secured Write Window

Operator Name Jeff Smith

Secured Write Value 80

FIGURE 14: SECURED WRITE VALUE CHANGED IN RUNTIME

Configuring and using the Verified Write AutomationObject

This example shows how to configure and use the Secured Write AutomationObject with the SignedWrite() function.

1. Create a new Galaxy called **SecurityApp**.
2. Create the following object instances:
   - $WinPlatform instance called **$WinPlatform_001**
   - $AppEngine instance called **$AppEngine_001**
   - $Area instance called **$Area_001**
   - $UserDefined object instance called **$SecVerUDO**
3. Create a UDA called **SecUDA** that is an **Integer** data type, and the security type is **Secured Write**.
4. Create an ArchestrA Graphic on the **$SecVerUDO** called **SignedWriteGraphic**.
5. Create a Text object and type **Operator Name**. Next to it add another text object **###**.
6. Create a Text object and type **Secured Write Value**. Next to it add another text object **###**.
7. Create a Button object called **SignedWrite**.

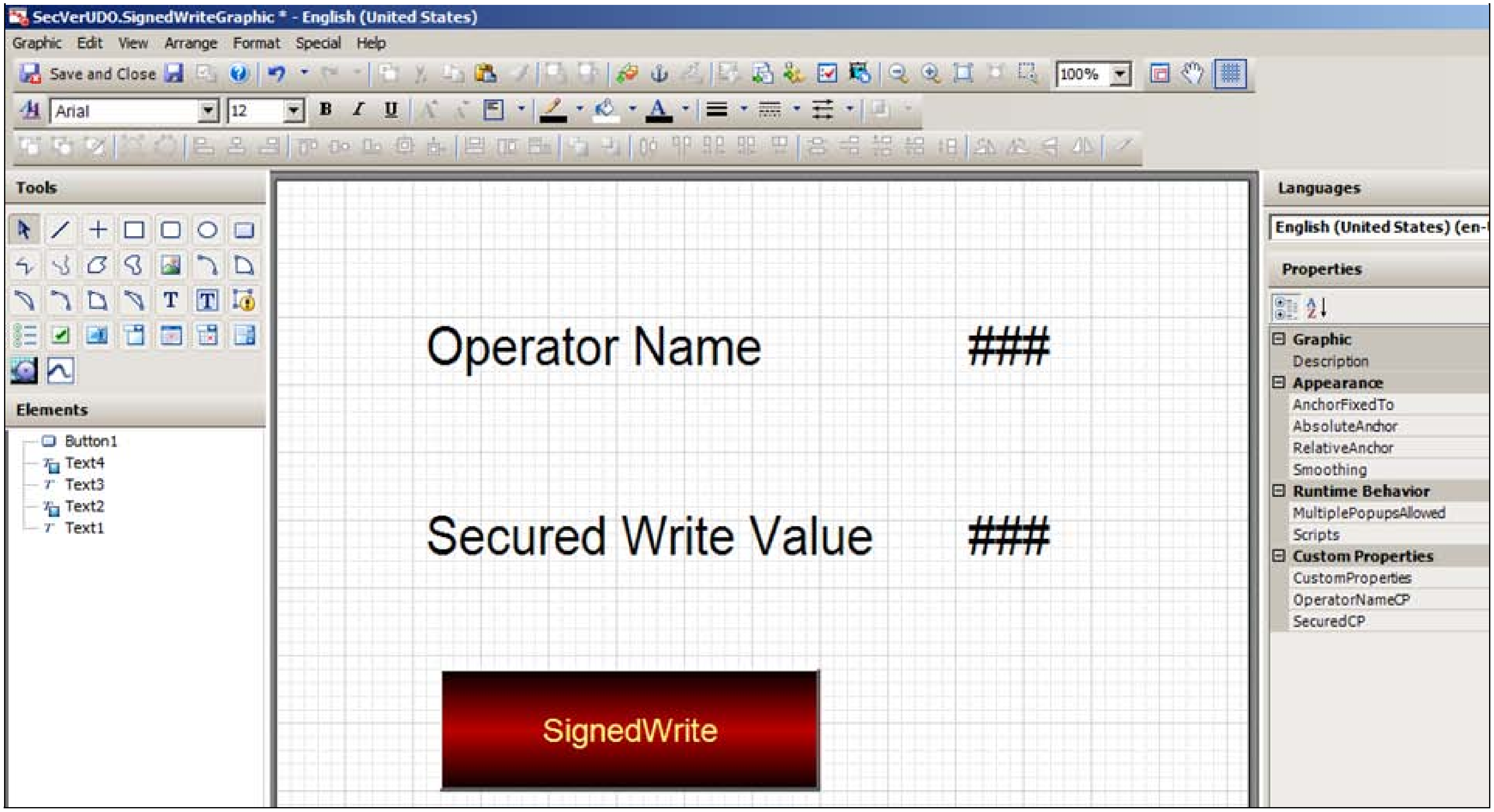
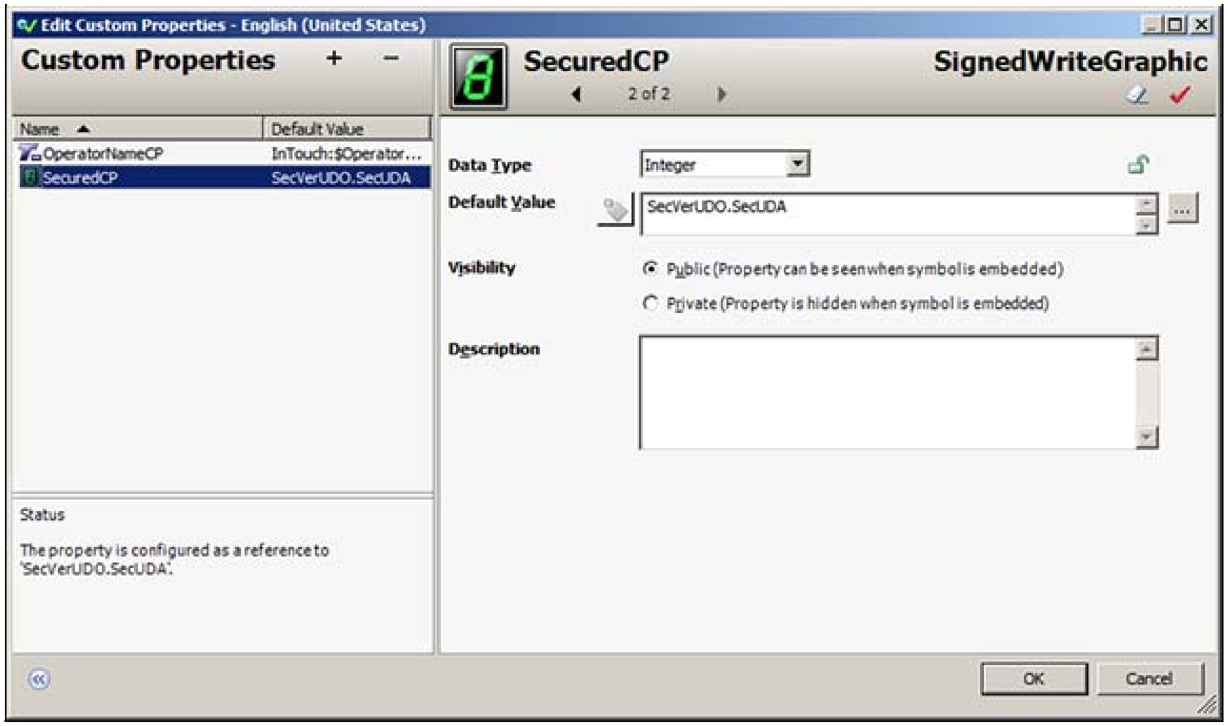


Figure 15: SignedWrite Graphic

8. Create the following custom properties:
   • Custom Property called **OperatorNameCP**: **String** data type.
   • Custom Property called **SecuredCP**: **Integer** data type.
9. Assign the **OperatorNameCP** to **InTouch:$OperatorName**.
10. Assign the **SecuredCP** to **SecVerUDO.SecUDA** (Figure 16 below).



Name	Default Value
OperatorNameCP	InTouch:$Operator...
SecuredCP	SecVerUDO.SecUDA

Data Type: Integer

Default Value: SecVerUDO.SecUDA

Visibility: Public (Property can be seen when symbol is embedded) [✓]; Private (Property is hidden when symbol is embedded)

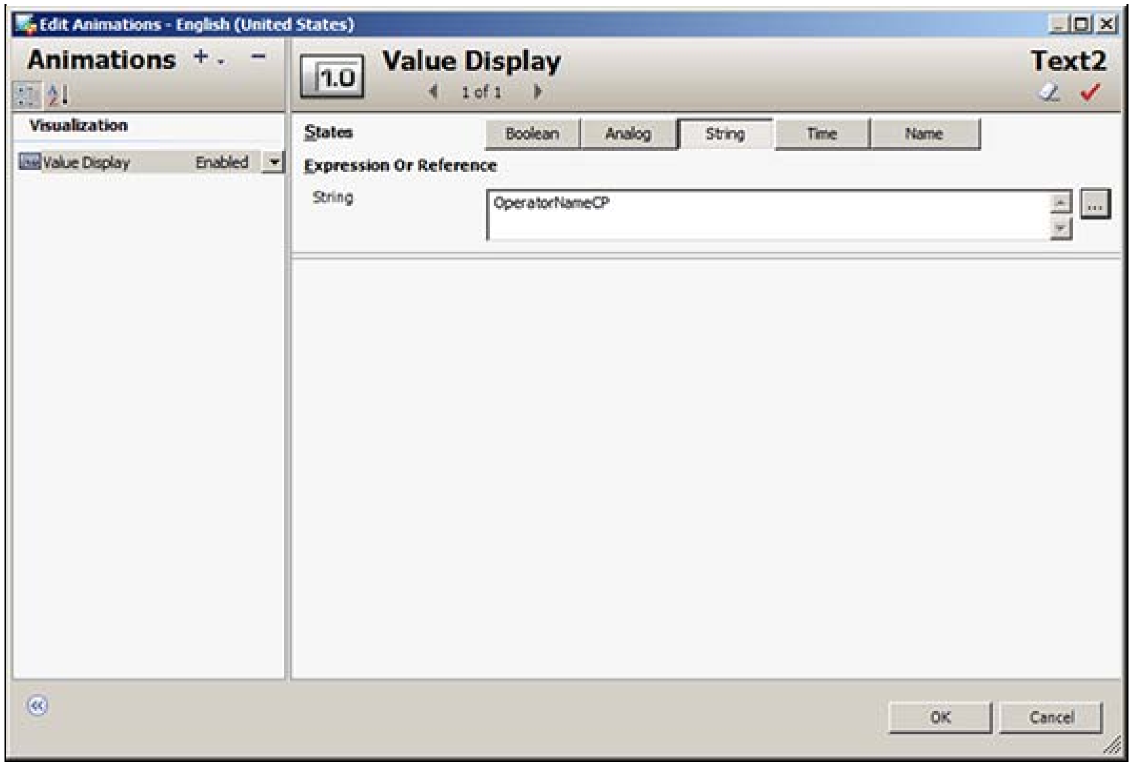
Description:

Status

The property is configured as a reference to 'SecVerUDO.SecUDA'.

Figure 16: Assign Custom Properties

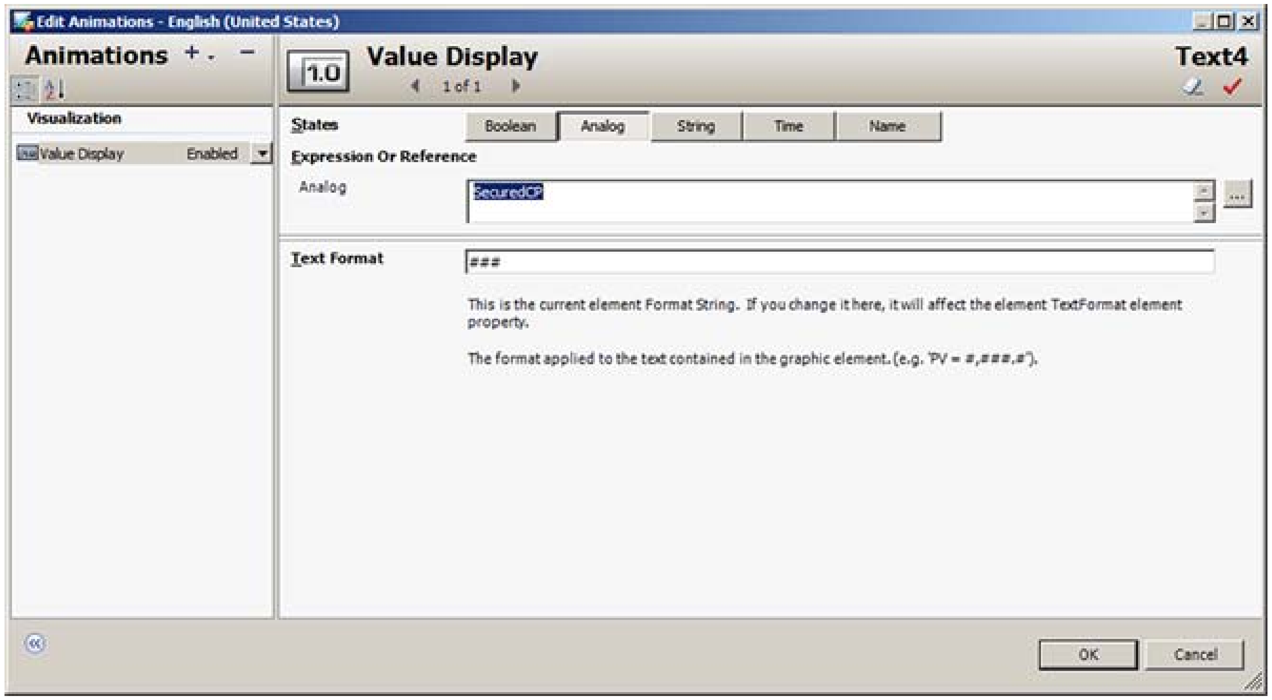
11. Assign a Value Display animation to the ### next to the **Operator Name** text to **OperatorNameCP** (Figure 17 below).



String: OperatorNameCP

FIGURE 17: VALUE DISPLAY ANIMATION FOR OPERATOR NAME TEXT

12. Assign a Value Display Analog animation to the ### next to **Secured Write Value** and reference it to **SecuredCP** (Figure 18 below).



Analog: SecuredCP

Text Format: ###

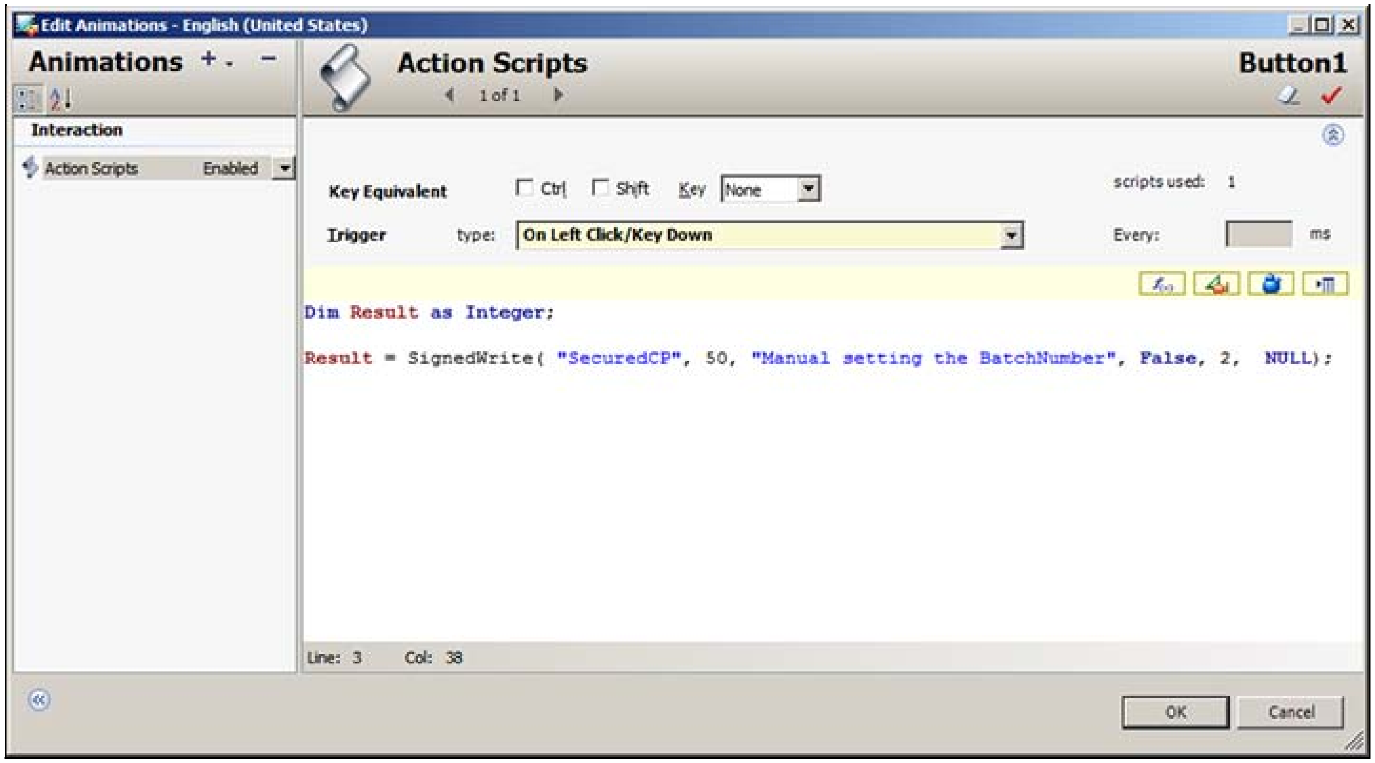
This is the current element Format String. If you change it here, it will affect the element TextFormat element property.

The format applied to the text contained in the graphic element. (e.g. 'PV = #,###.#').

FIGURE 18: ASSIGN ANIMATION TO SECURED WRITE VALUE

13. Add the following Action Script on the **SignedWrite** button:

```
Dim Result as Integer;
Result = SignedWrite( "SecuredCP", 50, "Manual setting the BatchNumber", False, 2, NULL);
```



Key Equivalent ☐ Ctrl ☐ Shift Key None

Trigger type: On Left Click/Key Down

Every: ms

```
Dim Result as Integer;

Result = SignedWrite( "SecuredCP", 50, "Manual setting the BatchNumber", False, 2,  NULL);
```

Line: 3 Col: 38

FIGURE 19: BUTTON ACTION SCRIPT

14. Save and Check In the **$SecVerUDO** object.
15. Create a new derived InTouchViewApp called **SecVerApp**.
16. Create an InTouch Window call it SignedWrite Window.
17. Embed the **SignedWriteGraphic** on the InTouch Window (Figure 20 below).

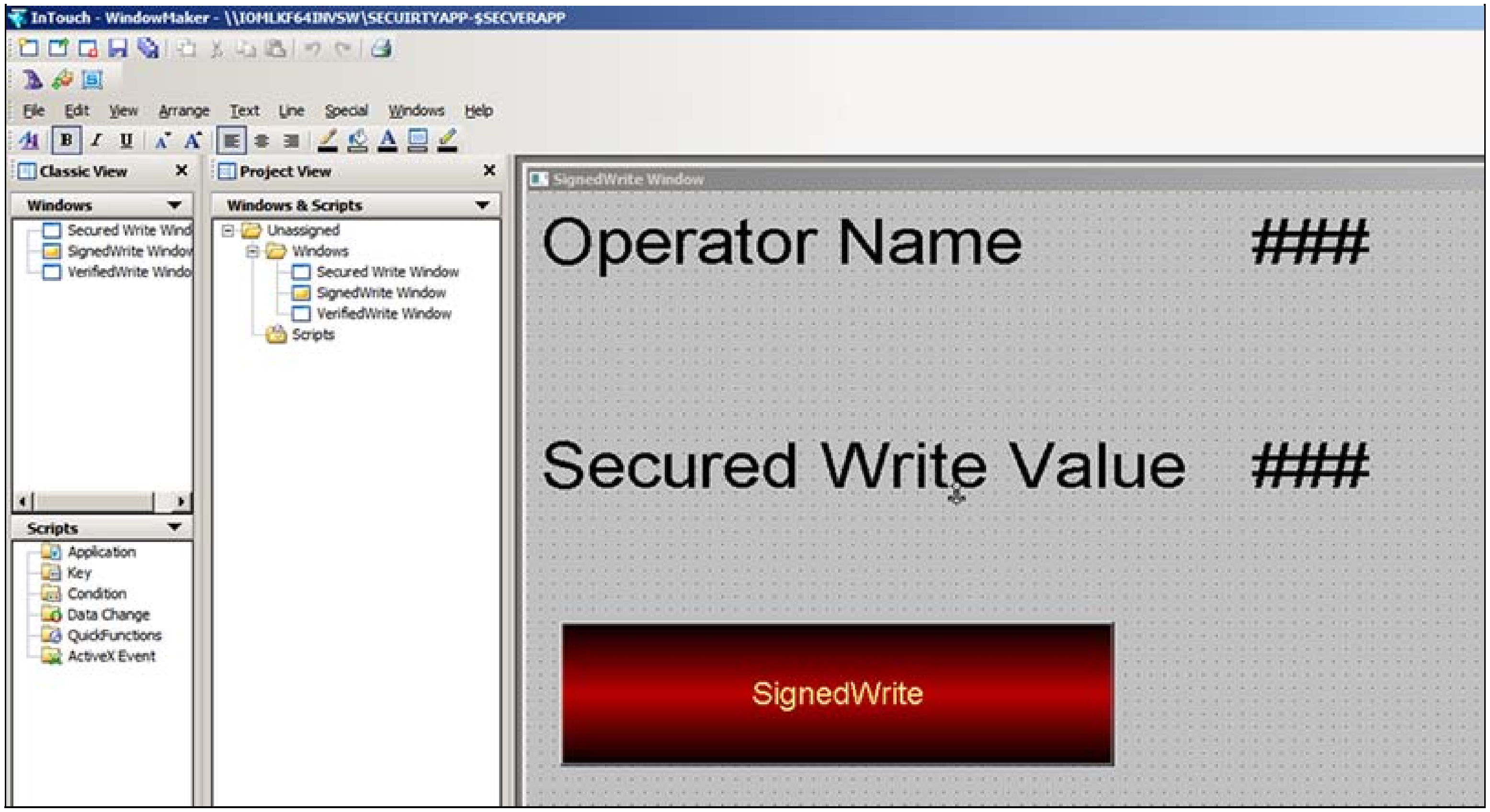


FIGURE 20: EMBEDDED SIGNEDWRITE GRAPHIC

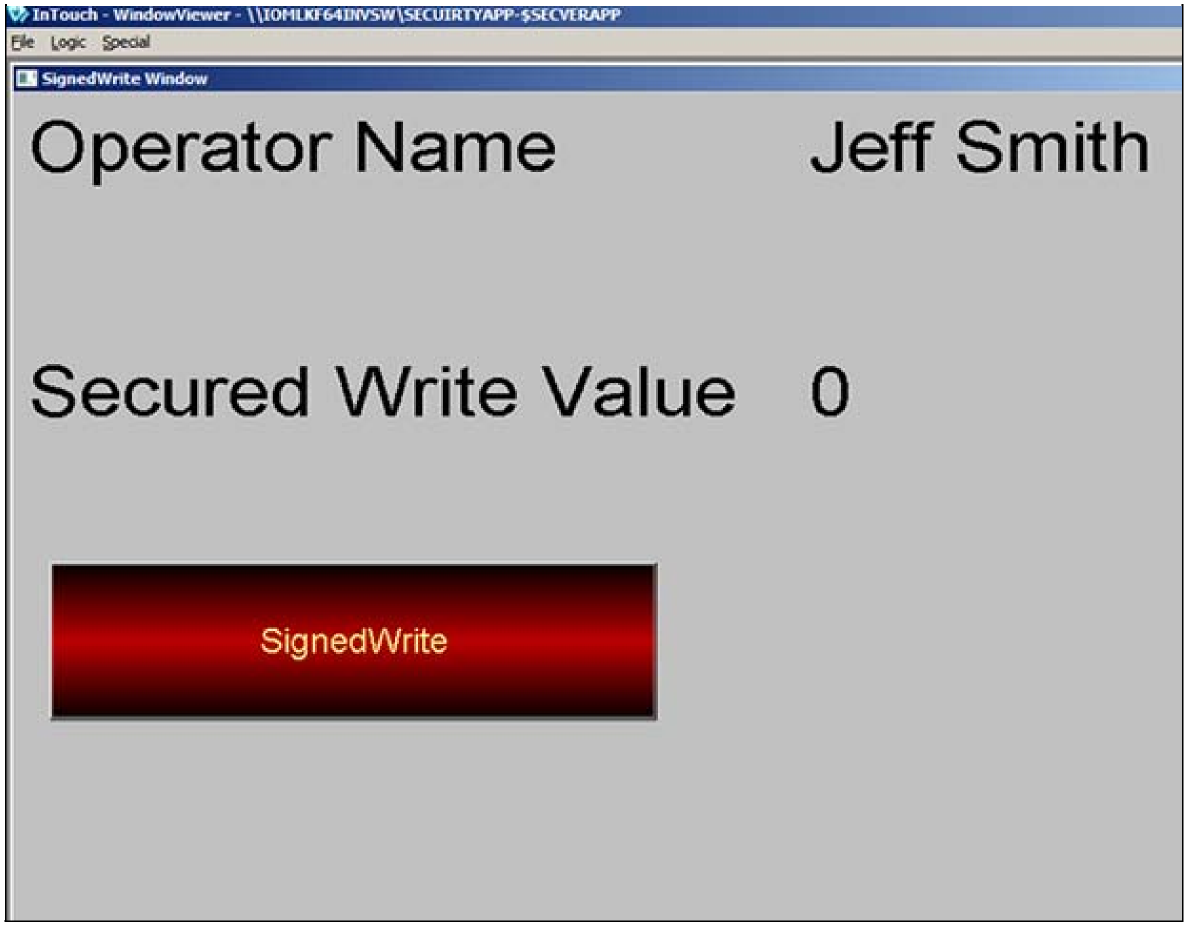
18. Go to **Special-> Security-> Select Security Type** as **ArchestrA**.

19. Deploy all your objects. (WinPlatform_001, AppEngine_001, Area_001, SecVerUDO).

20. Switch to Runtime mode.

21. Click **Special->Security->Log on**.

22. Login as **Operator** and password **operator** according to the above security model. Notice if you login as Operator then the OperatorName will be Jeff Smith as that is what is configured in the Galaxy security above.


InTouch - WindowViewer - \\IOMLKF64INVSW\SECUIRTYAPP-$SECVERAPP
File Logic Special
SignedWrite Window
SignedWrite

Operator Name Jeff Smith

Secured Write Value 0

Figure 21: Operator Name and Value in Runtime

23. Now click on the SignedWrite button and notice the secured write dialog box will come up and ask for signature to change the value as shown below.

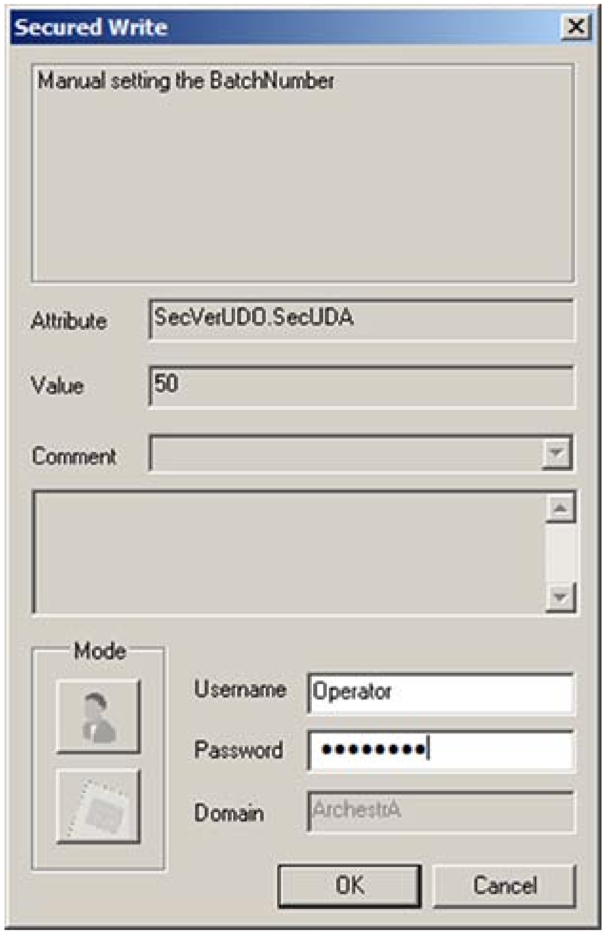
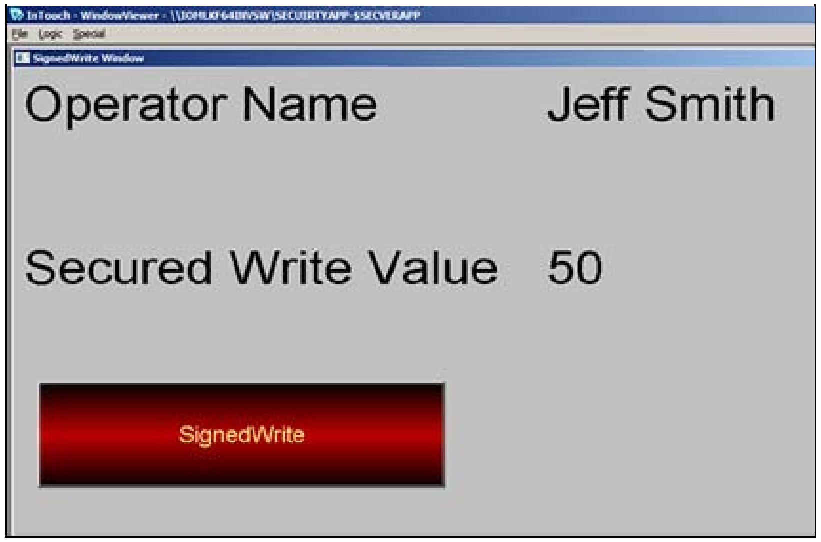


FIGURE 22: SECURED WRITE DIALOG

We cannot edit the comment since we set the coded value to False for the field where **Comment_is_Editable**.

24. Click **OK** after typing the correct password. Since the Operator had permissions to modify the value changed and now in WindowViewer you notice the change as shown below.



Operator Name Jeff Smith

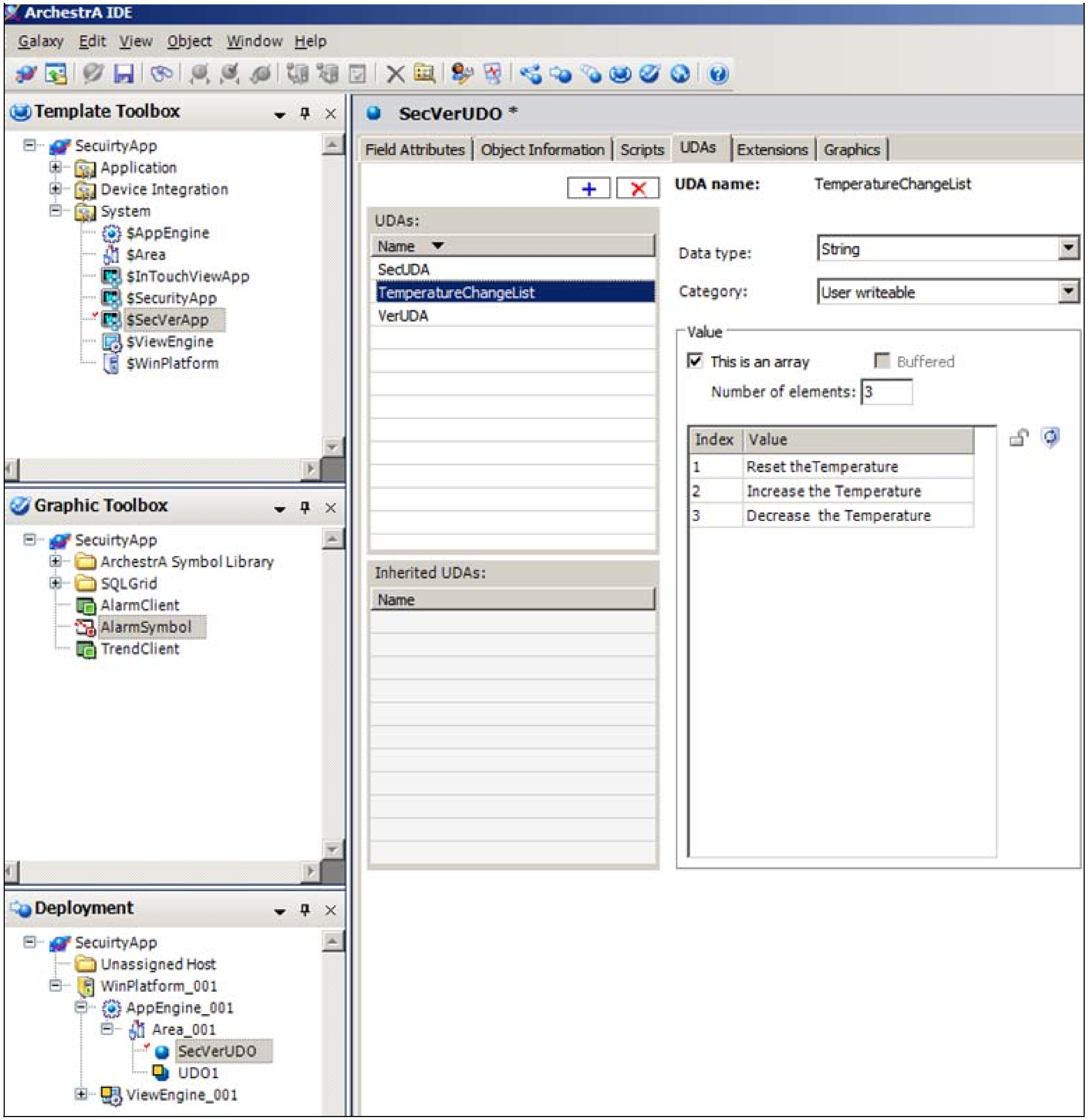
Secured Write Value 50

FIGURE 23: SECURED WRITE COMPLETE

Configuring and Using the Verified Write AutomationObject

This example shows how to configure and use the Secured Write AutomationObject with the SignedWrite() function.

1. Create a new Galaxy e.g. SecurityApp.
2. Create the following object instances:
   - $WinPlatform instance called **$WinPlatform_001**
   - $AppEngine instance called **$AppEngine_001**
   - $Area instance called **$Area_001**
   - $UserDefined object instance called **$SecVerUDO**
3. Create a UDA called **VerUDA** which is an **Integer** data type and ensure that the security type is **Verified Write**.
4. Create another UDA called **TemperatureChangeList** which is a String Array. This UDA has 3 elements (Figure 24 below).



UDA name: TemperatureChangeList

Data type: String

Category: User writeable

Value

[✓] This is an array [ ] Buffered

Number of elements: 3

Index	Value
1	Reset theTemperature
2	Increase the Temperature
3	Decrease the Temperature

FIGURE 24: TEMPERATURE CHANGE ARRAY

5. Create an ArchestrA Graphic on the $SecVerUDO called **VerifiedWriteGraphic**.

6. Create the following text elements (Figure 25 below).
   - Text object **Operator Name** and text object **###**
   - Text object **Sec\Ver Value** and text object **###**
   - Button object **SignedWrite**

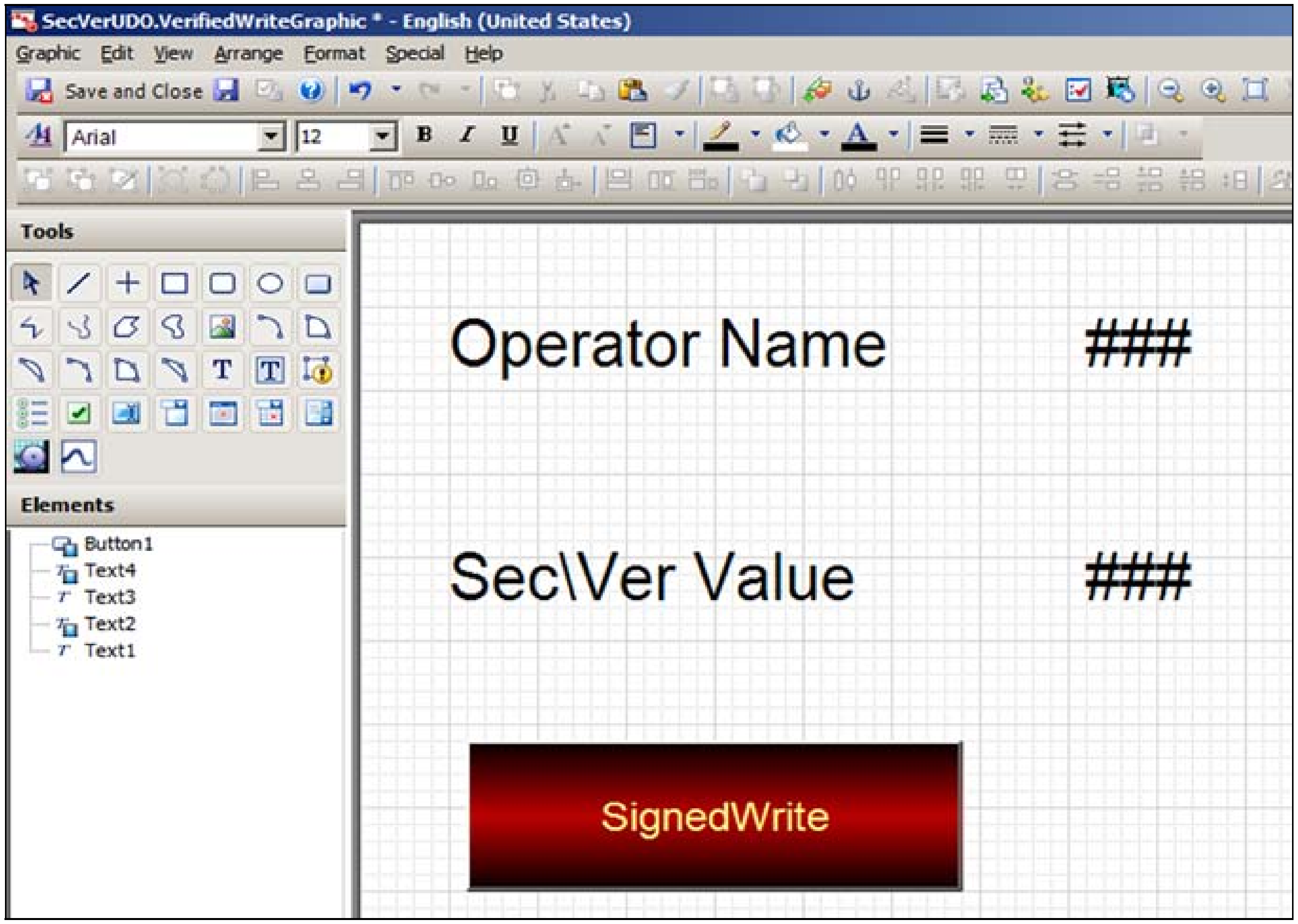
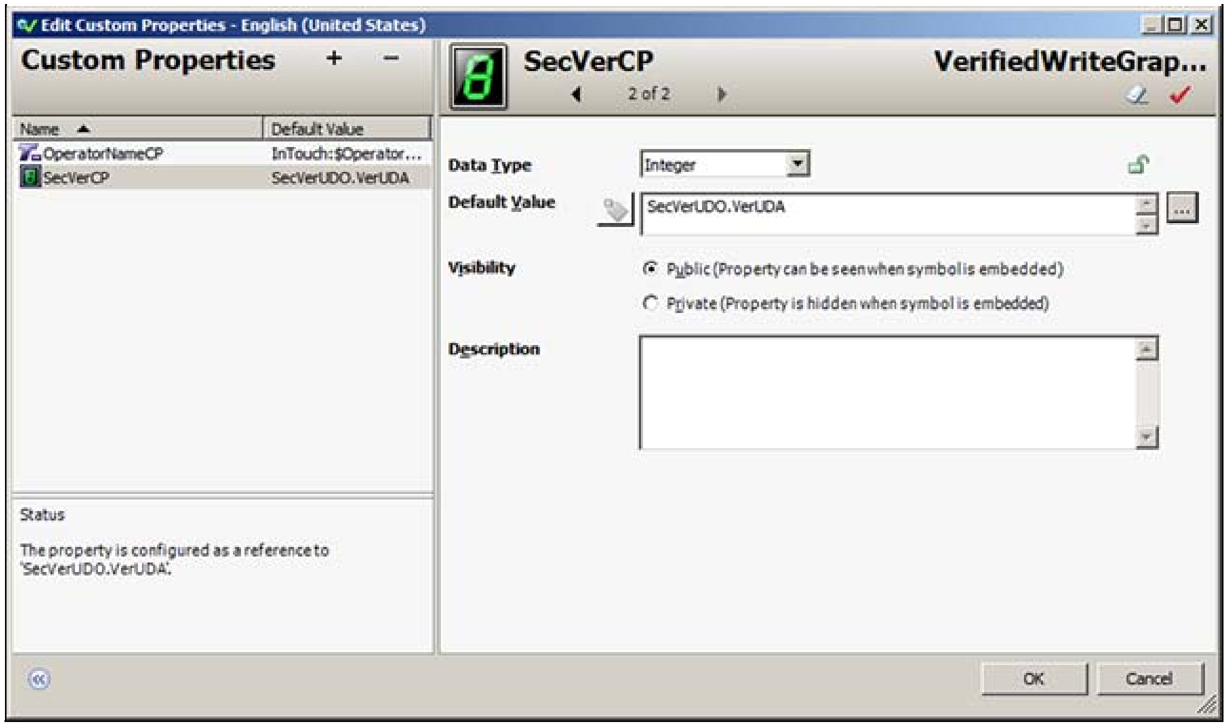


FIGURE 25: SECUREDWRITE GRAPHIC ELEMENTS

7. Create the following Custom Properties
   - Custom Property **OperatorNameCP**: **String** data type
   - Custom Property **SecVerCP**: **Integer** data type

8. Assign the OperatorNameCP to **InTouch:$OperatorName**.

9. Assign the SecVerCP to **SecVerUDO.VerUDA** as shown below


Edit Custom Properties - English (United States)

Custom Properties

Name	Default Value
OperatorNameCP	InTouch:$Operator...
SecVerCP	SecVerUDO.VerUDA

Status

The property is configured as a reference to 'SecVerUDO.VerUDA'.

SecVerCP — 2 of 2 — VerifiedWriteGrap...

Data Type: Integer

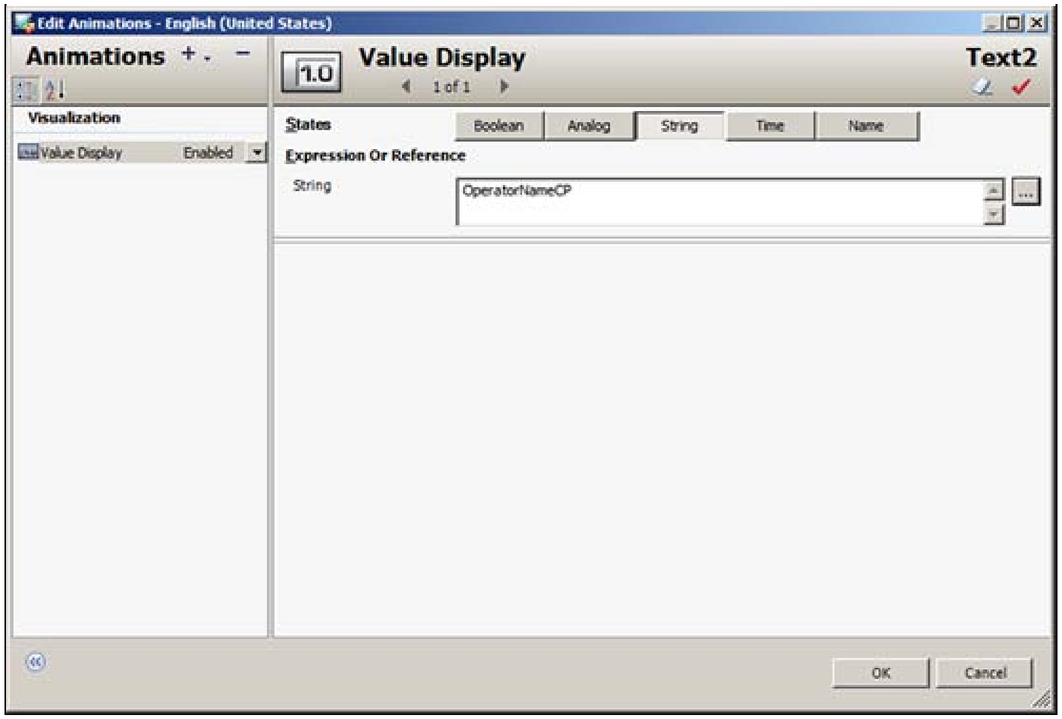
Default Value: SecVerUDO.VerUDA

Visibility: (●) Public (Property can be seen when symbol is embedded) ( ) Private (Property is hidden when symbol is embedded)

Description:

Figure 26: Custom Property SecVerCP

10. Assign a Value Display animation to the ### (for Operator Name text) to **OperatorNameCP** (Figure 27 below).



Value Display

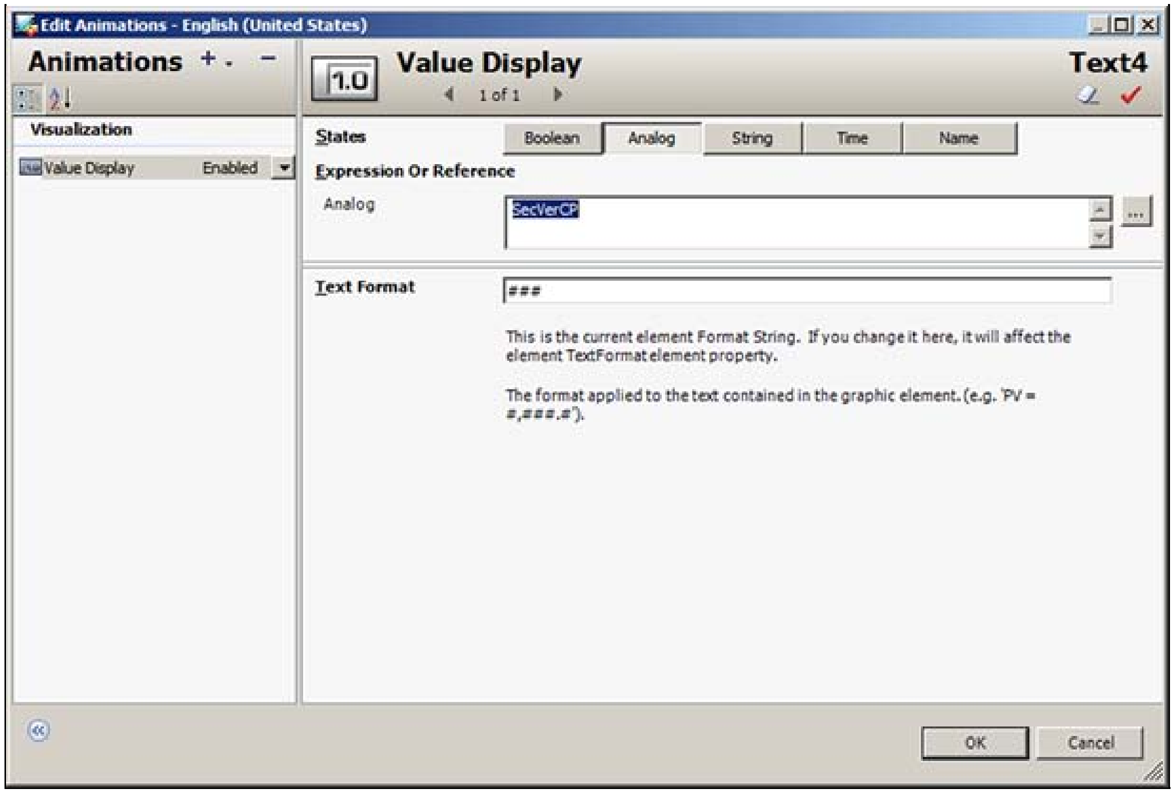
States: String

Expression Or Reference

String: OperatorNameCP

FIGURE 27: OPERATORNAMECP VALUE DISPLAY

11. Assign a Value Display Analog animation to ### for the **Sec\Ver** Value and reference it to **SecVerCP** (Figure 28 below).



Expression Or Reference

Analog: SecVerCP

Text Format: ###

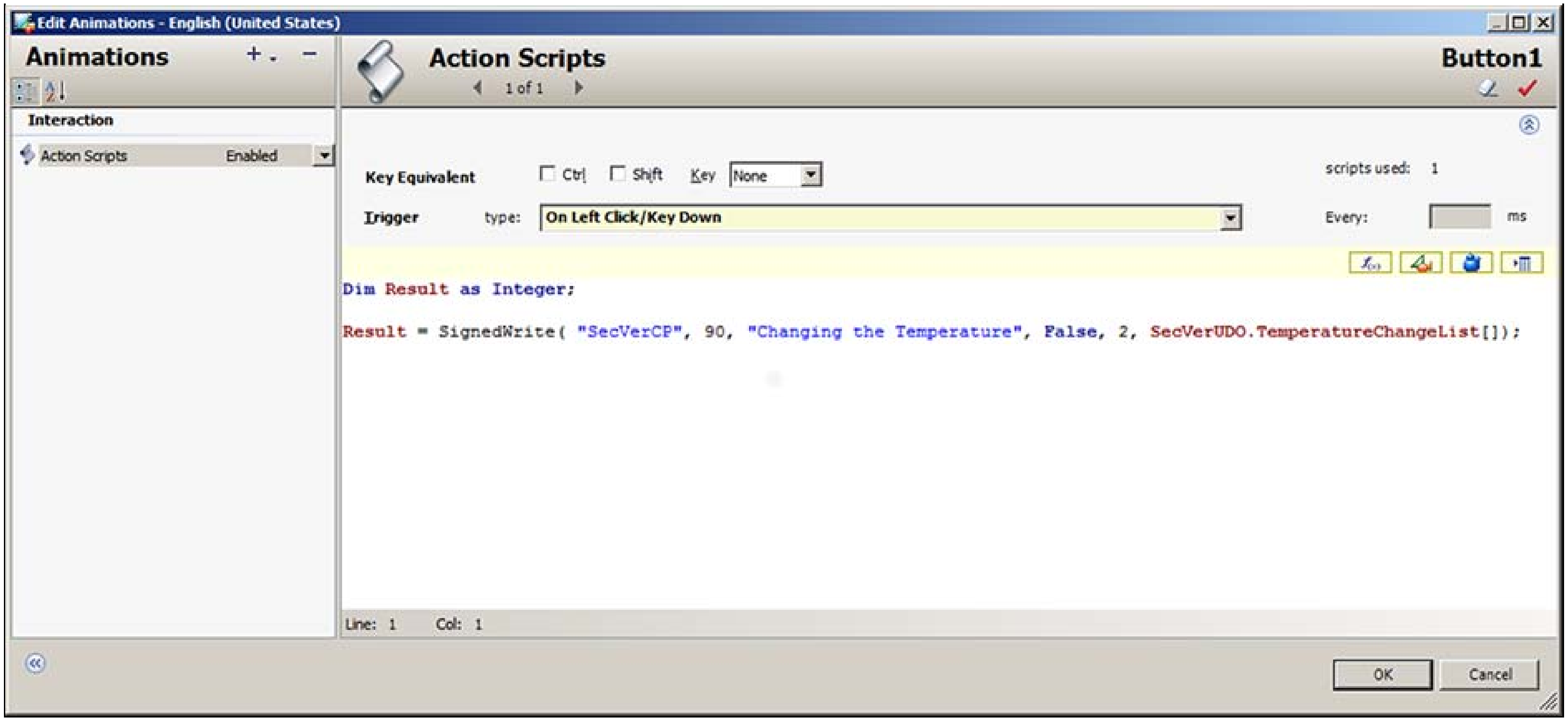
This is the current element Format String. If you change it here, it will affect the element TextFormat element property.

The format applied to the text contained in the graphic element. (e.g. 'PV = #,###.#').

FIGURE 28: SECVERCP VALUE DISPLAY

12. Add the following Action Script on the **SignedWrite** button:

```
Dim Result as Integer;
Result = SignedWrite( "SecVerCP", 90, "Changing the Temperature", False, 2, SecVerUDO.TemperatureChangeList[]);
```



```
Dim Result as Integer;

Result = SignedWrite( "SecVerCP", 90, "Changing the Temperature", False, 2, SecVerUDO.TemperatureChangeList[]);
```

Figure 29: Script Editor

13. Save and CheckIn the $SecVerUDO object.
14. Create a new derived InTouchViewApp called **SecVerApp**.
15. Create an InTouch Window call it VerifiedWrite Window.
16. Embed the VerifiedWriteGraphic on the InTouch Window (Figure 30 below).

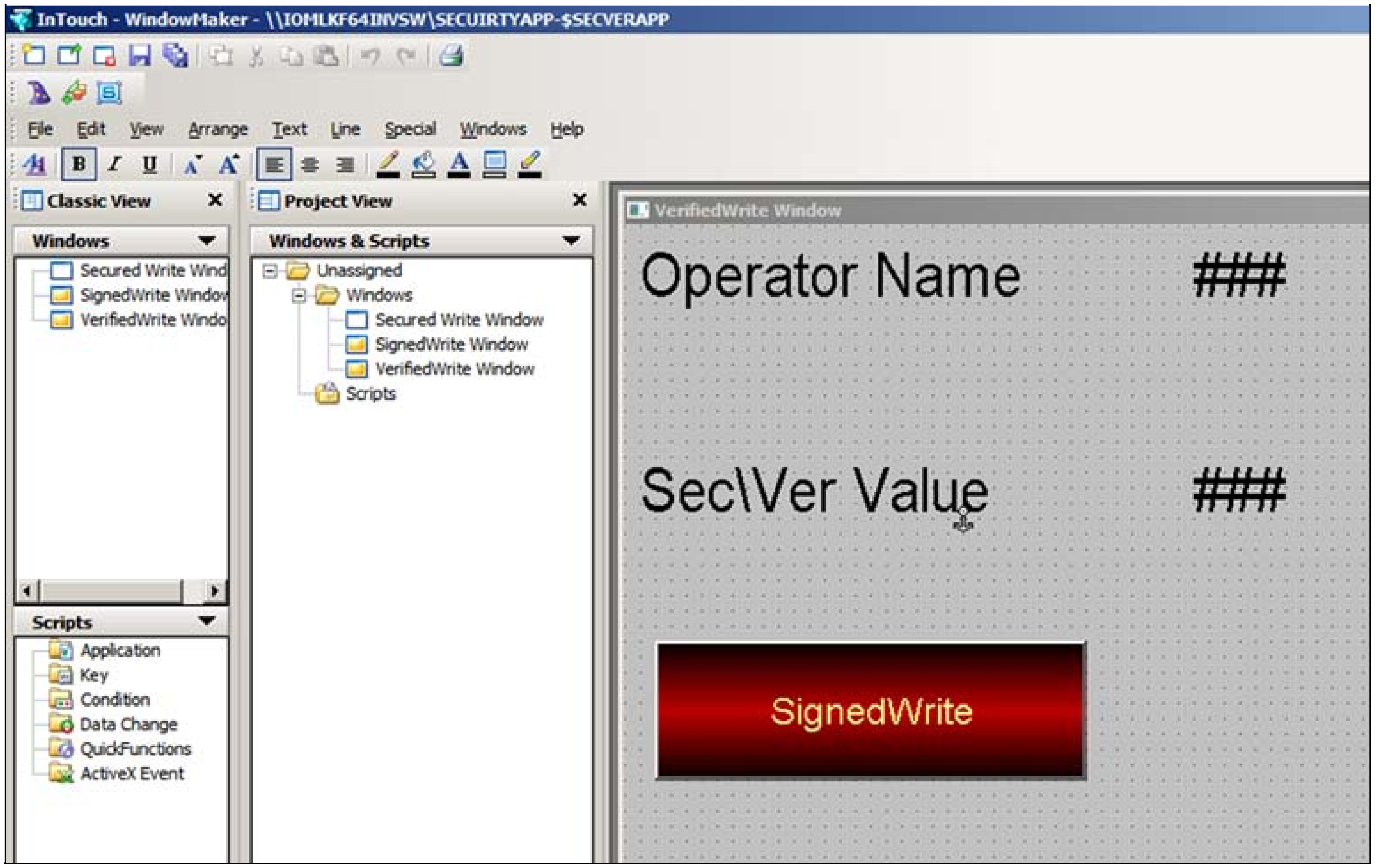
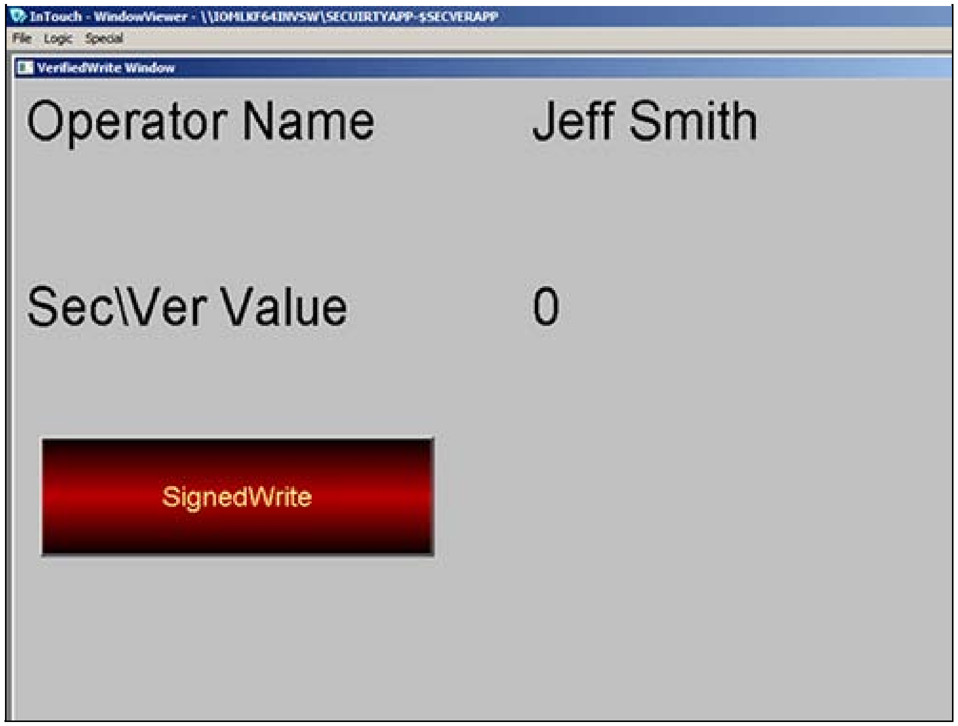


Figure 30: Embedded VerifiedWrite Window

17. On the main menu, click **Special->Security->Select Security Type** as **ArchestrA**.
18. Deploy all the objects. (WinPlatform_001, AppEngine_001, Area_001, SecVerUDO).
19. Switch to Runtime mode.
20. Click **Special->Security->Log on**.

    Login as **Operator** and password **operator** as the above security model. Notice if you login as Operator then the OperatorName will be Jeff Smith since that is what is configured in the Galaxy security above.


InTouch - WindowViewer - \\IOMLKF64INVSW\SECUIRTYAPP-$SECVERAPP
File Logic Special
VerifiedWrite Window
SignedWrite

Operator Name Jeff Smith

Sec\Ver Value 0

FIGURE 31: SECURED OPERATOR LOGIN

21. Now click on the **SignedWrite** button and notice the Verified Write dialog box appears. It asks for signature to change the value (Figure 32 below).

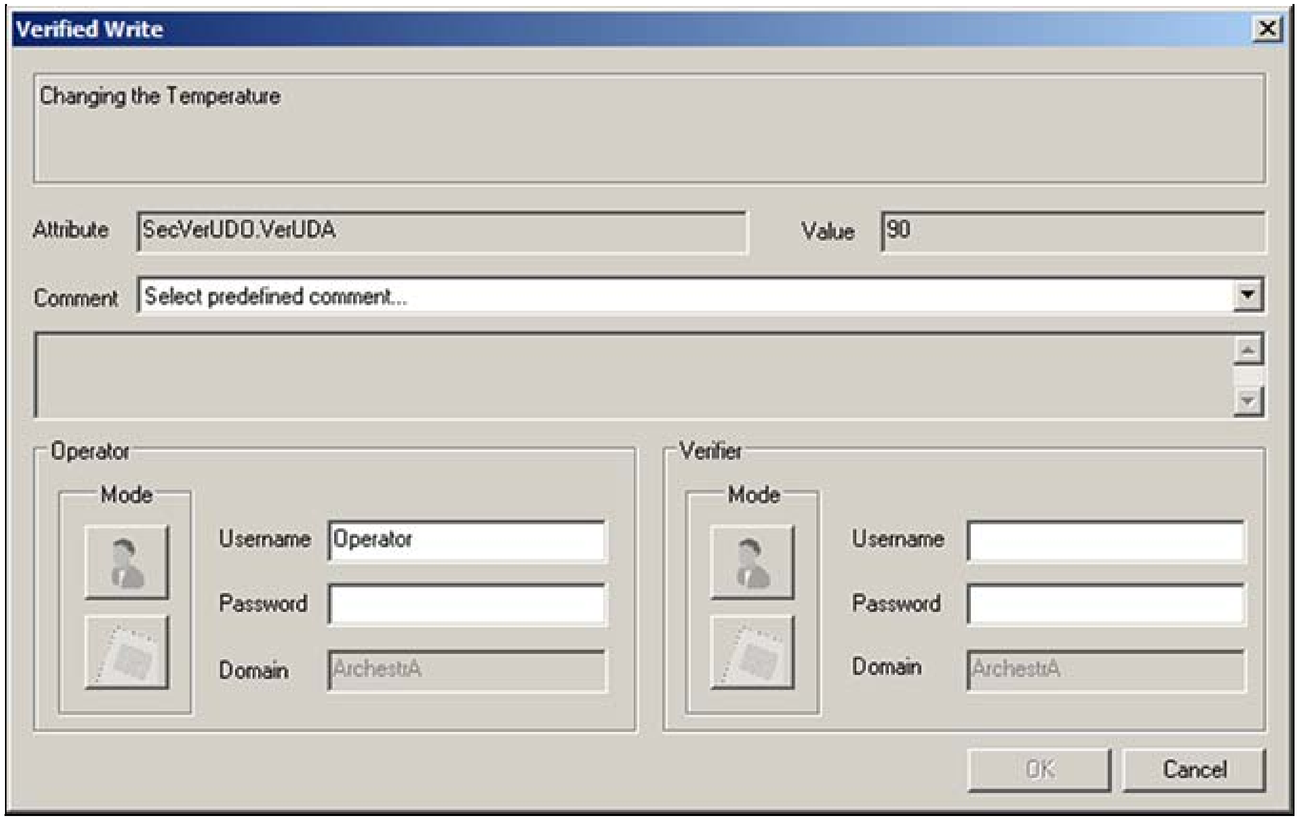


Figure 32: Verified Write Dialog Box

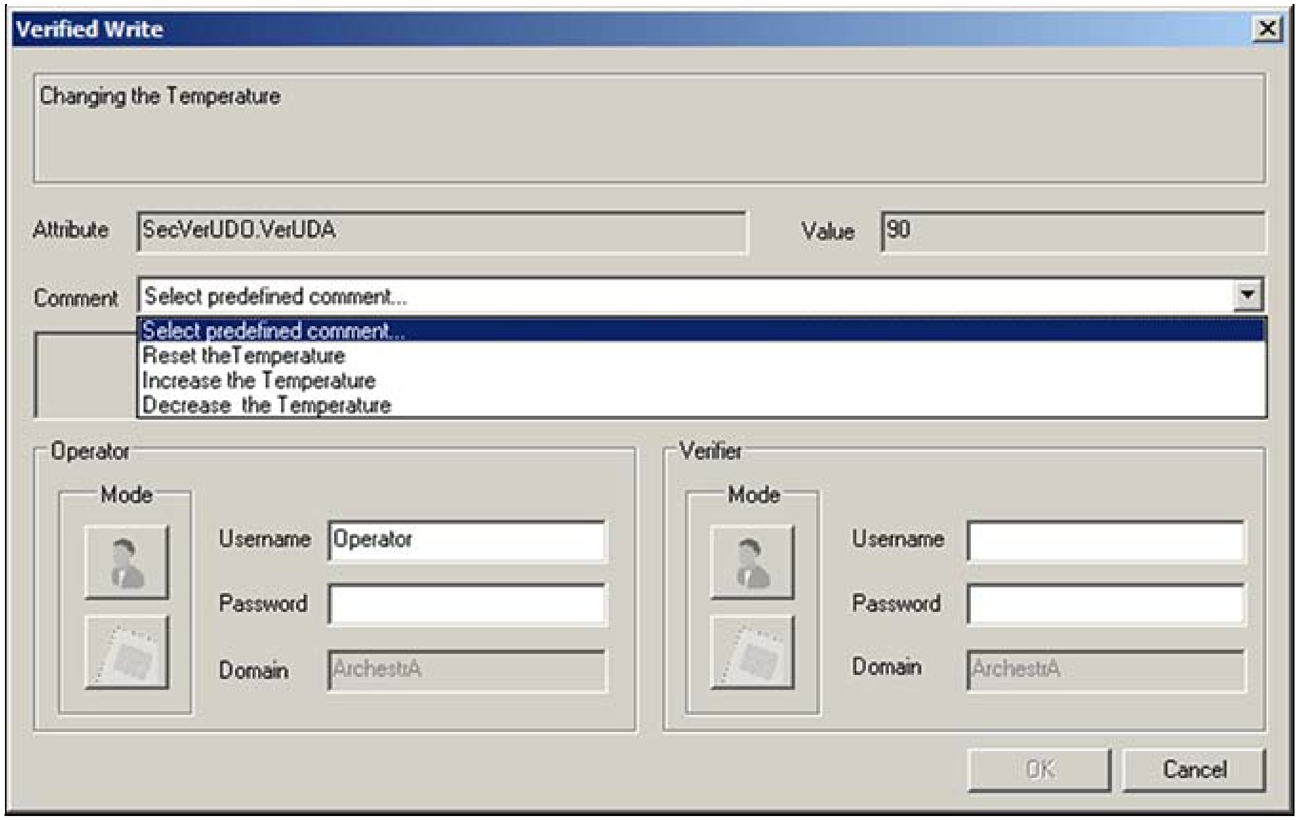
Notice that the Verified Write dialog also asks the Operator to sign, as well as the Verifier. This is because the **VerUDA** had **Verified Write** security classification.

For this example, the user name and password:

Operator:
UserName : Operator
Password: operator

Verifier:
UserName: Supervisor
Password: supervisor

Notice there is Pre-defined comment list as we set it in the script (Figure 33 below).


Verified Write

Changing the Temperature

Attribute: SecVerUDO.VerUDA

Value: 90

Comment: Select predefined comment...

- Select predefined comment...
- Reset theTemperature
- Increase the Temperature
- Decrease the Temperature

Operator

Username: Operator

Password:

Domain: ArchestrA

Verifier

Username:

Password:

Domain: ArchestrA

FIGURE 33: PREDEFINED COMMENTS FROM UDA ARRAY

22. Since the Operator had permissions to modify the value changed and now in WindowViewer you notice the change (Figure 34 below).

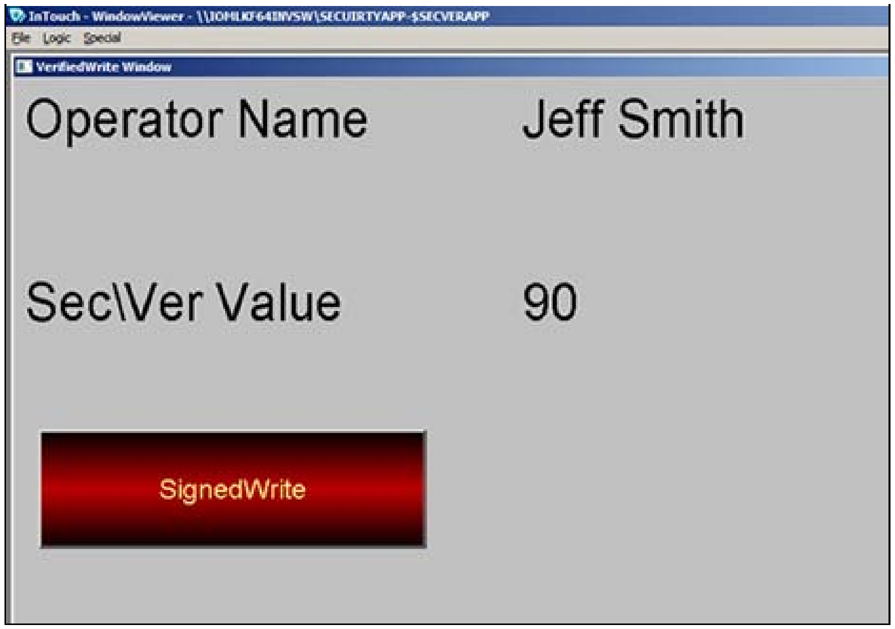


FIGURE 34: VERIFIED WRITE SUCCESSFUL

Return values indicate success or failure status.

- A non-zero value indicates type of failure.
- For more help on the different return values, refer to the Scripting.pdf/page 62 for Wonderware Application Server 3.5

Note: A return value of **0** does not indicate whether the attribute was updated, only that the function placed an entry on the queue to write to the attribute. The operator may decide to cancel the operation after the Secured Write or Verified write dialog box is displayed.

In this case the attribute is not updated and a message is placed in the Logger indicating that the user canceled the operation. Even if the user enters valid credentials and clicks OK, the attribute still might not have been updated because of inadequate permission or data coercion problems.

SignedWrite() Scripting Tips

Using Bound References in SignedWrite()

If the Attribute parameter string evaluates to the name of a Custom Property and that Custom Property is a bound reference to an Attribute, the SignedWrite() function will write to that indicated Attribute.

The Attribute must have the security classification of **Secured Write** or **Verified Write**.

- The SignedWrite() function supports Custom Properties that are nested bound references. That is, if the string evaluates to the name of a Custom Property and that Custom Property is a bound reference to another Custom Property (which itself is a bound reference), the SignedWrite() function will follow through the chain of bound references until it finds an item that is a value. If that item is an Attribute that has the security classification of Secured Write or Verified Write, the SignedWrite() function will write to that item.

Using SignedWrite() in WhileTrue, WhileFalse, or Periodic Type Scripts

Using the SignedWrite() function with WhileTrue, WhileFalse, or Periodic type scripts can repeatedly execute the script, causing another secured write dialog box to pop up with each trigger. We do not recommend using the SignedWrite() function with WhileTrue, WhileFalse, or Periodic types.

Using SignedWrite() with OnShow and OnHide Scripts

We do not recommend using the SignedWrite() function with OnShow and OnHide scripts. This can cause issues with window functionality, including the window title bar, windows losing correct focus, and windows opening on top of one another.

Note: The SignedWrite() function is supported only for client scripting and not for object scripting.

B. Shah

Back to top