Article
Michael Braam · Jun 2, 2021 3m read

Get the most out of InterSystems SAM - implement your own alert handler

InterSystems SAM is a great tool to monitor your **InterSystems IRIS** and **InterSystems IRIS For Health** clusters on prem or in a cloud environment. This article describes how you can implement a customized alert handler. This is currently an undocumented and most likely an unknown feature of InterSystems SAM. With future releases it will be probably made easier to leverage this useful concept.

In the interest of shortness, I will only mention InterSystems IRIS in this article, but the following applies to InterSystems IRIS and InterSystems IRIS For Health.

InterSystems SAM is a cluster of five containers and each of these containers plays a specific role. These are:

- **Grafana** to build dashboards to visualize your selected system metrics, i.e. metrics which InterSystems IRIS provides out of the box, and your own (application-)metrics.
- **Prometheus**, a cloud native monitoring tool which collects time series data from the target instances
- **Nginx** as the web server
- the **Alertmanager** which collects InterSystems IRIS alerts and Prometheus alerts
- and the **SAM Manager** which is the heart of SAM. This is an InterSystems IRIS instance in which all the metric data is stored

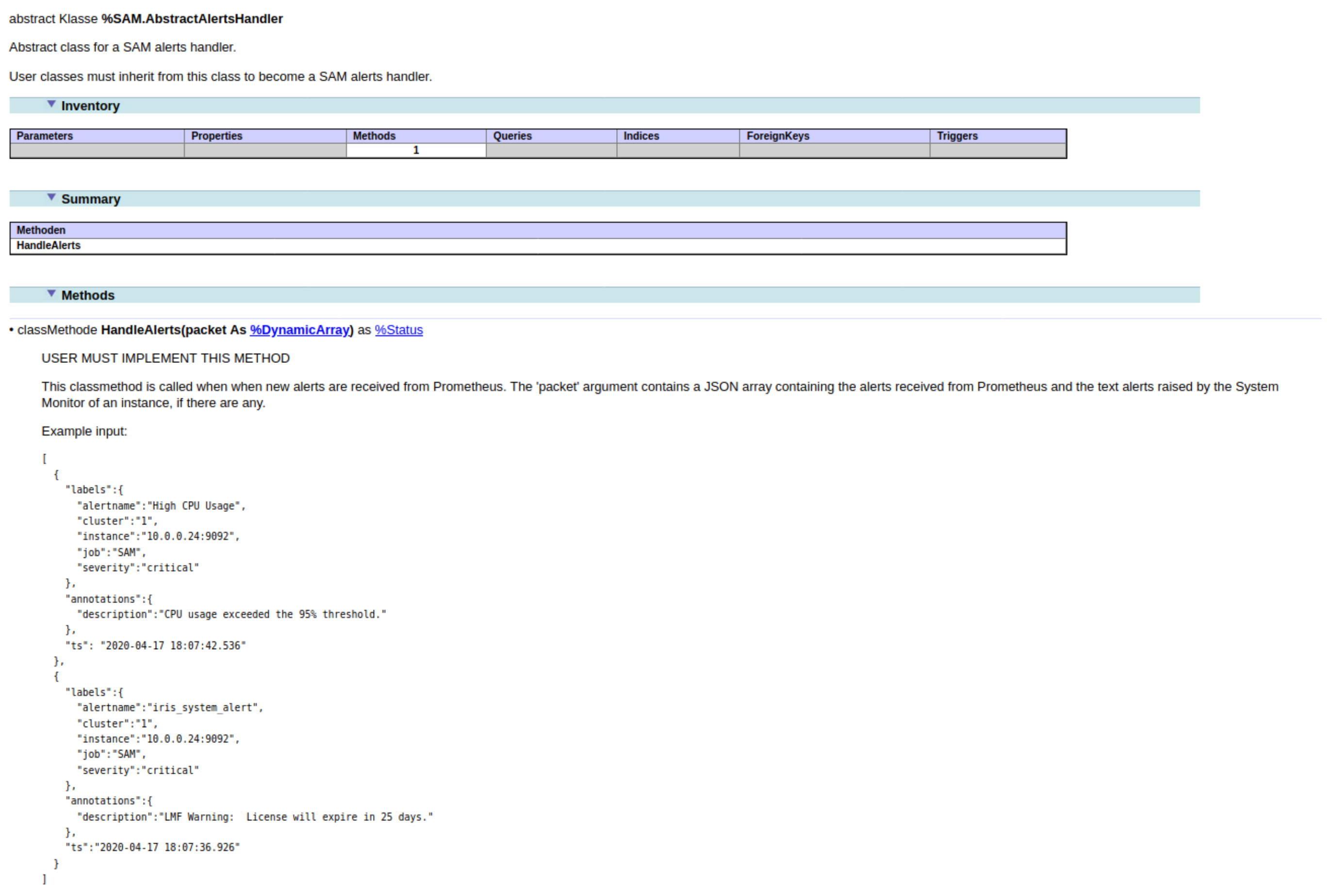
InterSystems SAM triggers two different classes of alerts.

- **InterSystems IRIS alerts** which are defined by InterSystems IRIS
- **User-defined alerts**. These are alerts which you can specify in so-called alert rules in the SAM portal. For details how to specify alert-rules see the documentation (https://docs.intersystems.com/sam/csp/docbook/Doc.View.cls?KEY=ASAM#ASAMedit_alert)

Alerts that are triggered are displayed in the SAM portal but what if you wish to notify someone who is responsible for the monitored InterSystems IRIS cluster via different communication channels like e-Mail or SMS etc. so that he can take care of the cluster and fix the problem. That's where a customized alert handler comes into play.

This article describes the neccessary steps to implement your own alert handler for your SAM Manager instance. In this example the alert handler will send an e-Mail to a preconfigured e-mail address along with the alert information.

The key here is that there is a "hidden" abstract class ***%SAM.AbstractAlertsHandler*** in the SAM Manager instance. See the class refernce below:



abstract Klasse **%SAM.AbstractAlertsHandler**

Abstract class for a SAM alerts handler.

User classes must inherit from this class to become a SAM alerts handler.

Inventory

Parameters	Properties	Methods	Queries	Indices	ForeignKeys	Triggers
		1				

Summary

Methoden
HandleAlerts

Methods

• classMethode **HandleAlerts(packet As %DynamicArray)** as %Status

USER MUST IMPLEMENT THIS METHOD

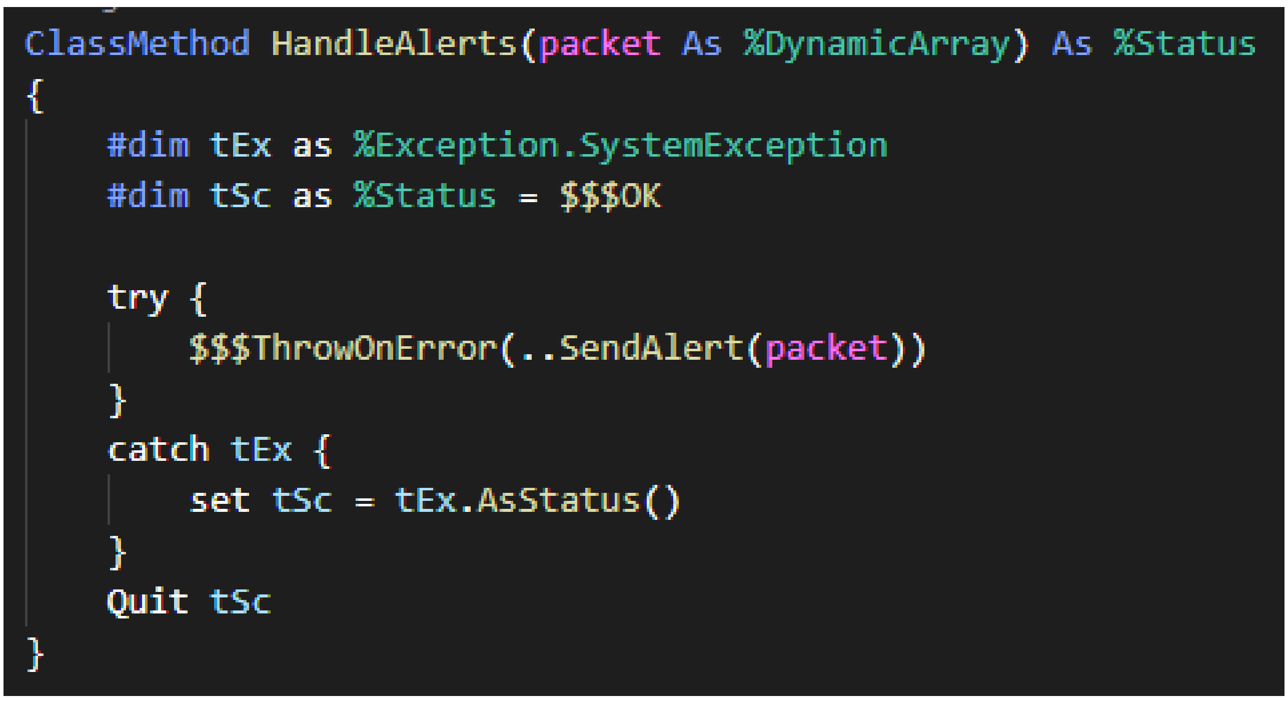
This classmethod is called when when new alerts are received from Prometheus. The 'packet' argument contains a JSON array containing the alerts received from Prometheus and the text alerts raised by the System Monitor of an instance, if there are any.

Example input:

```
[
  {
    "labels":{
      "alertname":"High CPU Usage",
      "cluster":"1",
      "instance":"10.0.0.24:9092",
      "job":"SAM",
      "severity":"critical"
    },
    "annotations":{
      "description":"CPU usage exceeded the 95% threshold."
    },
    "ts": "2020-04-17 18:07:42.536"
  },
  {
    "labels":{
      "alertname":"iris_system_alert",
      "cluster":"1",
      "instance":"10.0.0.24:9092",
      "job":"SAM",
      "severity":"critical"
    },
    "annotations":{
      "description":"LMF Warning:  License will expire in 25 days."
    },
    "ts":"2020-04-17 18:07:36.926"
  }
]
```

You have to derive your own alert handler from this abstract class first and then implement the classmethod HandleAlerts. The method receives a JSON array with the alert details.

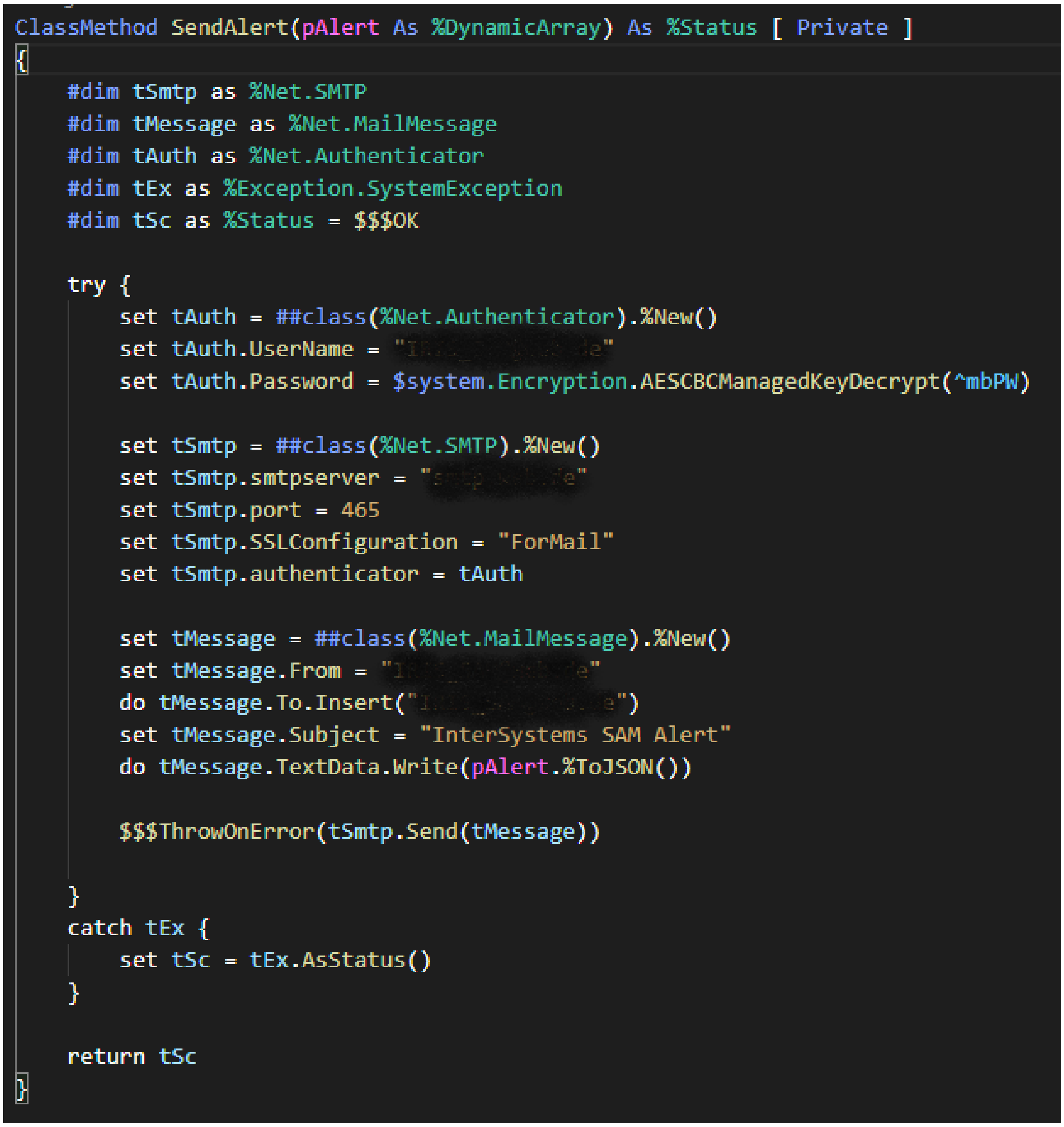
So in my simple example the alert handler is a class ***SAM.AlertHandler*** which subclasses %SAM.AbstractAlertsHandler. The implementation of the HandleAlerts() method is simple. See the full code below:



```
ClassMethod HandleAlerts(packet As %DynamicArray) As %Status
{
    #dim tEx as %Exception.SystemException
    #dim tSc as %Status = $$$OK

    try {
        $$$ThrowOnError(..SendAlert(packet))
    }
    catch tEx {
        set tSc = tEx.AsStatus()
    }
    Quit tSc
}
```

It delegates the work to the method ***SendAlert***. It assumes that you have an e-mail user account which you can use for the alerts. The e-mail password is stored in the database, but the password is encrypted using the data-element encryption capability of InterSystems IRIS. Therefore you have to create an encrytion key in the SAM Manager and this key must be loaded. Otherwise the alert handler will not be able to send the e-Mail. My e-mail provider requires SSL/TLS to send the e-mail. I have defined an SSL-configuration ***ForMail*** in the SAM Manager instance. The rest of the code is straight forward. See the code below:



```
ClassMethod SendAlert(pAlert As %DynamicArray) As %Status [ Private ]
{
    #dim tSmtp as %Net.SMTP
    #dim tMessage as %Net.MailMessage
    #dim tAuth as %Net.Authenticator
    #dim tEx as %Exception.SystemException
    #dim tSc as %Status = $$$OK

    try {
        set tAuth = ##class(%Net.Authenticator).%New()
        set tAuth.UserName = "[redacted]"
        set tAuth.Password = $system.Encryption.AESCBCManagedKeyDecrypt(^mbPW)

        set tSmtp = ##class(%Net.SMTP).%New()
        set tSmtp.smtpserver = "[redacted]"
        set tSmtp.port = 465
        set tSmtp.SSLConfiguration = "ForMail"
        set tSmtp.authenticator = tAuth

        set tMessage = ##class(%Net.MailMessage).%New()
        set tMessage.From = "[redacted]"
        do tMessage.To.Insert("[redacted]")
        set tMessage.Subject = "InterSystems SAM Alert"
        do tMessage.TextData.Write(pAlert.%ToJSON())

        $$$ThrowOnError(tSmtp.Send(tMessage))

    }
    catch tEx {
        set tSc = tEx.AsStatus()
    }

    return tSc
}
```

It simply copies the received JSON array with the alert details to the e-mail body and sends it to the recipients.

There is one remaining issue. You cannot directly create and edit the alert handler class in your SAM Manager instance. The default configuration does not allow connections from your IDE (IRIS studio or VSCode) to the SAM manager instance.

To resolve this I have created and edited the alert handler class in my InterSystems IRIS instance and have imported it into the SAM namespace of my SAM Manager instance. Another option would be to modify the docker-compose file to start InterSystems SAM, but I didn't want to touch this.

Once you have successfully loaded you alert handler class into your InterSystems SAM Manager instance the ***HandleAlerts*** method will be executed for every alert that is triggered in your monitored InterSystems IRIS cluster.

Enjoy!

#Best Practices #System Alerting and Monitoring (SAM) #InterSystems IRIS #InterSystems IRIS for Health

Source URL:https://community.intersystems.com/post/get-most-out-intersystems-sam-implement-your-own-alert-handler