AlarmGroup Notifications Code

Send notifications from alarm groups.

Sending notifications by e-mail is done through scripts. This page provides an example that uses the C# language to send e-mails when an alarm has been triggered.

Since this is a characteristic of the group, to use this feature, set the column "NotificationMethod" on Alarm / Groups to the method that will handle the email sending. This means that the method will be called whenever an alarm in that respective group is triggered.

Configuration Process
- Creating a Sending Method
- Binding Method to Alarm Group

Example: Sending E-mail C#
Execution

Solution Example

Alarm Email and SMS

Configuration Process

To ensure the proper functioning of this tool, it is necessary to confirm two things: **if the sending method is correctly described**, and **if the alarm is triggering properly**.

Creating a Sending Method

The notification method is the script that will run when the alarm was triggered.

Go to Scripts / Classes and create C# class from the Server domain, then create a public method as the following.

```
public void NotificationMethod(AlarmEventInfo[] events)
```

For this example, the name "NotificationMethod" was chosen, but any name is possible. The essential thing is that the method is public and that the arguments are as described.

Binding Method to Alarm Group

After creating a method, we need to bind it with an AlarmGroup. This can be done through the "NotificationMethod" column in the **Alarms / Groups** table.

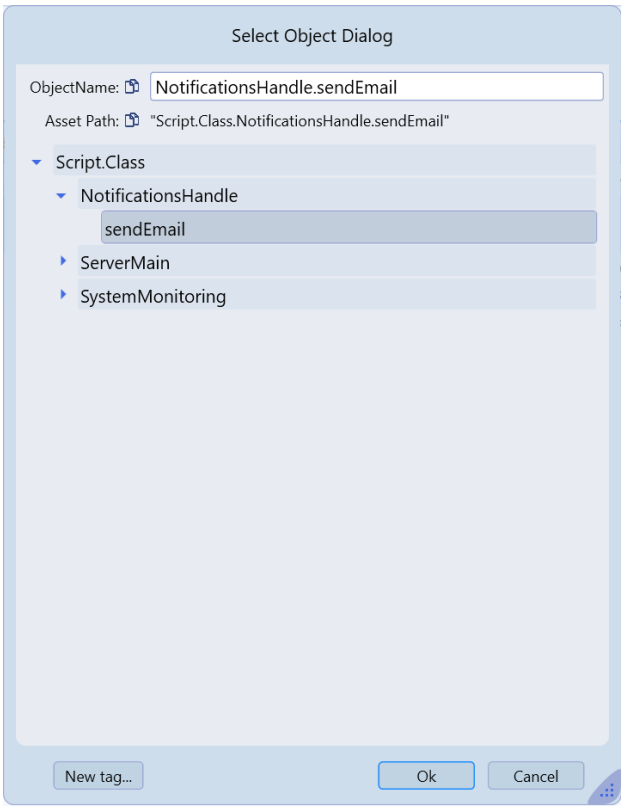
There are two ways to set this field, either by writing the full method path or by using the visual interface, clicking on "..." button on right side of this field.

If you chose to write the full path, the string will be like this:

Script.Class.<CLASS NAME>.<METHOD NAME>

For this example, it would be: "Script.Class.NotificationsHandle.sendEmail".

If you choose to do it by the interface, a display like this will appear:



The Script.Class tree shows every class with the domain Server.

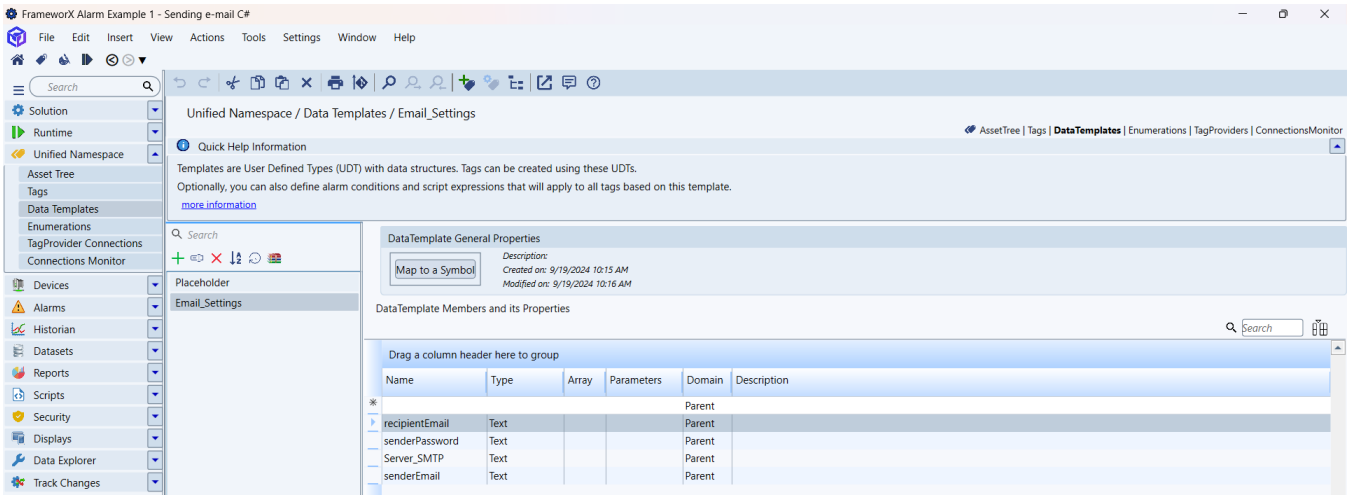
To select a method, navigate through this tree and select the method that will be used for the callback when an alarm happens. Note that the full path appears in the label "Asset Path".

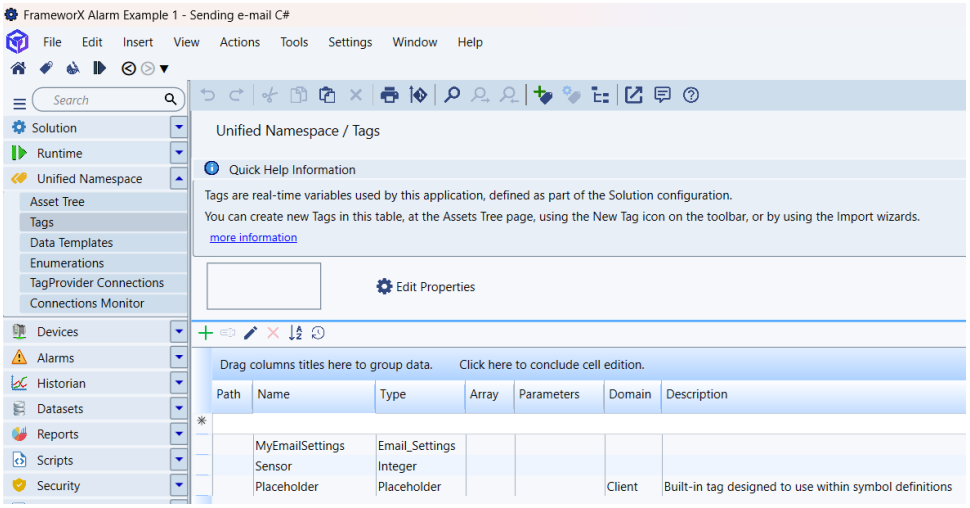
Click OK to save the changes.

Example: Sending E-mail C#

Creating Tags and Interface

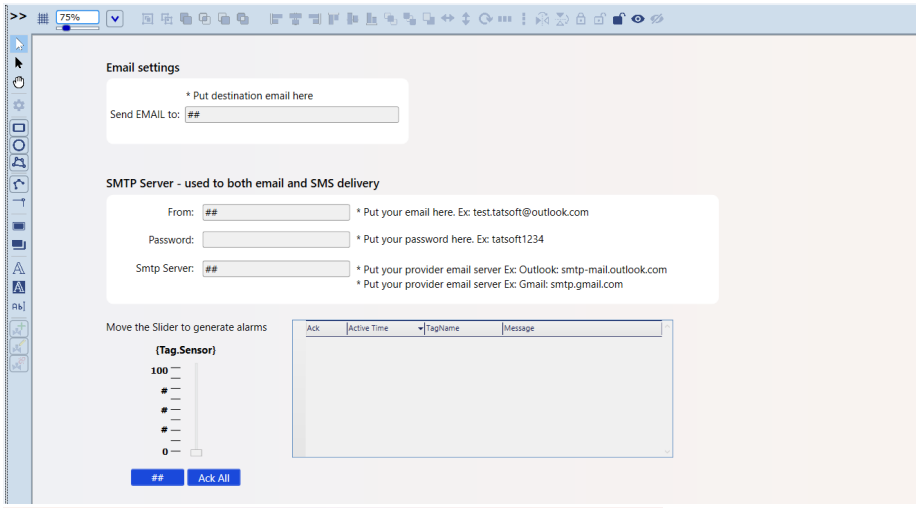
To send an e-mail, some information is needed, such as the source e-mail, the target e-mail, the SMTP link, and the password for the source account. Then, we need one tag for each of these properties. To better organize the solution tags, we will use a template specifically for these settings. Furthermore, the “Sensor” tag will be used to trigger the alarm condition. In order for your project to send an email when an alarm occurs, you need to configure some alarm items. See the images below:

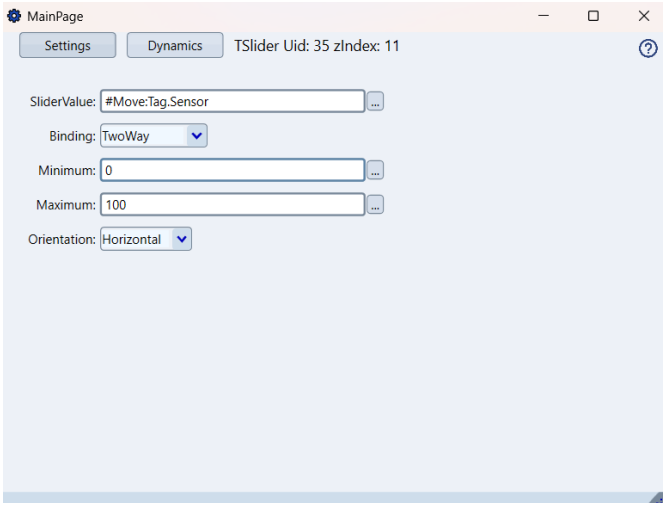




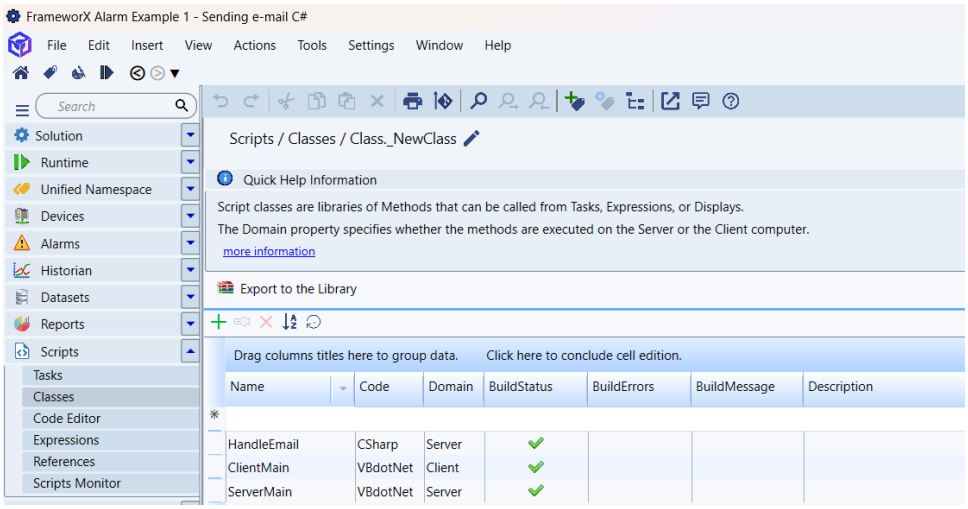
Display Configuration

To send an e-mail, we'll need the previously created information. You also need to add a way to change the sensor value and visualize the alarms.

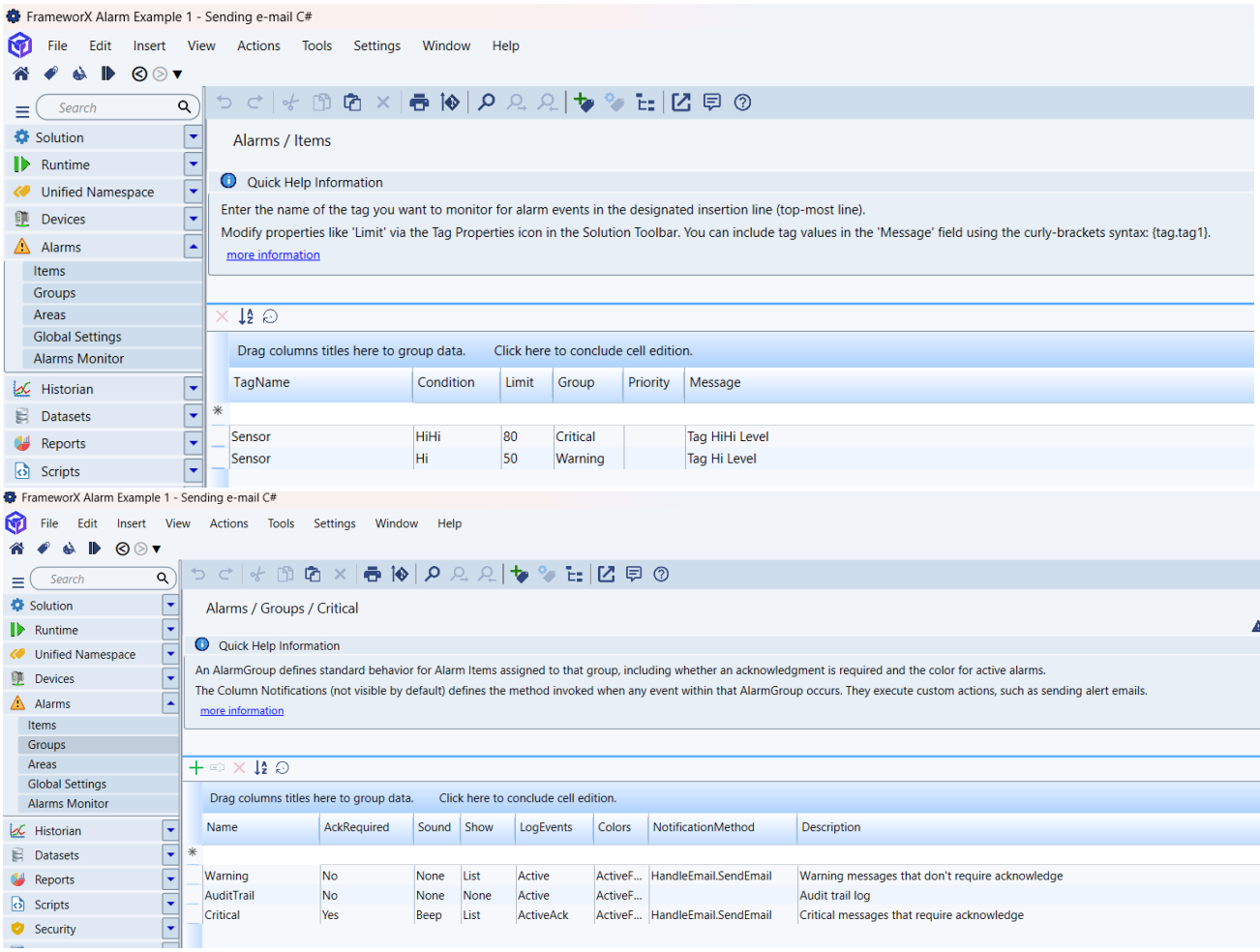




Create a Class Method



The next step is to configure the NotificationMethod column in the Alarms / Groups section like the following image:



The NotificationMethod then receives the alarm events that are being generated. See an example of code below in which an email is sent from a template called "MyEmailSettings"

```
public void SendEmail(AlarmEventInfo[] events)
{
 //Protection in case events its null
        if (events == null)
                return;

        //Get the first event
        AlarmEventInfo eventX = events[0];
        if  (eventX.State != 1)
                return;

        //Get information about the alarm event to create the body of the email
        string body = "Time: " + eventX.ActiveLocalTime.ToString() + "\n" +
        "Message = " +          eventX.Message + "\n" +
        "Area = " + eventX.Area + "\n" +
        "Group = " + eventX.Group + "\n" +
        "Tag = " +          eventX.TagName + "\n"          ;

        //Code to send email
        try
        {
            //Configuring the SMTP Client
                System.Net.Mail.SmtpClient mySmtpClient = new System.Net.Mail.SmtpClient(@Tag.
MyEmailSettings.Server_SMTP, 587);
        mySmtpClient.DeliveryMethod = System.Net.Mail.SmtpDeliveryMethod.Network;
        mySmtpClient.EnableSsl = true;
        mySmtpClient.UseDefaultCredentials = false;
```

```
        mySmtpClient.Credentials = new System.Net.NetworkCredential(@Tag.MyEmailSettings.senderEmail, @Tag.
MyEmailSettings.senderPassword);
        //mySmtpClient.Credentials = new System.Net.NetworkCredential("sender@example.com",
"yourPasswordHere");

                //Sending the email
                mySmtpClient.Send(@Tag.MyEmailSettings.senderEmail, @Tag.MyEmailSettings.recipientEmail,
"Email notification", body);

        }
        catch (Exception ex)
        {
                @Info.Trace("Error sending message: " + ex.Message);
        }

}
```

Execution

Every time an alarm happens when the project is running, the method configured in the NotificationMethod column is called. The method will receive an Array of the AlarmEventInfo class as a parameter. The following members can be used for the AlarmEventInfo class.

```
DateTimeOffset AckTime
DateTimeOffset ActiveLocalTime
DateTimeOffset ActiveTime
DateTimeOffset DateCreated
DateTimeOffset NormTime

string Area
string Category
string ColorBG
string ColorFG
string Comment
string Deadband
string DetailsValue
string Setpoint
string SetpointDeadband

int Condition
int Disable
int UnAck
int Priority
int Quality
int State

TimeSpan Duration

string Group
string Level
string Limit
string Limit1
string Limit2
string Message
string Setpoint
string SetpointDeadband
string TagName
string UserName
string Value
```

Once the NotificationMethod is defined, you can configure various actions, such as displaying a message box, playing an alarm sound, and more.

In this section...