White Paper

ThinManager Events

Contents

Introduction 3

Events Overview 3

System Requirements 3

Preparation 4

Update TermMon ActiveX Control on Client Servers 4

Configure ThinManager Server Permissions 4

Application 6

Event Use Cases 6

How Events Work 7

Creating Events 8

ThinManager Event Actions 9

Event Creation on the ThinManager Server 12

Event Trigger on the HMI Application 13

VBA Code Examples for FactoryTalk View SE 14

Enable, Disable, and View Event Log for Events 23

ThinManager REST API & Events 24

E Signature Events 30

Create an E Signature Event 31

VBA Code for E Signature Events 32

Introduction

Events Overview

ThinManager software has always supported delivering content to devices, users, and locations. Now, as of ThinManager software version 13.0, content can be delivered as Events. A ThinManager Event can trigger an action on a terminal. This paper will describe some recommended applications of Events and examples of how to configure and code them using VBA and the TermMon ActiveX control in FactoryTalk View SE. The examples given here can also be applied to other HMI applications that support VBA.

System Requirements

- ThinManager Software version 13.00.00 or newer
- TermMon ActiveX Control version 13.0.0 or newer

Preparation

Update TermMon ActiveX Control on Client Servers

Download the latest version of the TermMon ActiveX Control. It is available for download on the website https://downloads.thinmanager.com. After you download the TermMon ActiveX Control, you will need to copy the file onto the computer hosting your client and register the ActiveX Control. Follow the process found in technote QA45262, Updating the TermMon ActiveX in FactoryTalk View SE (custhelp.com). These steps can be adapted for other HMI applications, as the ActiveX Control need only be copied to a directory where the HMI application has Windows OS permissions to access it. The REGSVR32 command will point to the appropriate ActiveX Control location on the computer's drive.

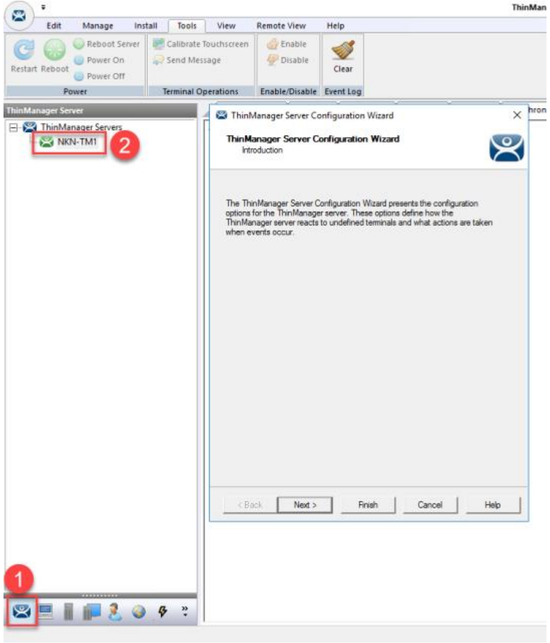
The TermMon ActiveX control should be updated on every machine that:

- Hosts the HMI client software
  - Example: FactoryTalk View SE Client
- Hosts the development environment for your HMI software
  - Example: FactoryTalk View Studio

Configure ThinManager Server Permissions

To use Events in ThinManager, you must give yourself permission to add and edit Events in the ThinManager server.

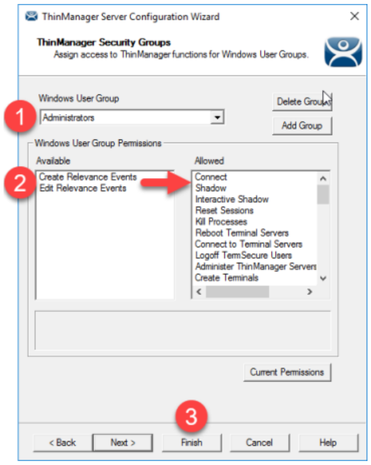
1. Open the ThinManager Admin Console. Click on the **ThinManager** icon in the bottom left corner. Double-click on the ThinManager **server name** in the left ThinManager Servers explorer to bring up the ThinManager Server Configuration Wizard.



ThinManager Server Configuration Wizard
Introduction

The ThinManager Server Configuration Wizard presents the configuration options for the ThinManager server. These options define how the ThinManager server reacts to undefined terminals and what actions are taken when events occur.

2. Click the **Next** button until you get to the ThinManager Security Groups page of the ThinManager Server Configuration Wizard. Select the desired Windows group (in this case it is Administrators). Double-click both the **Create Relevance Events** and the **Edit Relevance Events** in the left Available list to add them to the right Allowed list. Click the **Finish** button.



ThinManager Server Configuration Wizard

ThinManager Security Groups

Assign access to ThinManager functions for Windows User Groups.

Windows User Group: Administrators

Windows User Group Permissions

Available:
Create Relevance Events
Edit Relevance Events

Allowed:
Connect
Shadow
Interactive Shadow
Reset Sessions
Kill Processes
Reboot Terminal Servers
Connect to Terminal Servers
Logoff TermSecure Users
Administer ThinManager Servers
Create Terminals

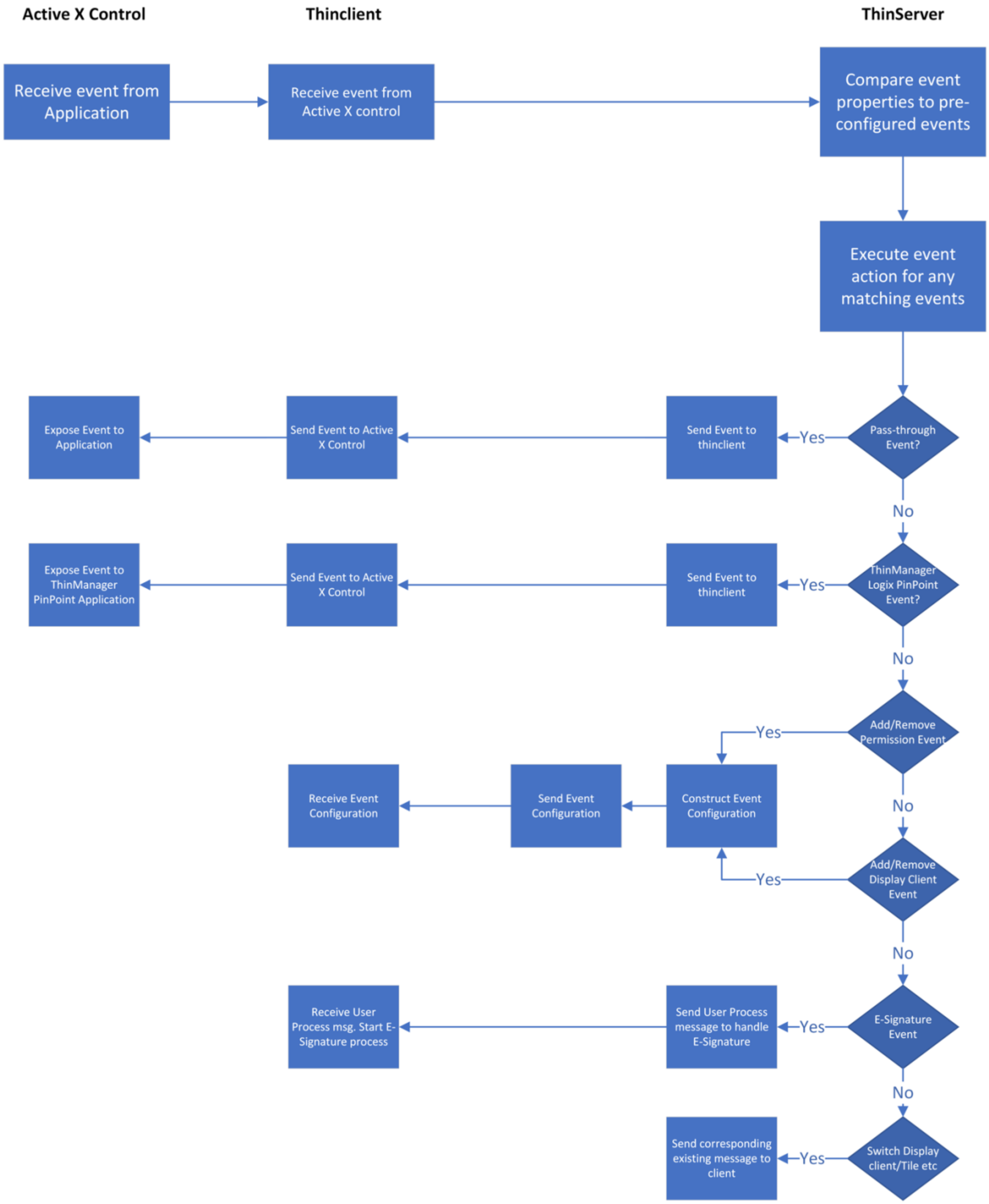
Application

Event Use Cases

- Open a camera Display Client whenever a security perimeter is breached (a tag value changes on the HMI application)
- Open an alarm Display Client on your HMI whenever there is certain severity of alarms present (a tag value changes on the HMI application)
- Open a Display Client with the click of a button
- Tile applications on a terminal with the press of an HMI button
- Pass an event from one specific terminal to another using the ThinManager server
- Allow ThinManager to authenticate an E Signature request from FactoryTalk View SE or other HMI software

How Events Work

Below is an example chart of how events work under the hood of ThinManager software:

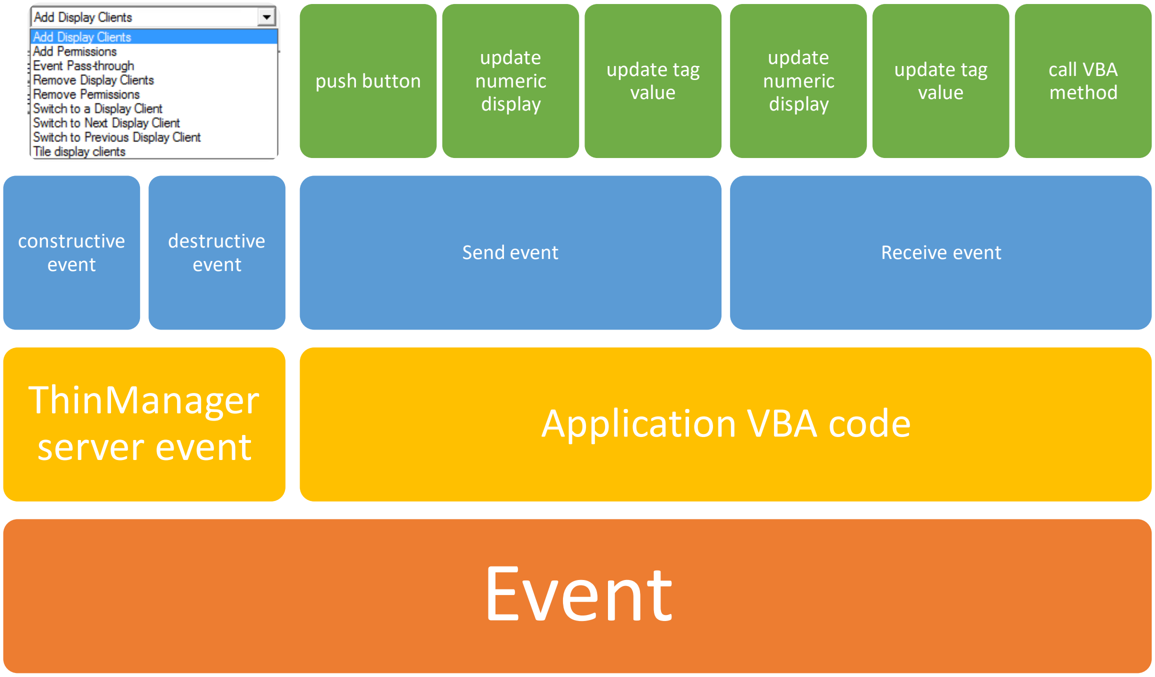

Active X Control
Thinclient
ThinServer
Receive event from Application
Receive event from Active X control
Compare event properties to pre-configured events
Execute event action for any matching events
Pass-through Event?
Yes
Send Event to thinclient
Send Event to Active X Control
Expose Event to Application
No
ThinManager Logix PinPoint Event?
Yes
Send Event to thinclient
Send Event to Active X Control
Expose Event to ThinManager PinPoint Application
No
Add/Remove Permission Event
Yes
Construct Event Configuration
Send Event Configuration
Receive Event Configuration
No
Add/Remove Display Client Event
Yes
No
E-Signature Event
Yes
Send User Process message to handle E-Signature
Receive User Process msg. Start E-Signature process
No
Switch Display client/Tile etc
Yes
Send corresponding existing message to client

Creating Events

Event creation with ThinManager can be broken into two different tasks:

1. HMI application design and VBA programming
2. Creation of the event(s) on the ThinManager server

Here is an overview of the components and options with ThinManager event from the HMI and the ThinManager server perspective:



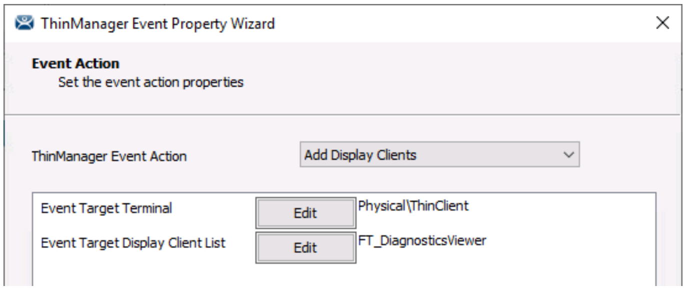
You can mix and match different types of HMI application interactions with different kinds of events, creating many exciting combinations and possibilities.

Note: There are constructive and destructive Event Actions. A constructive Event adds something to the terminal or configures something. A destructive Event takes away what was added. For example, you may add a display client using the Add Display Clients Event action. You could then use the complement of that function in another Event, Remove Display Clients, to remove the Display Client from the terminal. You cannot remove a Display Client that was previously assigned to a terminal profile. Similarly, you can only remove permissions from a terminal that were added by an Event.

ThinManager Event Actions

The following are available Event Actions with their associated parameters in ThinManager software. To edit a parameter, click the **Edit** button for each respectively:

- **Add Display Client**: Add a specified Display Client to a specified terminal
    - **Event Target Terminal**: Specify which terminal to add the Display Client to. The **To source terminal** option in the **Select target Terminal** window will add the Display Client on the terminal that sends the Event.
    - **Event Target Display Client List**: Add a specific Display Client to the terminal.

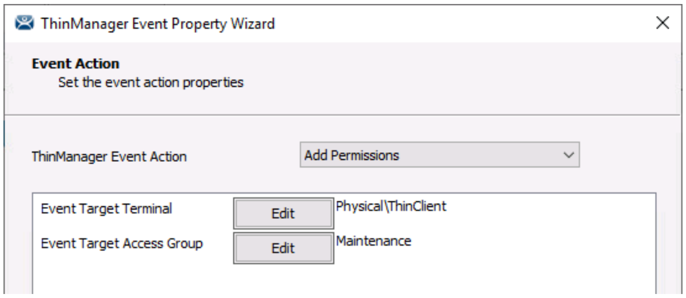

ThinManager Event Property Wizard
Event Action
Set the event action properties

ThinManager Event Action: Add Display Clients

Event Target Terminal | Edit | Physical\ThinClient

Event Target Display Client List | Edit | FT_DiagnosticsViewer

- **Add Permissions**: Add an Access Group to the permissions list of the terminal.
    - **Event Target Terminal**: Specify which terminal to add the Access Group to. The **To source terminal** option in the **Select target Terminal** window will assign the Access Group on the terminal that sends the Event.
    - **Event Target Access Group**: The specific Access Group you want to add to the terminal.

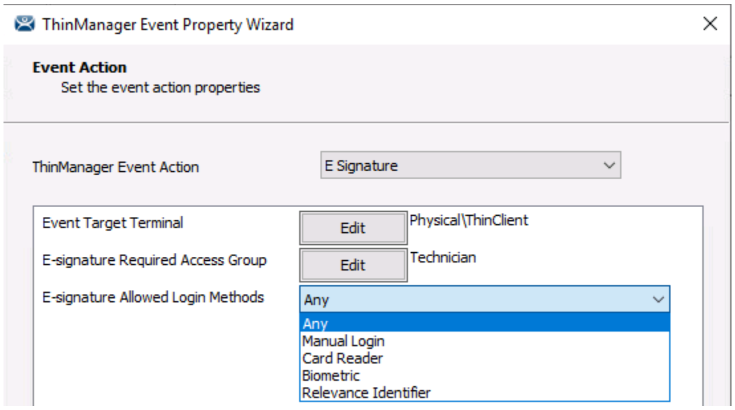

ThinManager Event Property Wizard
Event Action
Set the event action properties

ThinManager Event Action: Add Permissions

Event Target Terminal | Edit | Physical\ThinClient

Event Target Access Group | Edit | Maintenance

- **E Signature**: Trigger an E Signature action on the terminal from FactoryTalk View SE. This is used to sign off on actions and processes on the HMI application. This integrates directly with the ThinManager Users and their associated authentication mechanisms for login such as passwords, PINs, badge readers, and fingerprint readers.
  - **Event Target Terminal**: Specify which terminal to send confirmation of a valid ThinManager User login to. The **To source terminal** option in the **Select target Terminal** window will pass the login information to the terminal that sends the Event.
  - **E Signature Required Access Group**: The Access Group that is allowed to log in and validate the E Signature Event on the terminal.
  - **E Signature Allowed Login Methods**: Restrict which methods are allowed at the terminal to authenticate an E Signature request. Any allows all valid login methods for the ThinManager user logging in. Specifying a specific login method (example: Card Reader) restricts the login to only that allowed method.



ThinManager Event Property Wizard

Event Action

Set the event action properties

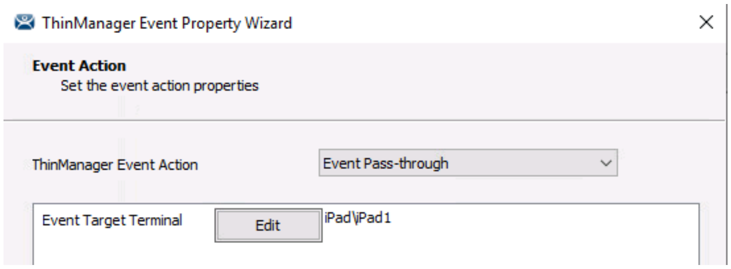
ThinManager Event Action: E Signature

Event Target Terminal: Physical\ThinClient

E-signature Required Access Group: Technician

E-signature Allowed Login Methods: Any

- **Event Pass-through**: Passes through an Event from one ThinManager terminal to another. This could be useful to trigger an action from one application to another using the ThinManager TermMon ActiveX control.
  - **Event Target Terminal**: Specify which terminal to send confirmation of a valid ThinManager User login to. The **To source terminal** option in the **Select target Terminal** window will pass the login information to the terminal that sends the Event.



ThinManager Event Property Wizard

Event Action

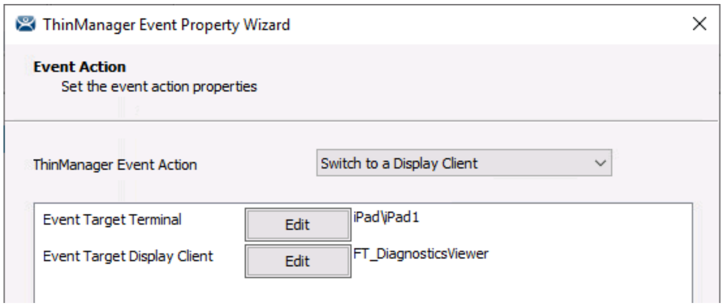
Set the event action properties

ThinManager Event Action: Event Pass-through

Event Target Terminal: iPad\iPad1

- **Remove Display Clients**: This does the same thing and has the same parameters that the **Add Display Clients** action does, but instead of adding a Display Client it removes a Display Client from a specified terminal. You can only remove Display Clients that have been added by a ThinManager Event.
- **Remove Permissions:** This does the same thing and has the same parameters that the **Add Permissions** action does, but instead of adding an Access Group it removes an Access Group from a specified terminal. You can only remove Access Groups that have been added by a ThinManager Event.

- **Switch to a Display Client**: Makes the Specified Display Client on a terminal the Display Client that is in the foreground of the Display. The Display Client must be present on the terminal for this Event to function correctly.
    - **Event Target Terminal**: Specify which terminal to switch the Display Client on. The **To source terminal** option in the **Select target Terminal** window will switch the Display Client on the terminal that sends the Event.
    - **Event Target Display Client List**: The specific Display Client to switch on the terminal.



ThinManager Event Property Wizard

Event Action

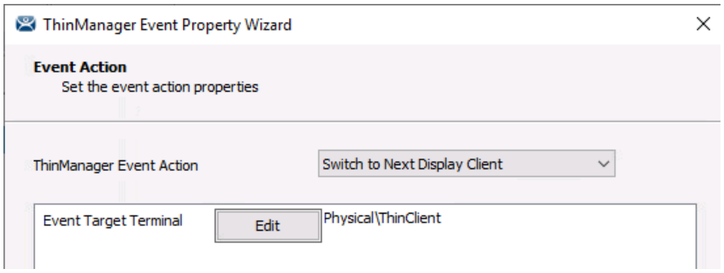
Set the event action properties

ThinManager Event Action: Switch to a Display Client

Event Target Terminal: iPad\iPad1

Event Target Display Client: FT_DiagnosticsViewer

- **Switch to Next Display Client**: Switch to the next Display Client on the Display Client list on a terminal. Note that you must have more than one Display Client on that terminal for this Event to work.
    - **Event Target Terminal**: Specify which terminal to switch to the Next Display Client. The **To source terminal** option in the **Select target Terminal** window will switch to the next Display Client on the terminal that sends the Event.



ThinManager Event Property Wizard

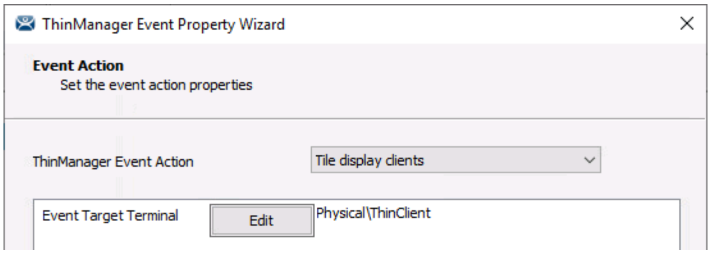
Event Action

Set the event action properties

ThinManager Event Action: Switch to Next Display Client

Event Target Terminal: Physical\ThinClient

- **Switch to Previous Display Client**: This does the same thing as the **Switch to Next Display Client** Event action but goes to the previous Display Client instead of the next Display Client on the terminal's Display Client list. Note that you must have more than one Display Client on that terminal for this Event to work.
- **Tile display clients**: Tiles the Display Clients on a specified terminal in a grid format.
    - **Event Target Terminal**: Specify which terminal to tile the Display Client on. The **To source terminal** option in the **Select target Terminal** window will tile the Display Client on the terminal that sends the Event.



ThinManager Event Property Wizard

Event Action

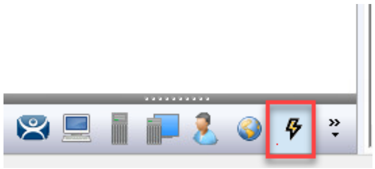
Set the event action properties

ThinManager Event Action: Tile display clients

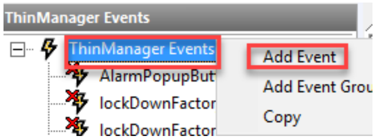
Event Target Terminal: Physical\ThinClient

Event Creation on the ThinManager Server

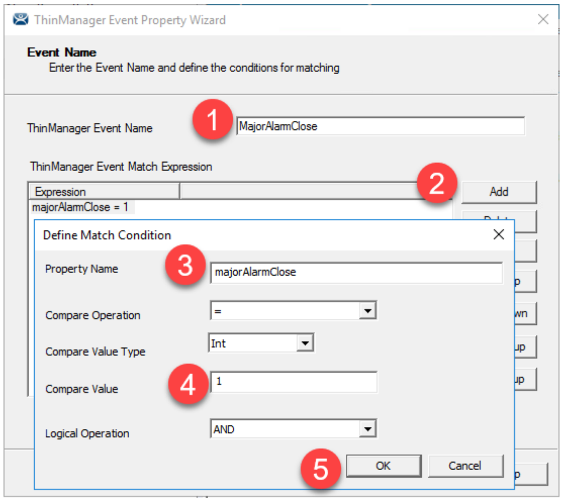
1. Open the ThinManager Server and click on the **Events** icon in the lower left-hand corner. You may need to expand the navigation bar if you do not see the Events icon.



2. Right-click ThinManager Events and click **Add Event** to bring up the ThinManager Event Property Wizard.



3. Name the Event, then click the **Add** button to define an Event Match Expression. Name the expression and put in a Compare Value and click **OK**.


ThinManager Event Property Wizard

Event Name
Enter the Event Name and define the conditions for matching

1 ThinManager Event Name: MajorAlarmClose

ThinManager Event Match Expression

Expression
majorAlarmClose = 1

Define Match Condition

3 Property Name: majorAlarmClose

Compare Operation: =

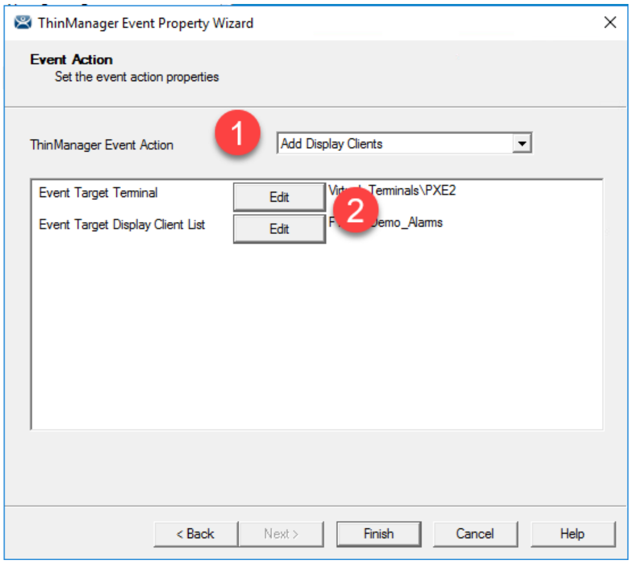
Compare Value Type: Int

4 Compare Value: 1

Logical Operation: AND

Note: Leaving the Compare Operation, Compare Value Type, and Logical Operation as defaults is desirable to keep things simple. However, if you want to add complexity to the expression by introducing different conditions and logic when an Event is triggered.

4. Click the **Next** button on the Event Wizard to get to the Event Action page.
5. Select the desired ThinManager Event Action in the drop-down menu. Configure the desired options by clicking the **Edit** button for each respective option.

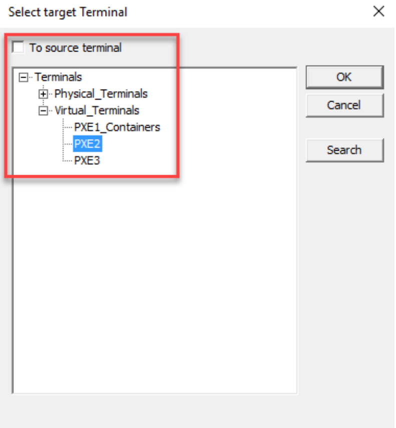

ThinManager Event Property Wizard
Event Action
Set the event action properties
1
2
Edit
Edit
< Back
Next >
Finish
Cancel
Help

ThinManager Event Action: Add Display Clients

Event Target Terminal: Vir... Terminals\PXE2

Event Target Display Client List: F... Demo_Alarms

a. For example, if you want the Event Action to be Add a Display Client, you will choose an Event Destination Terminal and a Display Client. In the Select target Terminal window, the **To source terminal** checkbox will direct any Event from any terminal back at itself. This would be desired if you want the Event to trigger on multiple terminals without having to create multiple Events.



6. Click the **Finish** button to exit the ThinManager Event Property Wizard.

The Event is now active and listening for any incoming events to be sent from the HMI side.

Event Trigger on the HMI Application

There are several ways to trigger a ThinManager Event in VBA code from the HMI perspective. Here are some principles from a FactoryTalk View SE design perspective that should be kept in mind:

- VBA code is tied to each display. Each display's VBA code is isolated from the other display's VBA code.
- You can create one Display that is always running in the background, hidden using the Display [DisplayName] /ZA command. See FactoryTalk View SE and RSView32: Display background always updating (custhelp.com) for more details.

- You can assign VBA code to different VBA events, such as a push button or the change of a variable in a numeric display.
- You must instantiate an object of TermMon ActiveX control and expose it to VBA control on each display that will have VBA code processing ThinManager Events.

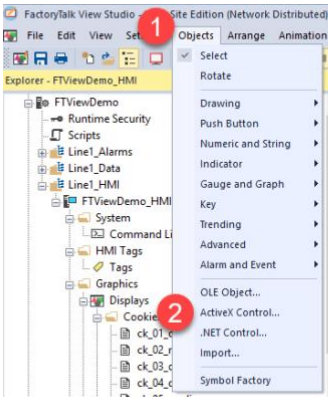
VBA Code Examples for FactoryTalk View SE

Add the TermMon ActiveX to a FactoryTalk View SE Display

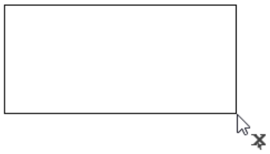
If you have not already, follow the steps in the Update TermMon ActiveX Control on Client Servers section of this document to update the TermMon ActiveX control to the latest version that supports ThinManager Events.

The following is an example of adding the TermMon ActiveX control to a FactoryTalk View SE Display and then adding the supporting VBA code to activate the TermMon ActiveX control.

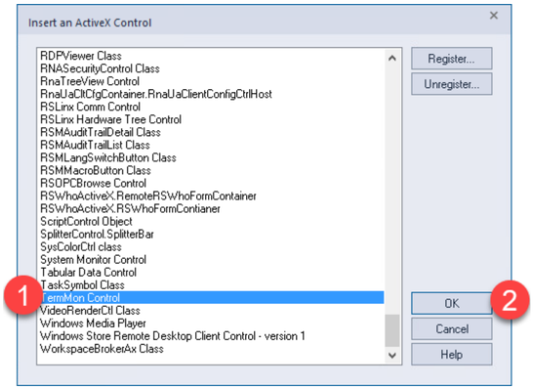
1. Open FactoryTalk View Studio.
2. Select your application and open it.
3. Open an existing Display you want to add the TermMon ActiveX control to (or create a new Display).
4. Go to the **Objects** menu and click on **ActiveX Control**.



5. Drag your mouse cursor across the display to draw a new ActiveX control on the screen.

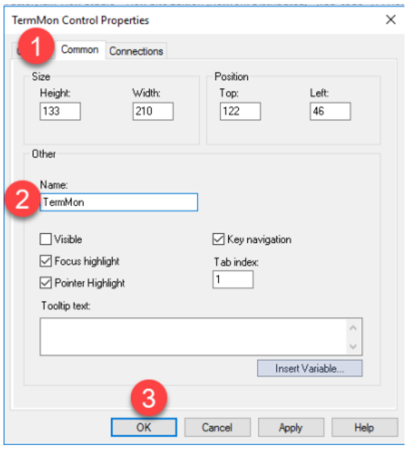


6. When the Insert an ActiveX Control window pops up, scroll down to **TermMon Control**. Highlight it and click **OK**.


Insert an ActiveX Control

RDPViewer Class
RNASecurityControl Class
RnaTreeView Control
RnaUaCltCfgContainer.RnaUaClientConfigCtrlHost
RSLinx Comm Control
RSLinx Hardware Tree Control
RSMAuditTrailDetail Class
RSMAuditTrailList Class
RSMLangSwitchButton Class
RSMMacroButton Class
RSOPCBrowse Control
RSWhoActiveX.RemoteRSWhoFormContainer
RSWhoActiveX.RSWhoFormContianer
ScriptControl Object
SplitterControl.SplitterBar
SysColorCtrl class
System Monitor Control
Tabular Data Control
TaskSymbol Class
TermMon Control
VideoRenderCtl Class
Windows Media Player
Windows Store Remote Desktop Client Control - version 1
WorkspaceBrokerAx Class

7. Click on the **Common** tab of the TermMon Control Properties window that pops up. Name the control whatever you like in the Name field. Click **OK** to close the window.


TermMon Control Properties

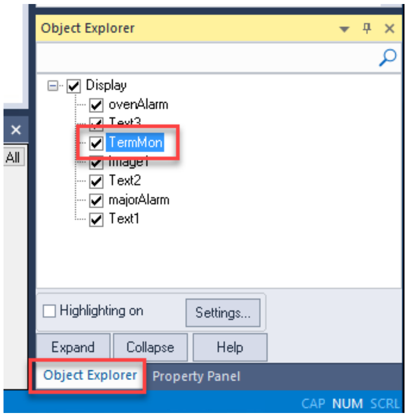
Size
Height: 133
Width: 210

Position
Top: 122
Left: 46

Other
Name: TermMon

☐ Visible
☑ Focus highlight
☑ Pointer Highlight
☑ Key navigation
Tab index: 1
Tooltip text:

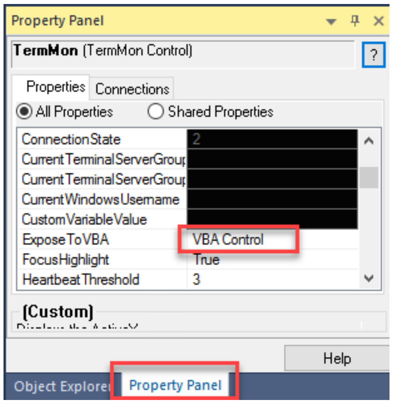
8. Click on the newly drawn TermMon ActiveX control (it may be hard to find as it may be invisible). You may use the Object Explorer to highlight it instead of finding it on the display.


Object Explorer

☑ Display
☑ ovenAlarm
☑ Text3
☑ TermMon
☑ Image1
☑ Text2
☑ majorAlarm
☑ Text1

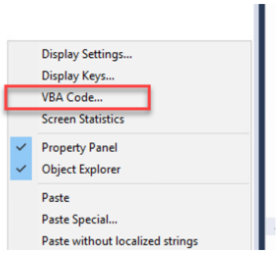
☐ Highlighting on

9. Click the **Property Panel** tab at the lower right-hand corner of the screen or open from the View → Property Panel menu at the top of the screen. Scroll down to the ExposeToVBA field and change the value to **VBA Control**.

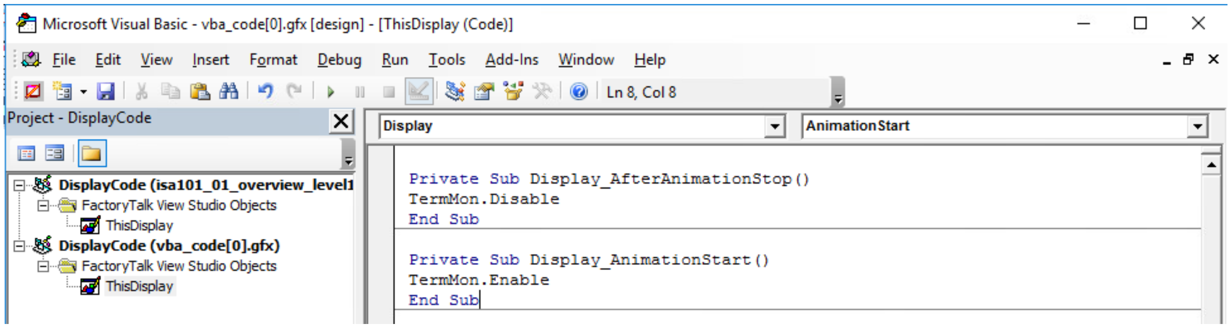


ConnectionState	[redacted]
CurrentTerminalServerGroup	[redacted]
CurrentTerminalServerGroup	[redacted]
CurrentWindowsUsername	[redacted]
CustomVariableValue	[redacted]
ExposeToVBA	VBA Control
FocusHighlight	True
HeartbeatThreshold	3

10. Right-click on the display and select **VBA Code** to pop up the Microsoft Visual Basic editor.



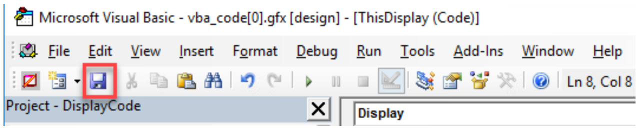
11. Add the following code to enable and disable the TermMon ActiveX control on the display being opened and closed respectively:


Microsoft Visual Basic - vba_code[0].gfx [design] - [ThisDisplay (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 8, Col 8
Project - DisplayCode
DisplayCode (isa101_01_overview_level1
FactoryTalk View Studio Objects
ThisDisplay
DisplayCode (vba_code[0].gfx)
FactoryTalk View Studio Objects
ThisDisplay
Display
AnimationStart

```
Private Sub Display_AfterAnimationStop()
TermMon.Disable
End Sub

Private Sub Display_AnimationStart()
TermMon.Enable
End Sub
```

```
Private Sub Display_AfterAnimationStop()
TermMon.Disable
End Sub

Private Sub Display_AnimationStart()
TermMon.Enable
End Sub
```

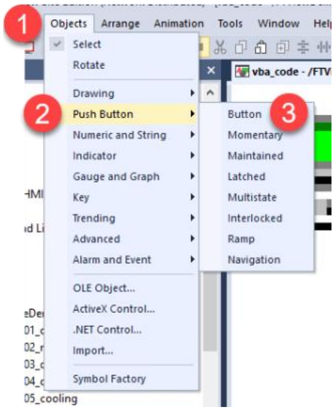
12. Click the **Save** button at the top of the screen to save the VBA code.


Microsoft Visual Basic - vba_code[0].gfx [design] - [ThisDisplay (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 8, Col 8
Project - DisplayCode
Display

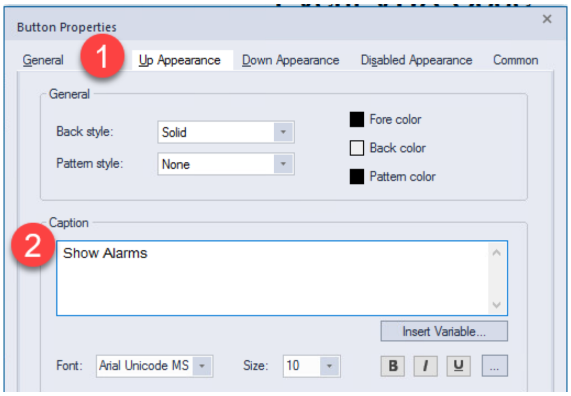
Button Trigger Example

Create a button that will trigger an Event whenever it is pressed on the screen.

1. Create a new button object by going to the Object menu at the top of FactoryTalk View Studio and selecting Button → Push Button. Drag your cursor across the display to draw the button object.



2. The Button Properties window will pop up. Click on the **Up Appearance** tab and type in the text you want your button to have in the **Caption** field.



General

Back style: Solid

Pattern style: None

■ Fore color

□ Back color

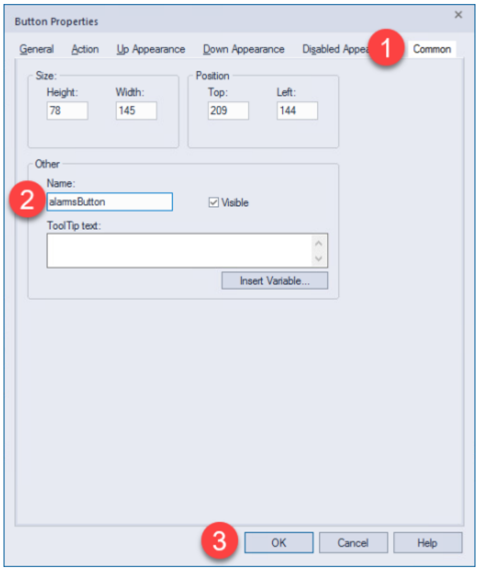
■ Pattern color

Caption

Show Alarms

Font: Arial Unicode MS   Size: 10

3. Click the **Common** tab and type in the desired name you want the button to have in the VBA code in the **Name** field. Click the **OK** button.



Size:

Height: 78   Width: 145

Position

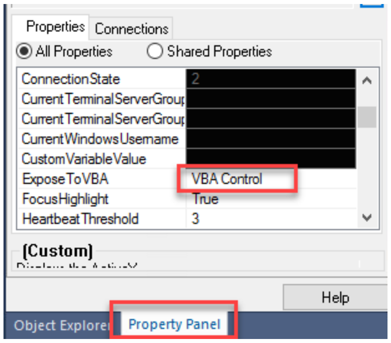
Top: 209   Left: 144

Other

Name: alarmsButton   ☑ Visible

ToolTip text:

4. Click the new button. Click the **Property Panel** tab at the lower right-hand corner of the screen or open from the View → Property Panel menu at the top of the screen. Scroll down to the ExposeToVBA field and change the value to **VBA Control**.

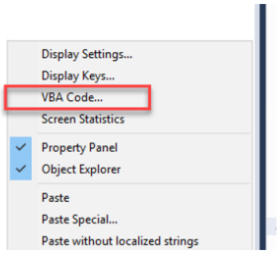


● All Properties   ○ Shared Properties

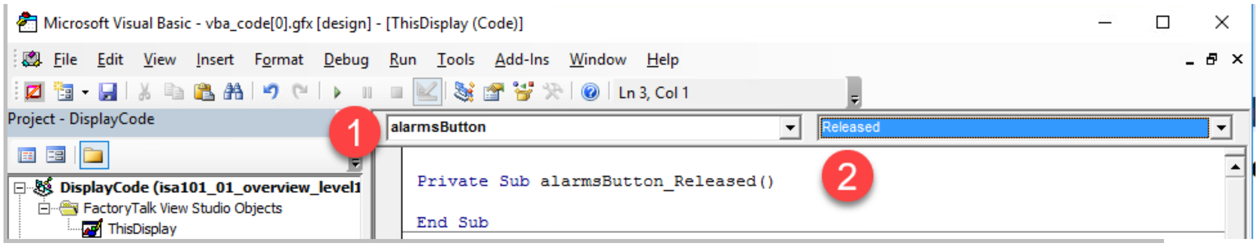
ConnectionState	2
CurrentTerminalServerGrou	
CurrentTerminalServerGrou	
CurrentWindowsUsername	
CustomVariableValue	
ExposeToVBA	VBA Control
FocusHighlight	True
HeartbeatThreshold	3

(Custom)

5. Right-click on the display and select **VBA Code** to pop up the Microsoft Visual Basic editor.



6. Using the two drop-downs, select the name of the button and the Released event. This should generate a new blank routine to fill in with code.


Microsoft Visual Basic - vba_code[0].gfx [design] - [ThisDisplay (Code)]
File
Edit
View
Insert
Format
Debug
Run
Tools
Add-Ins
Window
Help
Ln 3, Col 1
Project - DisplayCode
DisplayCode (isa101_01_overview_level1
FactoryTalk View Studio Objects
ThisDisplay
1
2

alarmsButton

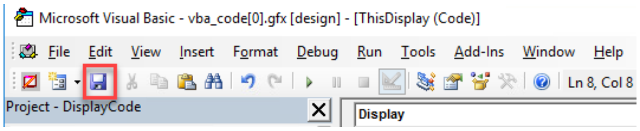
Released

```
Private Sub alarmsButton_Released()

End Sub
```

7. Add the following code, taking care to replace the eventName string with the match expression you created for your Event in the ThinManager Admin Console. See the Event Creation on the ThinManager Server section of this white paper.

```
'Create a string variable and put the Event Match Expression in
it
Dim eventName As String
eventName = "majorAlarmExists=1"

'Send the Event to ThinManager
TermMon.SendGenericEvent (eventName)
```

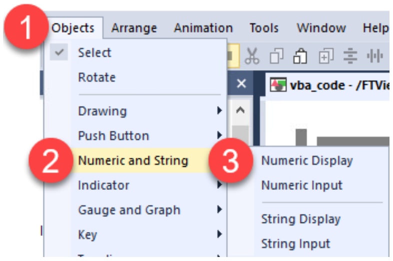
8. Click the **Save** button at the top of the screen to save the VBA code.

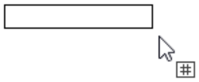


Numeric Display with Tag Value Trigger Example

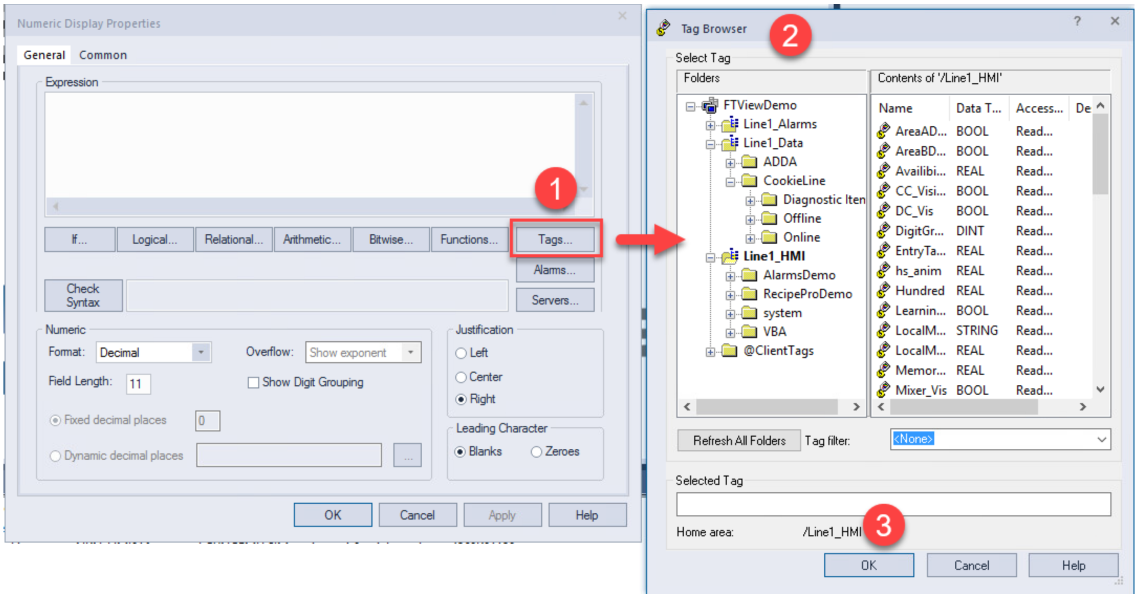
Create a Numeric Display that will trigger an Event whenever the tag matches a certain integer value.

1. Create a new numeric display object by going to the Object menu at the top of FactoryTalk View Studio and selecting Numeric and String → Numeric Display. Drag your cursor across the display to draw the numeric display object.





2. The Numeric Display Properties window will pop up. In the General tab, you can assign a tag value to the Numeric Display by pressing the **Tags…** button and browsing to a tag in your HMI server tag database or on a PLC. Browse to the tag and then click the OK button on both windows to close. You may also choose to use any valid functions or logical operators available to you. For example, in the second screenshot below, we use a combination of two alarm functions to determine if there are high-priority or urgent alarms.



Format: Decimal
Overflow: Show exponent
Field Length: 11
□ Show Digit Grouping
◉ Fixed decimal places 0
○ Dynamic decimal places
Justification: ○ Left ○ Center ◉ Right
Leading Character: ◉ Blanks ○ Zeroes

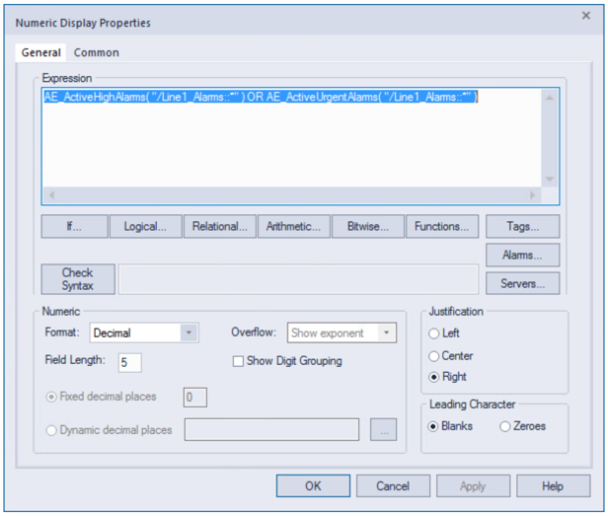
FTViewDemo
- Line1_Alarms
- Line1_Data
  - ADDA
  - CookieLine
    - Diagnostic Iten
    - Offline
    - Online
- **Line1_HMI**
  - AlarmsDemo
  - RecipeProDemo
  - system
  - VBA
- @ClientTags

Contents of '/Line1_HMI'

Name	Data T...	Access...	De
AreaAD...	BOOL	Read...	
AreaBD...	BOOL	Read...	
Availibi...	REAL	Read...	
CC_Visi...	BOOL	Read...	
DC_Vis	BOOL	Read...	
DigitGr...	DINT	Read...	
EntryTa...	REAL	Read...	
hs_anim	REAL	Read...	
Hundred	REAL	Read...	
Learnin...	BOOL	Read...	
LocalM...	STRING	Read...	
LocalM...	REAL	Read...	
Memor...	REAL	Read...	
Mixer_Vis	BOOL	Read...	

Tag filter: <None>

Home area: /Line1_HMI



Numeric Display Properties

General | Common

Expression

```
AE_ActiveHighAlarms( "/Line1_Alarms::*" ) OR AE_ActiveUrgentAlarms( "/Line1_Alarms::*" )
```

Numeric

Format: Decimal

Overflow: Show exponent

Field Length: 5

□ Show Digit Grouping

● Fixed decimal places 0

○ Dynamic decimal places

Justification

○ Left

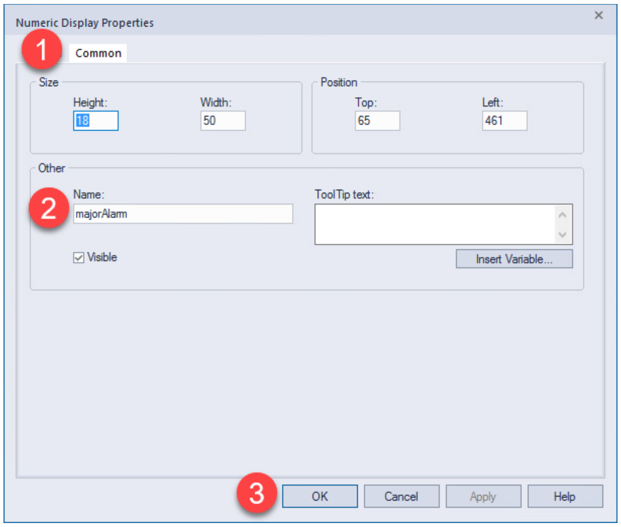
○ Center

● Right

Leading Character

● Blanks ○ Zeroes

3. Click the **Common** tab and type in the desired name you want the numeric display to have in VBA code in the **Name** field. Click the **OK** button.



Numeric Display Properties

Common

Size

Height: 18

Width: 50

Position

Top: 65

Left: 461

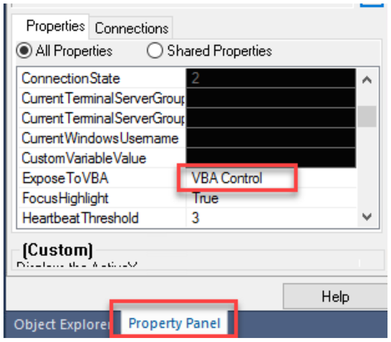
Other

Name: majorAlarm

ToolTip text:

☑ Visible

4. Click the new numeric display object. Click the **Property Panel** tab at the lower right-hand corner of the screen or open from the View → Property Panel menu at the top of the screen. Scroll down to the ExposeToVBA field and change the value to **VBA Control**.



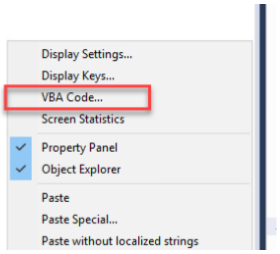
Properties | Connections

● All Properties ○ Shared Properties

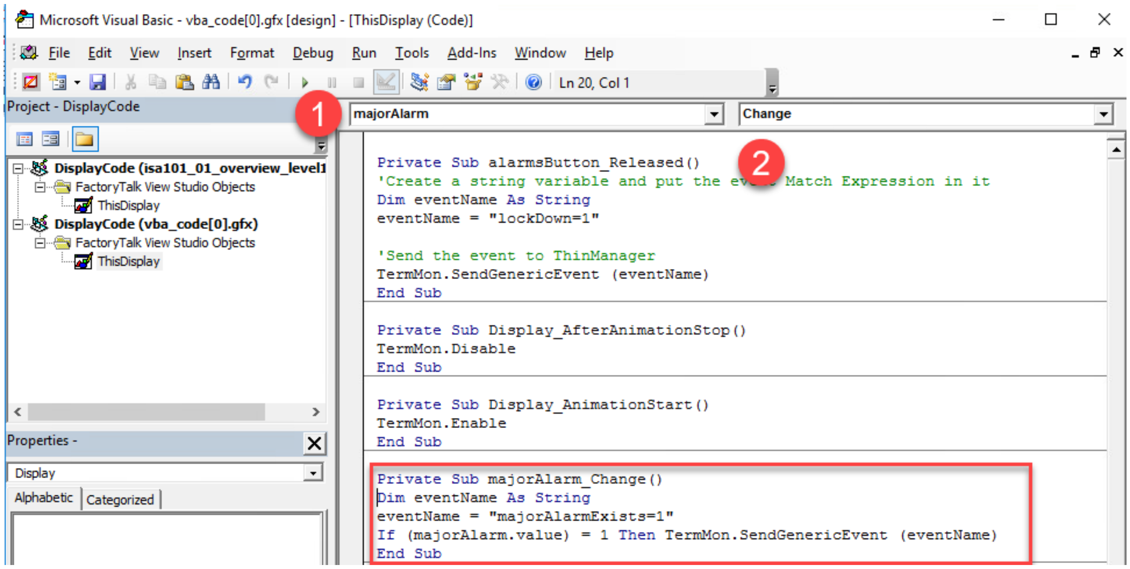
ConnectionState	[redacted]
CurrentTerminalServerGrou	[redacted]
CurrentTerminalServerGrou	[redacted]
CurrentWindowsUsername	[redacted]
CustomVariableValue	[redacted]
ExposeToVBA	VBA Control
FocusHighlight	True
Heartbeat Threshold	3

(Custom)

5. Right-click on the display and select **VBA Code** to pop up the Microsoft Visual Basic editor.



6. Using the two drop-downs, select the name of the numeric display and the Change event. This should generate a new blank routine to fill in with code.

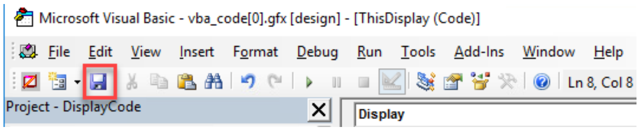

Microsoft Visual Basic - vba_code[0].gfx [design] - [ThisDisplay (Code)]
File
Edit
View
Insert
Format
Debug
Run
Tools
Add-Ins
Window
Help
Ln 20, Col 1
Project - DisplayCode
1
majorAlarm
Change
2
DisplayCode (isa101_01_overview_level1
FactoryTalk View Studio Objects
ThisDisplay
DisplayCode (vba_code[0].gfx)
FactoryTalk View Studio Objects
ThisDisplay
Properties -
Display
Alphabetic
Categorized

```
Private Sub alarmsButton_Released()
'Create a string variable and put the ev     Match Expression in it
Dim eventName As String
eventName = "lockDown=1"

'Send the event to ThinManager
TermMon.SendGenericEvent (eventName)
End Sub

Private Sub Display_AfterAnimationStop()
TermMon.Disable
End Sub

Private Sub Display_AnimationStart()
TermMon.Enable
End Sub

Private Sub majorAlarm_Change()
Dim eventName As String
eventName = "majorAlarmExists=1"
If (majorAlarm.value) = 1 Then TermMon.SendGenericEvent (eventName)
End Sub
```

7. Add the following code, taking care to replace the eventName string with the match expression you created for your Event in the ThinManager Admin Console. See the Event Creation on the ThinManager Server section of this white paper.

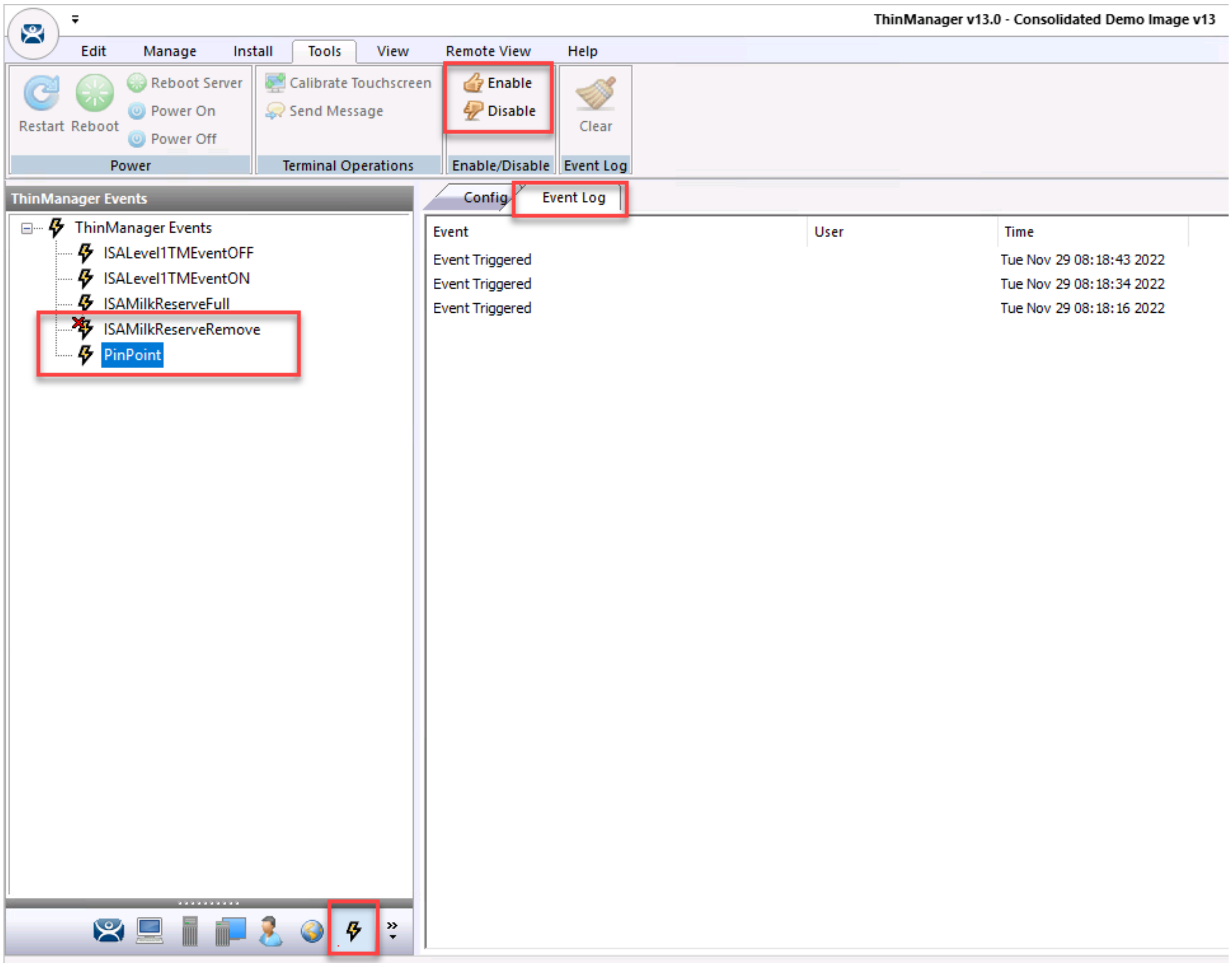
```
Private Sub majorAlarm_Change()
Dim eventName As String
eventName = "majorAlarmExists=1"
If (majorAlarm.value) = 1 Then TermMon.SendGenericEvent
(eventName)
End Sub
```

8. Click the **Save** button at the top of the screen to save the VBA code.



Enable, Disable, and View Event Log for Events

You can enable, disable, and view the Event Log for an Event. Go to the Events section of the ThinManager Admin Console. Highlight any Event. To enable or disable the Event, click on the **Tools** tab, then select **Enable** or **Disable**. Any disabled Events will have an X over the Event icon. To view the Event Log of an Event, highlight the Event and click on the **Event Log** tab. This will show when Events were triggered.

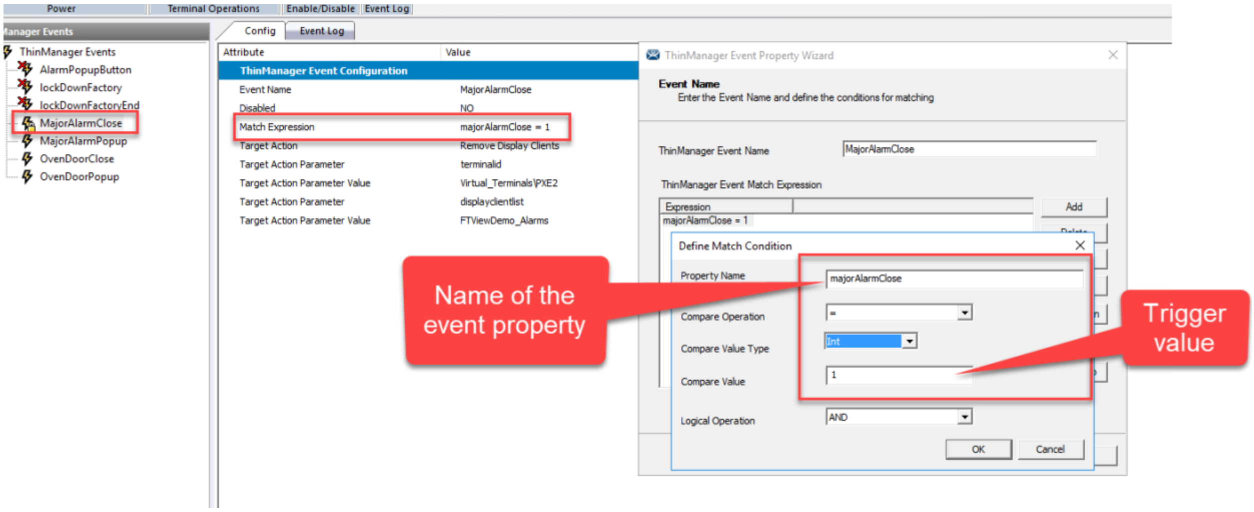


Event	User	Time
Event Triggered		Tue Nov 29 08:18:43 2022
Event Triggered		Tue Nov 29 08:18:34 2022
Event Triggered		Tue Nov 29 08:18:16 2022

ThinManager REST API & Events

You can configure and trigger Events using the new API found in ThinManager version 13 and newer. This is an alternative method to using VBA code in HMI software. Here is a quick guide on how to do so:

1. Take note of the following pieces of information:
    a. The Property Name of the expression used for the Event
    b. The Compare Value used for the Event (trigger value)
2. Write a script to trigger the Event using REST API



Attribute	Value
ThinManager Event Configuration	
Event Name	MajorAlarmClose
Disabled	NO
Match Expression	majorAlarmClose = 1
Target Action	Remove Display Clients
Target Action Parameter	terminalid
Target Action Parameter Value	Virtual_Terminals\PXE2
Target Action Parameter	displayclientlist
Target Action Parameter Value	FTViewDemo_Alarms

ThinManager Event Property Wizard

Event Name
Enter the Event Name and define the conditions for matching

ThinManager Event Name: MajorAlarmClose

ThinManager Event Match Expression

Expression: majorAlarmClose = 1

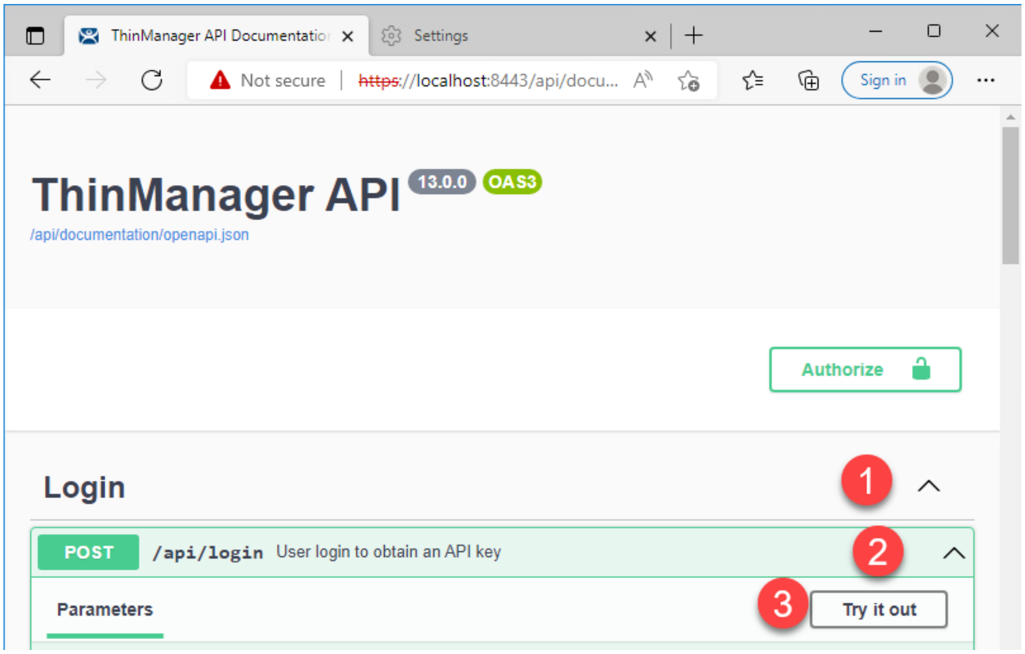
Define Match Condition

Property Name: majorAlarmClose
Compare Operation: =
Compare Value Type: Int
Compare Value: 1
Logical Operation: AND

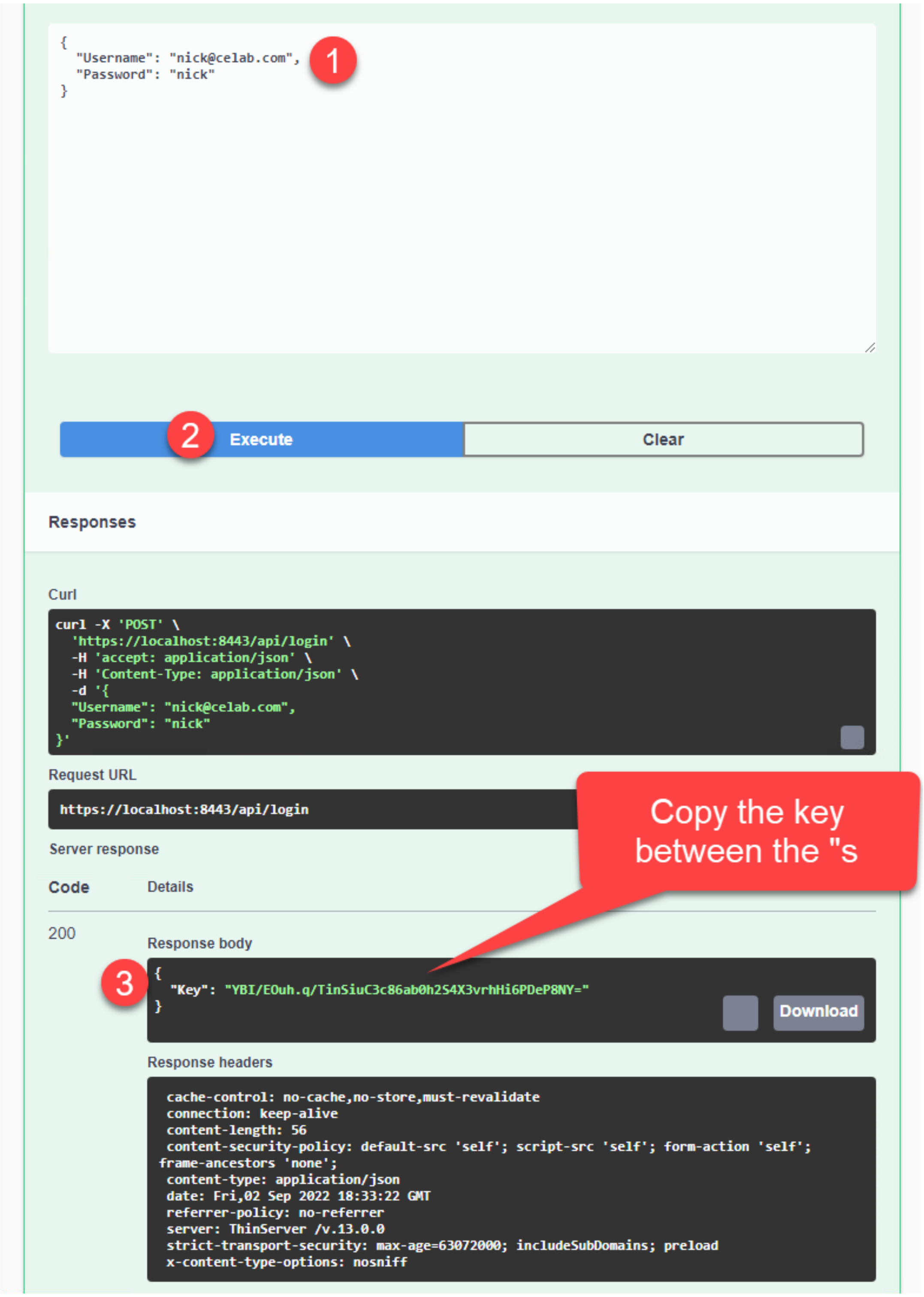
For this example, we will use the ThinManager API webpage to test triggering an Event. We navigate to https://localhost:8443/api/documentation. Since this is running a Swagger web interface, we can test the code.

To test the code, you need to authenticate with the ThinManager server using a certificate. You can do this by using the webpage to log in and generate a key for you:

1. Expand the Login section of the webpage by clicking on the **Login** bar and click on the **POST /api/login** bar to expand it. Click on the **Try it out** button.


ThinManager API Documentation
Settings
Not secure
https://localhost:8443/api/docu...
Sign in
ThinManager API
13.0.0
OAS3
/api/documentation/openapi.json
Authorize
Login
1
POST
/api/login
User login to obtain an API key
2
Parameters
3
Try it out

2. Replace the username and password with their respective values. Press the **Execute** button. Scroll down to the Server response section and copy and paste the key found in between the "s of the text within the Response body box.
    a. Note: You should use your UPN (User Principal Name) login account for Windows to log into the ThinManager server with the API. For example, domainname\user would be represented as user@domainname.com if you are using the UPN form. This user also needs to have permission to connect to and administer the ThinManager server as is set up in the ThinManager Security section of the ThinManager server.



```
{
  "Username": "nick@celab.com",
  "Password": "nick"
}
```

```
curl -X 'POST' \
  'https://localhost:8443/api/login' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "Username": "nick@celab.com",
  "Password": "nick"
}'
```

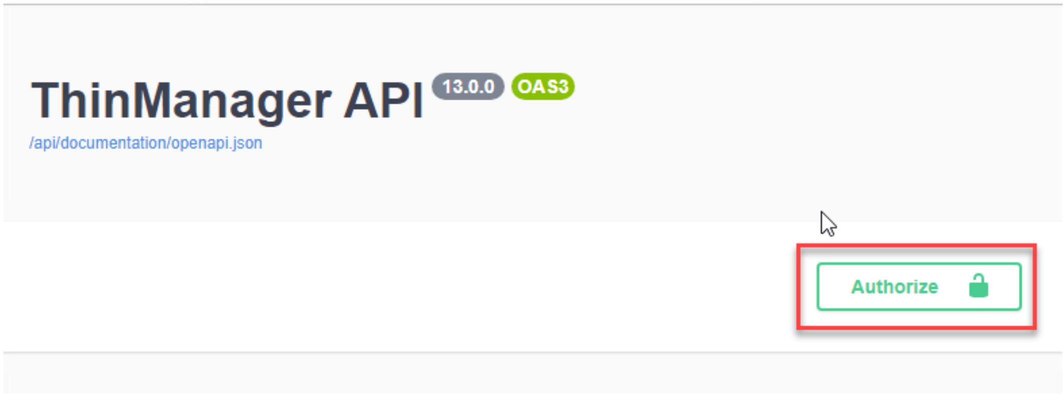
```
https://localhost:8443/api/login
```

```
{
  "Key": "YBI/EOuh.q/TinSiuC3c86ab0h2S4X3vrhHi6PDeP8NY="
}
```

```
 cache-control: no-cache,no-store,must-revalidate
 connection: keep-alive
 content-length: 56
 content-security-policy: default-src 'self'; script-src 'self'; form-action 'self';
frame-ancestors 'none';
 content-type: application/json
 date: Fri,02 Sep 2022 18:33:22 GMT
 referrer-policy: no-referrer
 server: ThinServer /v.13.0.0
 strict-transport-security: max-age=63072000; includeSubDomains; preload
 x-content-type-options: nosniff
```

Copy the key between the "s

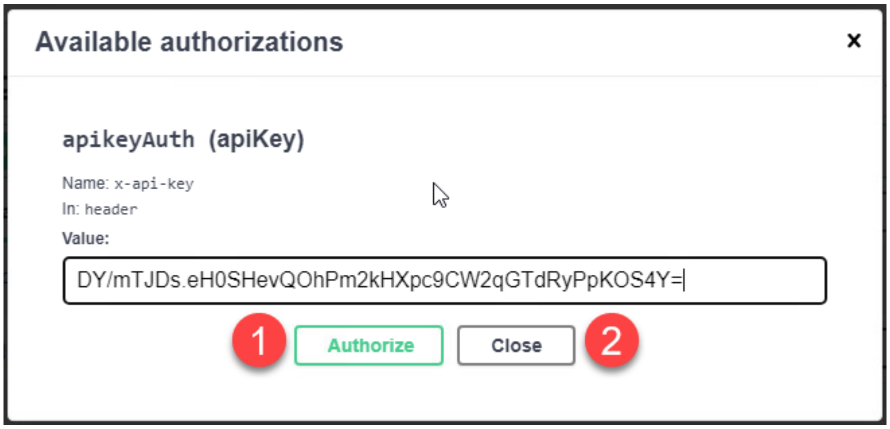
3. Scroll up to the top of the webpage and click on the **Authorize** button.



ThinManager API 13.0.0 OAS3

/api/documentation/openapi.json

4. Paste in the key copied previously and click the **Authorize** followed by the **Close** button.



Available authorizations

apikeyAuth (apiKey)

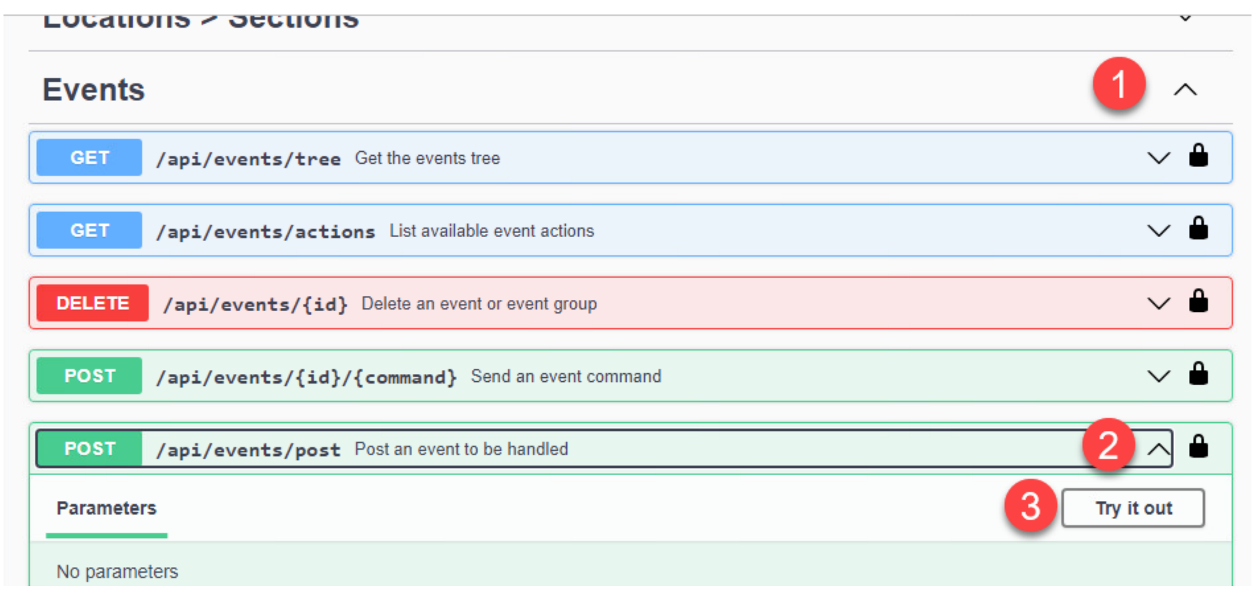
Name: x-api-key

In: header

Value:

DY/mTJDs.eH0SHevQOhPm2kHXpc9CW2qGTdRyPpKOS4Y=

Once the server is authorized, we can trigger an Event by going to the **Events -> POST /api/events/post** method, which will post an Event to be handled by the ThinManager server:



Locations > Sections

Events

GET /api/events/tree Get the events tree

GET /api/events/actions List available event actions

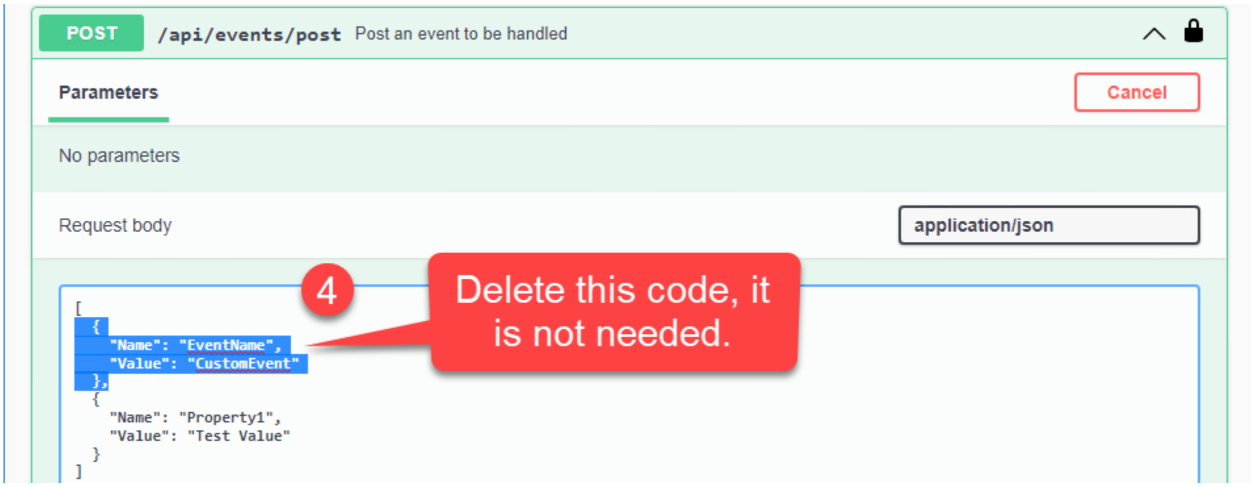
DELETE /api/events/{id} Delete an event or event group

POST /api/events/{id}/{command} Send an event command

POST /api/events/post Post an event to be handled

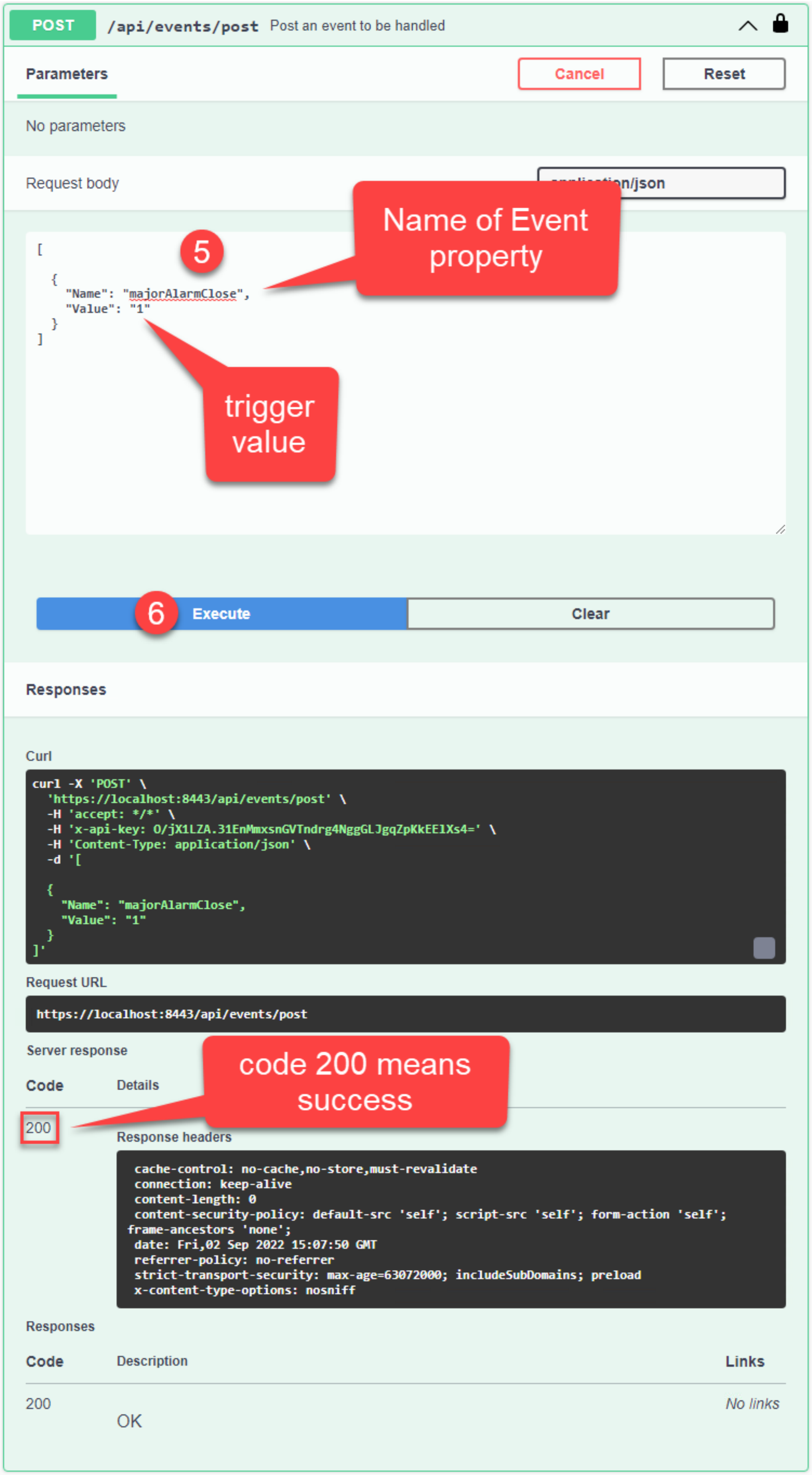
Parameters

No parameters



```
[
  {
    "Name": "EventName",
    "Value": "CustomEvent"
  },
  {
    "Name": "Property1",
    "Value": "Test Value"
  }
]
```

Delete this code, it is not needed.



`/api/events/post` Post an event to be handled

Parameters

No parameters

Request body

```
[
  {
    "Name": "majorAlarmClose",
    "Value": "1"
  }
]
```

5

Name of Event property

trigger value

6

Responses

Curl

```
curl -X 'POST' \
  'https://localhost:8443/api/events/post' \
  -H 'accept: */*' \
  -H 'x-api-key: O/jX1LZA.31EnMmxsnGVTndrg4NggGLJgqZpKkEE1Xs4=' \
  -H 'Content-Type: application/json' \
  -d '[

  {
    "Name": "majorAlarmClose",
    "Value": "1"
  }
]'
```

Request URL

```
https://localhost:8443/api/events/post
```

Server response

code 200 means success

Code	Details
200	Response headers

```
 cache-control: no-cache,no-store,must-revalidate
 connection: keep-alive
 content-length: 0
 content-security-policy: default-src 'self'; script-src 'self'; form-action 'self';
frame-ancestors 'none';
 date: Fri,02 Sep 2022 15:07:50 GMT
 referrer-policy: no-referrer
 strict-transport-security: max-age=63072000; includeSubDomains; preload
 x-content-type-options: nosniff
```

Responses

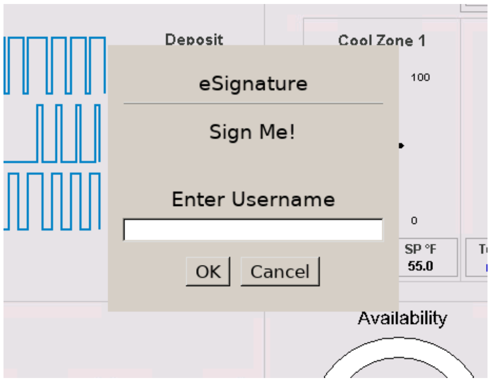
Code	Description	Links
200	OK	*No links*

A note that receiving code 200 means that the server was sent the event and it was received, but it does not necessarily mean that an event was triggered. The event still must be valid by matching an expression the server is expecting. You should manually verify that the event was received by the ThinManager server by checking the Event Log tab for the event you are trying to trigger. See the Enable, Disable, and View Event Log for Events section of this whitepaper for more details about the Event Log tab.

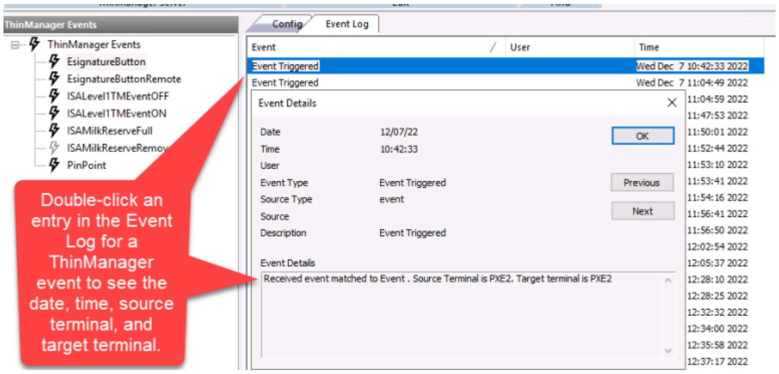
E Signature Events

E Signature Events are used to sign off on actions and processes within applications used during production. This integrates directly with the ThinManager Users and their associated authentication mechanisms for login such as passwords, PINs, badge readers, and fingerprint readers. This is separate from the E Signature functionality found in some HMI applications such as FactoryTalk View SE software. E Signatures with ThinManager require ThinManager software version 13.1 or newer. ThinManager E Signature implementation does not implement logging to an E Signature database. The control of what happens in the application (change a tag value, navigate to another display) is up to the programmer of the application to implement. Here is what the ThinManager E Signature functionality provides:

1. Give a method to require a user to log in to the terminal using a ThinManager user.



2. Provide information back to the application requesting an E Signature if the login was successful against AD and other parameters such as who the user was who logged in (see the *Get Data Back from the E Signature* section below).
3. Log that the E Signature Event took place in the ThinManager Event Log tab for the E Signature Event. This logging is limited to the date and time the E Signature was triggered, as well as the source and destination terminal. It does not provide any context information about which user logged in or what specific action the user was signing.



Event	User	Time
Event Triggered		Wed Dec 7 10:42:33 2022
Event Triggered		Wed Dec 7 11:04:49 2022
		11:04:59 2022
		11:47:53 2022
		11:50:01 2022
		11:52:44 2022
		11:53:10 2022
		11:53:41 2022
		11:54:16 2022
		11:56:41 2022
		11:56:50 2022
		12:02:54 2022
		12:05:37 2022
		12:28:10 2022
		12:28:25 2022
		12:32:32 2022
		12:34:00 2022
		12:35:58 2022
		12:37:17 2022

Event Details

Date	12/07/22
Time	10:42:33
User	
Event Type	Event Triggered
Source Type	event
Source	
Description	Event Triggered

Event Details

```
Received event matched to Event . Source Terminal is PXE2. Target terminal is PXE2
```

Double-click an entry in the Event Log for a ThinManager event to see the date, time, source terminal, and target terminal.

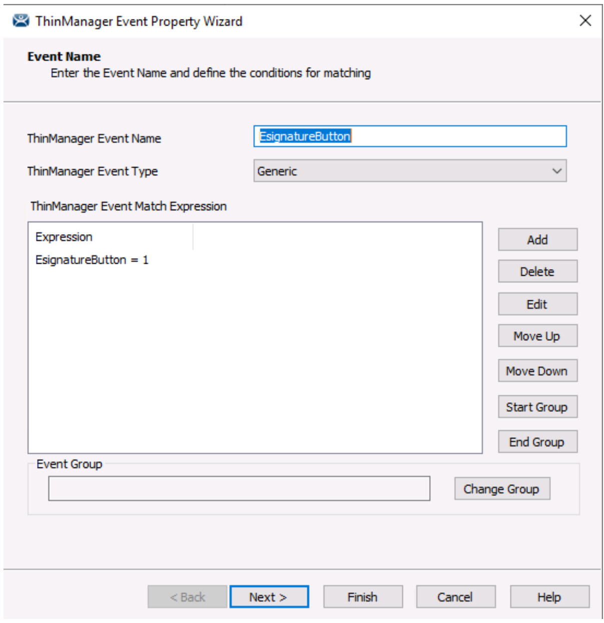
The programmer can optionally implement the following in their application:

1. Logging details of the E Signature to a database such as FactoryTalk Diagnostics in FactoryTalk View SE, or some other ODBC database. For more information about how to log to the FactoryTalk Diagnostics Log using VBA code, see RAID 54424, Sample VBA: Using a Wrapper Module to Log Standard Error Messages.

2. Controls on what happens based on a successful or failed E Signature login from the application. For example, you may wait to change a value in a critical process from the application until you have validated a successful E Signature login from the ThinManager terminal.

Create an E Signature Event

Create a new Event as described in the *Event Creation on the ThinManager Server* section of this white paper. The following are an example of the parameters you can configure for the **Event Name** and **Event Action** pages of the ThinManager Event Property Wizard, but please set them to your preferred values.



ThinManager Event Property Wizard

Event Name
Enter the Event Name and define the conditions for matching

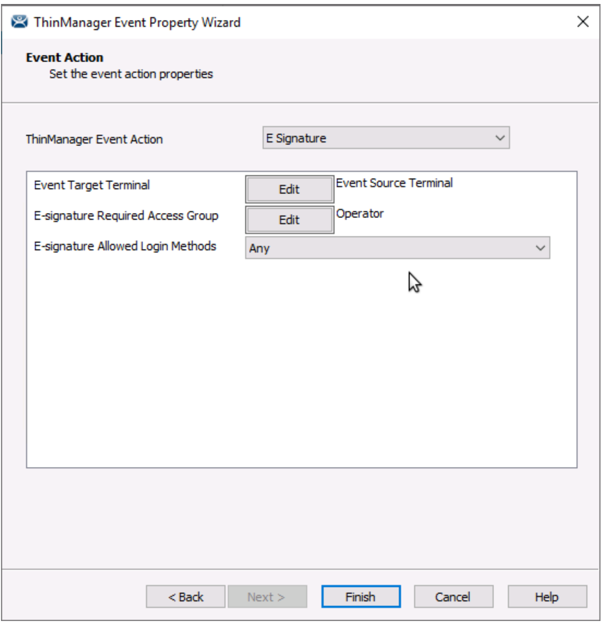
ThinManager Event Name: EsignatureButton

ThinManager Event Type: Generic

ThinManager Event Match Expression

Expression
EsignatureButton = 1

Event Group



ThinManager Event Property Wizard

Event Action
Set the event action properties

ThinManager Event Action: E Signature

Event Target Terminal: Event Source Terminal

E-signature Required Access Group: Operator

E-signature Allowed Login Methods: Any

VBA Code for E Signature Events

Numeric Input

Trigger an E Signature whenever the value of a numeric input is changed. In this example, we send two parameters called **CustomProperty1** and **CustomProperty2**. **CustomProperty1** is the match string of the ThinManager Event as shown in the screenshot for the Expression of the Event. The **CustomProperty2** parameter specifies a custom message to display on the login prompt for the E Signature login box. This will also be passed back from the ThinManager server as is shown in the Get Data Back from the E Signature section below. If this is not passed to the ThinManager server, the default message of “Signature Requested” will be displayed and passed back.

A generic Event is then sent to the ThinManager server using the **SendGenericEvent** method, with **CustomProperty1** and **CustomProperty2** passed through as parameters. They need to be passed to the server each as new lines of code (in most programming languages that would be `\n`). In VBA, to pass through a new line you concatenate the two strings with `& vbNewLine &`. Thus, the arguments passed to the Event are `CustomProperty1 & vbNewLine & CustomProperty2`.

```
Private Sub NumericInput1_Change()

Dim CustomProperty1 As String
CustomProperty1 = "EsignatureButton=1"
CustomProperty2 = "CONTEXT=SignThis"
TermMonControl1.SendGenericEvent (CustomProperty1 & vbNewLine &
CustomProperty2)
'MsgBox "You just triggered an E-Signature Event"

End Sub
```

Button

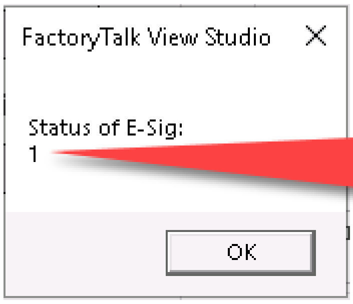
Trigger an E Signature whenever the value of a numeric input is changed. See the explanation in the *Numeric Input* section above for the description of code and parameters as they are identical except for the VBA event and method.

```
Private Sub GO_Button_12_Click()

Dim CustomProperty1 As String
CustomProperty1 = "EsignatureButton=1"
CustomProperty2 = "CONTEXT=Sign Me!"
TermMonControl1.SendGenericEvent (CustomProperty1 & vbNewLine &
CustomProperty2)
‘MsgBox "You just triggered an E-Signature Event"

End Sub
```

Get Data Back from the E Signature

Whenever an E Signature is trigged, get back the results of the E Signature using the **OnESignatureProcessEnd** event in VBA. This returns the status which indicates if a login was successful (0 indicates an invalid login, 1 indicates a successful login), and the properties of an E Signature Event such as the username and the content of the Event.

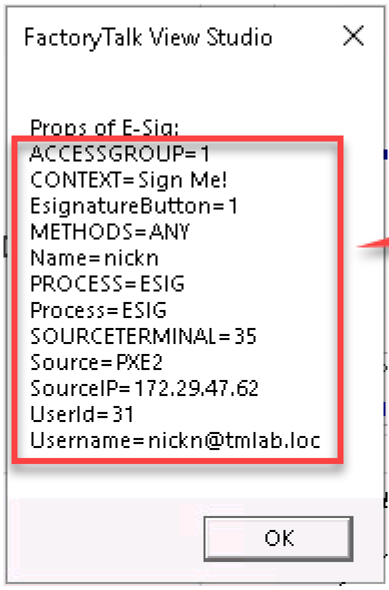
```
Private Sub TermMonControl1_OnESignatureProcessEnd(ByVal status
As String, ByVal props As String)

MsgBox "Status of E-Sig:" & vbNewLine & status
MsgBox "Props of E-Sig:" & vbNewLine & props

End Sub
```



FactoryTalk View Studio

Status of E-Sig:
1

Status of 1 indicates a successful login!



FactoryTalk View Studio

```
Props of E-Sig:
ACCESSGROUP=1
CONTEXT=Sign Me!
EsignatureButton=1
METHODS=ANY
Name=nickn
PROCESS=ESIG
Process=ESIG
SOURCETERMINAL=35
Source=PXE2
SourceIP=172.29.47.62
UserId=31
Username=nickn@tmlab.loc
```

All the information made available from the ThinManager server after the E-Signature event

Updated 01/04/2023

ThinManager – A Rockwell Automation Technology
1220 Old Alpharetta Rd - Suite 390
Alpharetta, GA 30005

1-877-239-4282
sales@thinmanager.com

For more information, please visit **thinmanager.com**