Automation and Security Products

E-Mail

Purpose

The purpose of this script library is to add functions for automatically sending an e-mail when an alarm in the system is activated. The function can be filtered on alarm severity.

Prerequisities

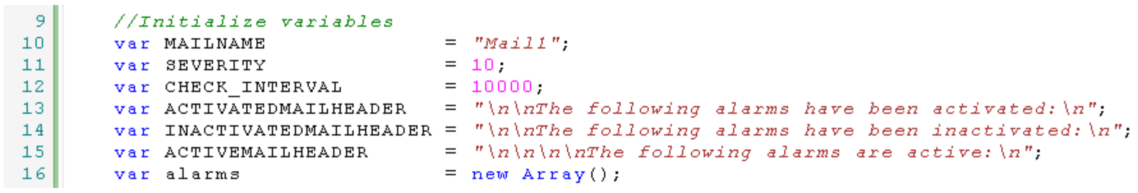
Import the script library, *scExtensions.zip,* under script libraries in Ethiris Admin.
Import the script library, *scMail.zip,* under script libraries in Ethiris Admin.
Create an e-mail in Ethiris Admin, called '*Mail1*'.
Requires Ethiris 14 or later.

License requirements

The solution requires the license function "Script libraries". Check the Kentima license model on Kentimas website for further information.

Script library with explanation

scMail is an autonomous scriptlibrary which means that the user should not call the script from the main script, instead the parameters below should be changed inside the script library where they are located.



```
 9    //Initialize variables
10    var MAILNAME                = "Mail1";
11    var SEVERITY                = 10;
12    var CHECK_INTERVAL          = 10000;
13    var ACTIVATEDMAILHEADER     = "\n\nThe following alarms have been activated:\n";
14    var INACTIVATEDMAILHEADER = "\n\nThe following alarms have been inactivated:\n";
15    var ACTIVEMAILHEADER        = "\n\n\n\nThe following alarms are active:\n";
16    var alarms                  = new Array();
```

Line 10: A constant which will contain the script name of the e-mail you want to use.
Line 11: A constant which will contain the severity you want to filter alarms on.
Line 12: A constant which will contain how often, in milliseconds, to check for new alarms.
Line 13: A constant which will contain the header for activated alarms, in the e-mail body.
Line 14: A constant which will contain the header for inactivated alarms, in the e-mail body.
Line 15: A constant which will contain the header for active alarms, in the e-mail body.
Line 16: A variable, of data type Array, which will contain a list of the alarms which will trigger an e-mail.

```
18      // Check for existance of E-mail '<MAILNAME>'
19      if(typeof DataStore[MAILNAME] == "undefined")
20      {
21          throw new Error("E-Mail '" + MAILNAME + "' is missing from the configuration");
22      }
23
24      // Check for existance of script library 'scExtensions'
25      if(typeof scExtensions == "undefined")
26      {
27          throw new Error("Script library 'scExtensions' is missing."); //Script library is missing, we can't continue
28      }
```

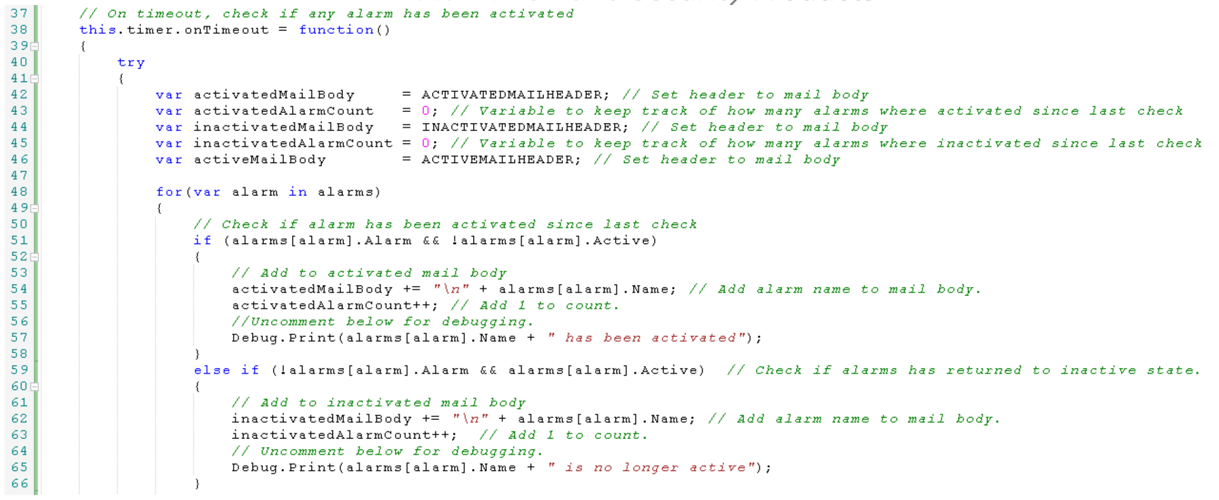
Line 19: A check to see if the e-mail name entered in constant *MAILNAME actually exists*.
Line 25: A check to see if the required script library *scExtensions* actually exists.

```
30      // Select ONE of the following options for finding alarms.
31      scExtensions.FindAllAlarms(alarms, "", DataStore);
32      //scExtensions.FindAllAlarms(alarms, "", DataStore, SEVERITY);
```

Line 31: Use function *FindAllAlarms,* which is defined in the script library *scExtensions.* The function till return all alarms defined in the system. Optionally you can enter the parameter *SEVERITY* to filter alarms on a specific sverity.

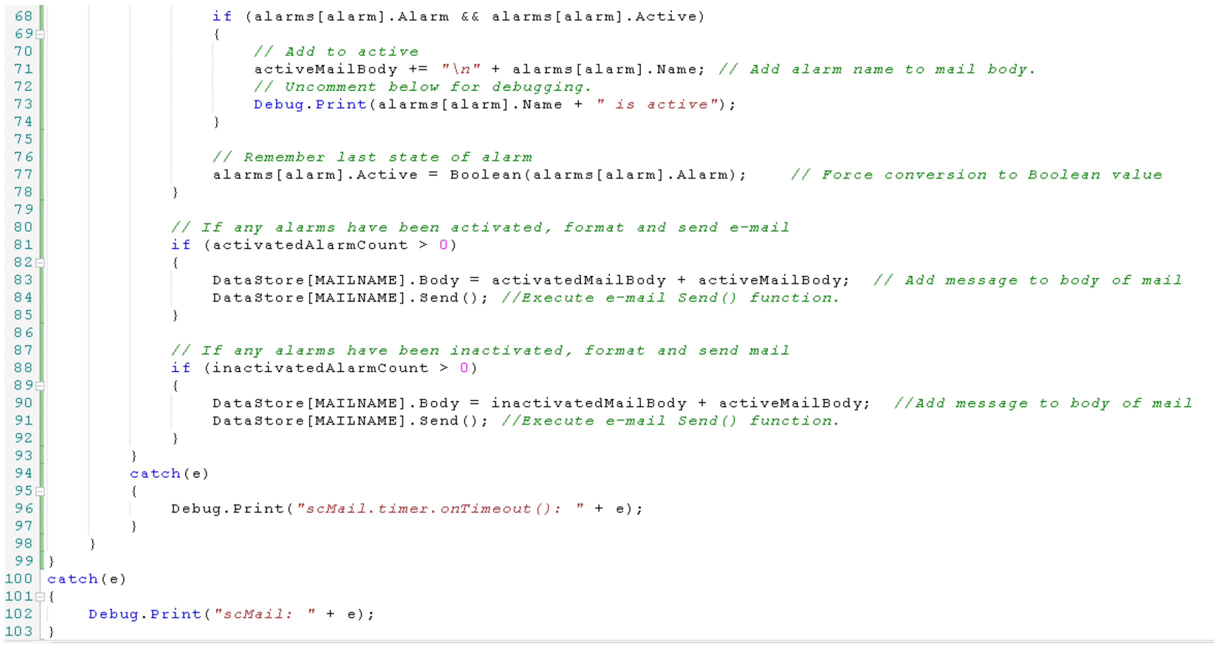
```
34      // Create a cyclic timer for sending e-mails, interval is set in the constant <CHECK_INTERVAL>
35      this.timer = new Timer(CHECK_INTERVAL, true);
```

Line 31: Define a timer object, using constant *CHECK_INTERVAL* as timeout parameter. The value '*true*' means it will start and run cyclically.



```
37      // On timeout, check if any alarm has been activated
38      this.timer.onTimeout = function()
39      {
40          try
41          {
42              var activatedMailBody     = ACTIVATEDMAILHEADER; // Set header to mail body
43              var activatedAlarmCount   = 0; // Variable to keep track of how many alarms where activated since last check
44              var inactivatedMailBody   = INACTIVATEDMAILHEADER; // Set header to mail body
45              var inactivatedAlarmCount = 0; // Variable to keep track of how many alarms where inactivated since last check
46              var activeMailBody        = ACTIVEMAILHEADER; // Set header to mail body
47
48              for(var alarm in alarms)
49              {
50                  // Check if alarm has been activated since last check
51                  if (alarms[alarm].Alarm && !alarms[alarm].Active)
52                  {
53                      // Add to activated mail body
54                      activatedMailBody += "\n" + alarms[alarm].Name; // Add alarm name to mail body.
55                      activatedAlarmCount++; // Add 1 to count.
56                      //Uncomment below for debugging.
57                      Debug.Print(alarms[alarm].Name + " has been activated");
58                  }
59                  else if (!alarms[alarm].Alarm && alarms[alarm].Active)  // Check if alarms has returned to inactive state.
60                  {
61                      // Add to inactivated mail body
62                      inactivatedMailBody += "\n" + alarms[alarm].Name; // Add alarm name to mail body.
63                      inactivatedAlarmCount++;  // Add 1 to count.
64                      // Uncomment below for debugging.
65                      Debug.Print(alarms[alarm].Name + " is no longer active");
66                  }
```

Line 38: Define *onTimeout* function, which will be triggered each time *CHECK_INTERVAL* runs out.
Line 48: Go through list of alarms and build the corresponding e-mail body, based on current alarms states.
Line 51: Check if any alarms were activated since last check.
Line 59: Check if any alarms were inactivated since last check.



```
68                  if (alarms[alarm].Alarm && alarms[alarm].Active)
69                  {
70                      // Add to active
71                      activeMailBody += "\n" + alarms[alarm].Name; // Add alarm name to mail body.
72                      // Uncomment below for debugging.
73                      Debug.Print(alarms[alarm].Name + " is active");
74                  }
75
76                  // Remember last state of alarm
77                  alarms[alarm].Active = Boolean(alarms[alarm].Alarm);    // Force conversion to Boolean value
78              }
79
80              // If any alarms have been activated, format and send e-mail
81              if (activatedAlarmCount > 0)
82              {
83                  DataStore[MAILNAME].Body = activatedMailBody + activeMailBody;  // Add message to body of mail
84                  DataStore[MAILNAME].Send(); //Execute e-mail Send() function.
85              }
86
87              // If any alarms have been inactivated, format and send mail
88              if (inactivatedAlarmCount > 0)
89              {
90                  DataStore[MAILNAME].Body = inactivatedMailBody + activeMailBody;  //Add message to body of mail
91                  DataStore[MAILNAME].Send(); //Execute e-mail Send() function.
92              }
93          }
94          catch(e)
95          {
96              Debug.Print("scMail.timer.onTimeout(): " + e);
97          }
98      }
99  }
100 catch(e)
101 {
102     Debug.Print("scMail: " + e);
103 }
```

Line 68: Check if any alarms were already active since last check.
Line 81: Check if there are any activated alarms and execute e-mail *SEND()* method on **line 84.**
Line 88: Check if there are any inactivated alarms and execute e-mail *SEND()* method on **line 91.**
Line 94: *Try/catch* is used catch any exceptions thrown by the script code and print it to the *Debug Output Panel*, in Ethiris Admin.