# Custom Job Sample

## Overview

Peregrine MS Scheduler provides the ability to schedule any job in the enterprise even outside of Neuron ESB. Using Custom Job, you can implement your solution in any .NET language and Visual Studio. The custom job defined using the interface provided by Peregrine MS, can be configured, triggered and monitored using Peregrine MS. In this sample, a demo functionality is implemented. It checks a log folder for specific log files containing specific keyword and emails if a match is found. This functionality is for demo only and it can be any other functionality you choose to implement to try the custom job.

## Solution Framework

You need to create a .NET framework 4.8 DLL project and reference following four DLLs from Neuron Instance folder. Usually this folder is C:\Program Files\Neudesic\Neuron ESB V3\DEFAULT.

Quartz.dll

Log4net.dll

Neuron.Esb.dll

Peregrine.Scheduler.Interfaces.Core.dll

You need to implement IPeregrineJob interface your custom job class. This interface provides the link between your custom class and Peregrine MS scheduler host.

This interface has following members.

|  |  |  |
| --- | --- | --- |
| Name | Type | Purpose |
| JobKey | Custom class JobKey | JobKey consists of Job Name and Job Group defined in the Peregrine MS job setup. |
| JobDataDictionary | SerializationDictionary <string, object> | JobDataDictionary consists of all the Job Data configured for the Job in Peregrine MS job setup. |
| EntityName | String | Represents the name of Neuron Entity in Peregrine MS job setup. Not applicable for custom job. |
| Configuration | Neuron ESB class ESBConfiguration | Gives access to the currently configured Neuron solution configuration. |
| Log | ILog interface | Gives access to the Neuron ESB Scheduler host log. You can use this to log any messages to the Neuron ESB Scheduler log. |
| IsCancelled | Boolean | Execute method of the interface passes in a JobExecutionContext object. One of the members of JobExecutionContext is CancellationToken. When Cancellation is requested from the Peregrine MS admin, this token will be set. You can implement functionality to check for this and implement cancellation cleanup logic. After that you set the IsCancelled to true and Cancellation Detail to your custom message. |
| CancellationDetail | String |
| LastException | String | If the last invocation of the job failed, this will have the exception message from the last invocation. |
| LastExecutionTime | Timespan | This will indicate how long the last invocation of this job ran. |
| LastFinishTimeUTC | Datetime | This will indicate the exact time when the last invocation of this job finished. |
| Request | String | Optional request message that is configured on the Job Setup is passed in this property |
| Response | String | This property can be set by your implementation to return any response for use in next invocation. |
| TransactionID | Guid | Unique Id assigned to this invocation |
| MessageId | Guid | Unique Id assigned to the request message |
| TriggerName | String | Name of the trigger associated with this job in Peregrine MS setup |
| Description | String | Description of the job associated with this job in Peregrine MS setup |
| Group | String | Group name associated with this job in Peregrine MS setup |
| Name | String | Name associated with this job in Peregrine MS setup |

This interface has only one method Execute with the following signature.

public Task Execute(IJobExecutionContext context)

The IJobExecutionContext interface provides context of the job execution for use in your implementation. This interface is implemented by Quartz.Net and provides all access to Quartz internals. Refer to Quartz.Net documentation for details.

Important to note that the MergedJobDataMap property of this context object should be used to properly account for the fact that Job Data can be defined at job level but may have been overridden at trigger level. For example, a nightly trigger may have different job data than a weekly trigger.

## Solution setup

The sample Visual Studio Solution for this is located under the <Neuron Install Folder>\Samples\Scenarios\Scheduler\CustomLogMonitor. In this solution example, a demo functionality is shown.

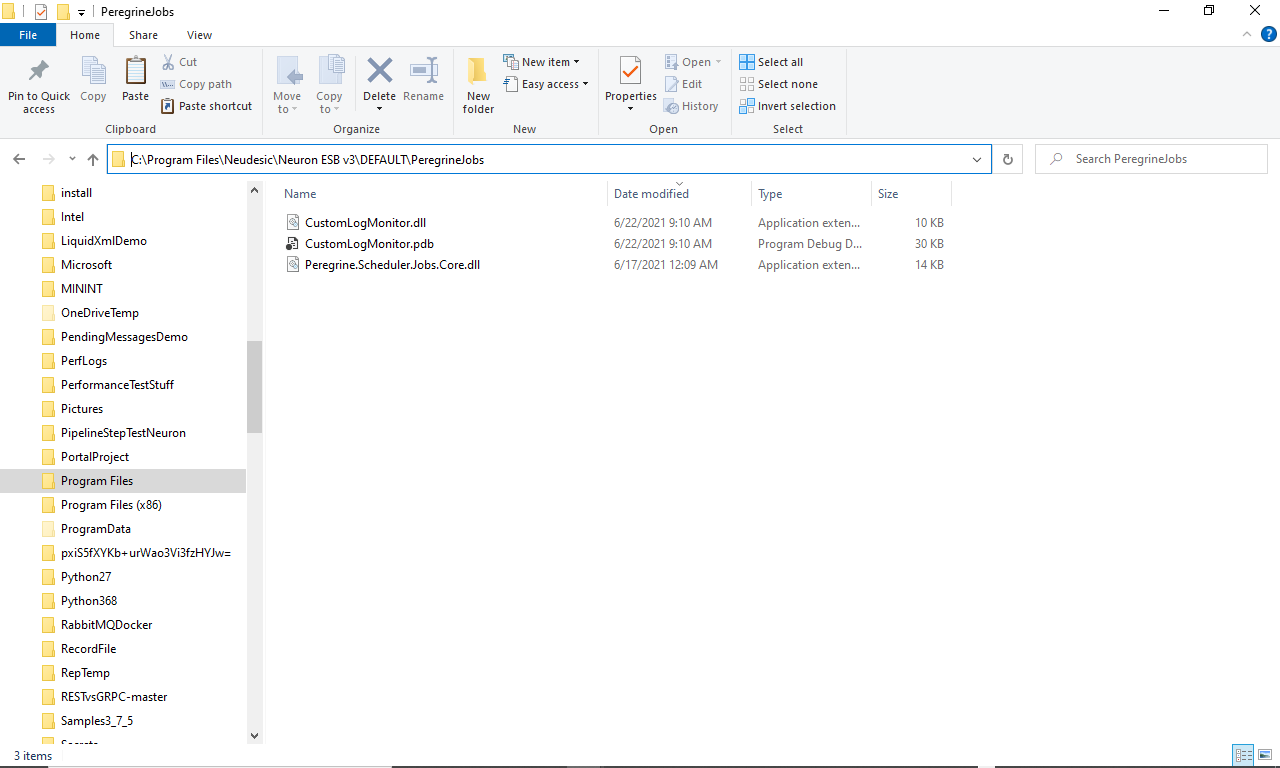
The Execute method of the IPeregrineJob interface is implemented. In this method, it reads tail end of some log files for specific search term and emails the results every time the job is invoked. It checks for cancellation token and promptly stops execution if the job is cancelled from the Peregrine MS admin.

It demonstrates the use of Job Data to pass in configuration information for the custom job.

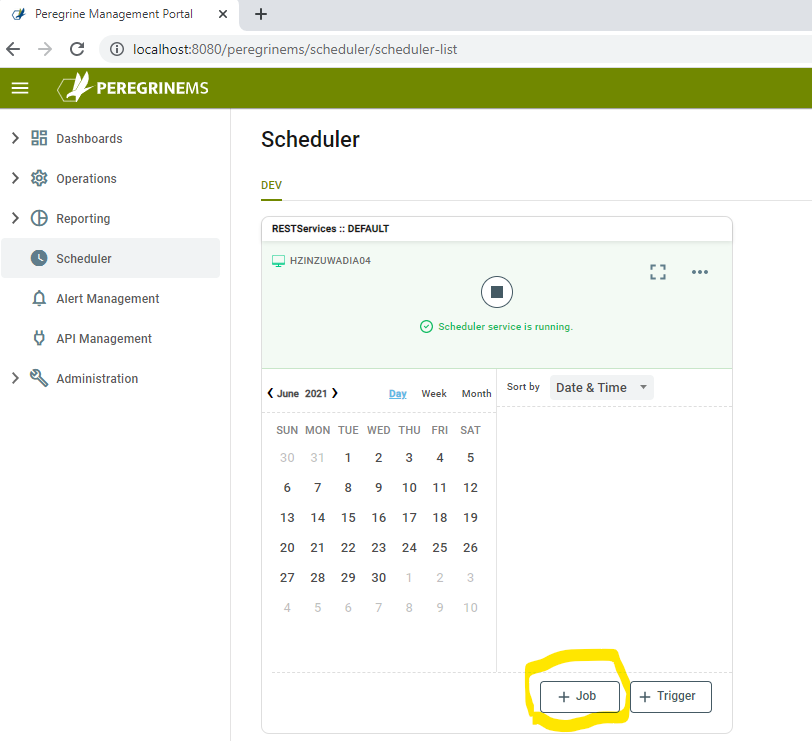
This functionality demo requires access and rights to SMTP server for sending email. If you don’t have this available, change the implementation to write to a temporary file instead of sending email. The email functionality is not essential to run this sample.

## Peregrine MS Setup

Once the Visual Studio solution is built, copy the dll file to PeregrineJobs folder under instance folder. Make sure to stop the Neuron Scheduler Host instance if you are redeploying. After copying the files, make sure to start the Neuron Scheduler Host instance if it is not running.



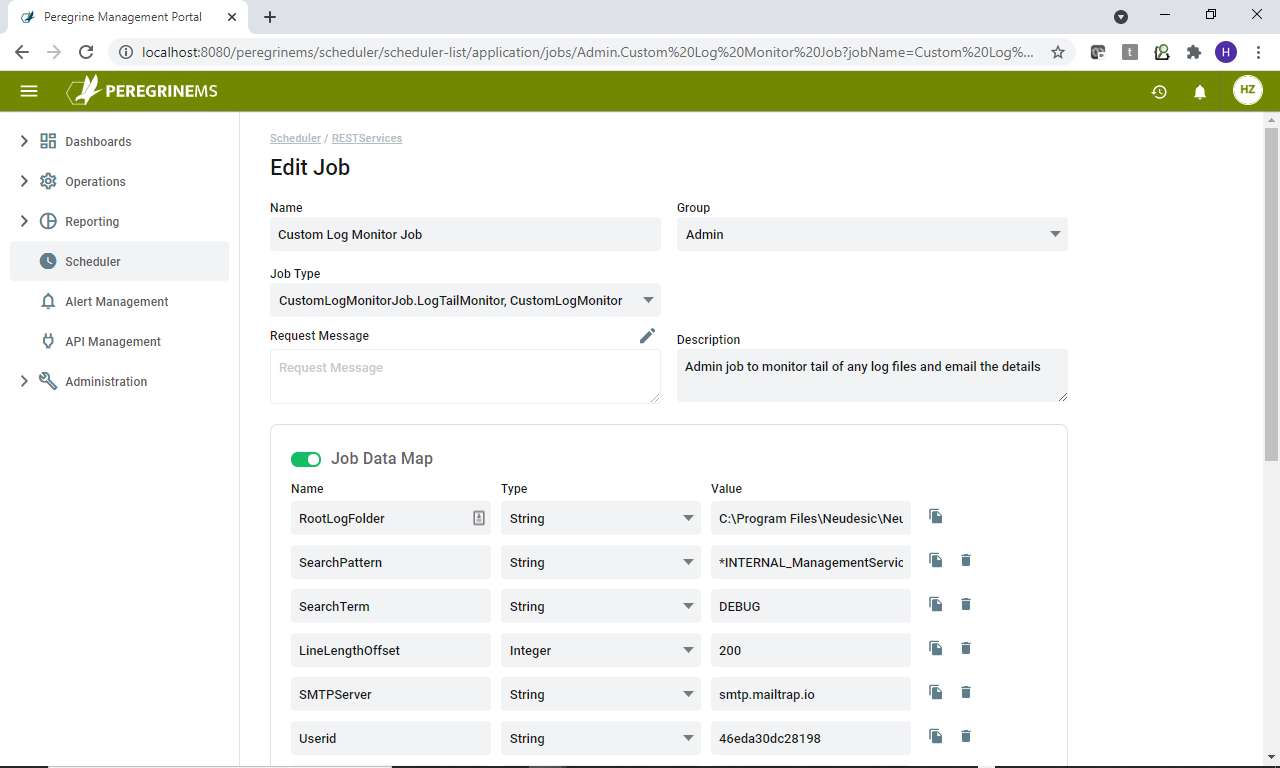
Setup the Job in Peregrine MS under Scheduler add job button as below.



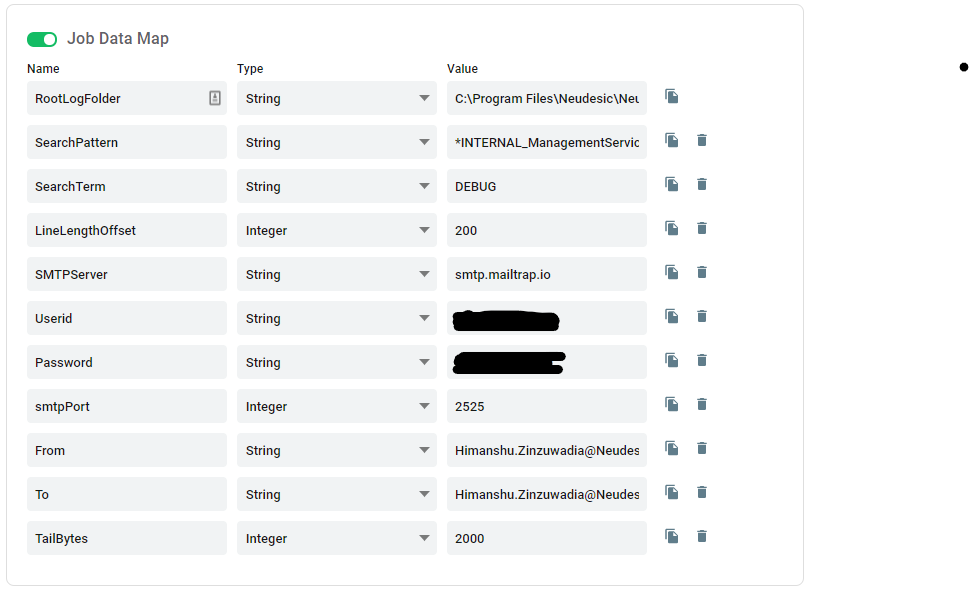
Setup the job as shown below.

The Job Type drop-down box should show your custom job type, if you completed the previous step of dropping your dll to the Neuron Instance Folder and Stopping / Starting the Scheduler Endpoint Host.

Enter name of job, a group that may be existing or a new value can be entered, select Job Type and enter a description.



Under job data map, enter the values for all the configuration items in the table below. You can use the copy button and trash button next to the Job Data Map item to add or delete an item. Substitute appropriate values in your own setup.

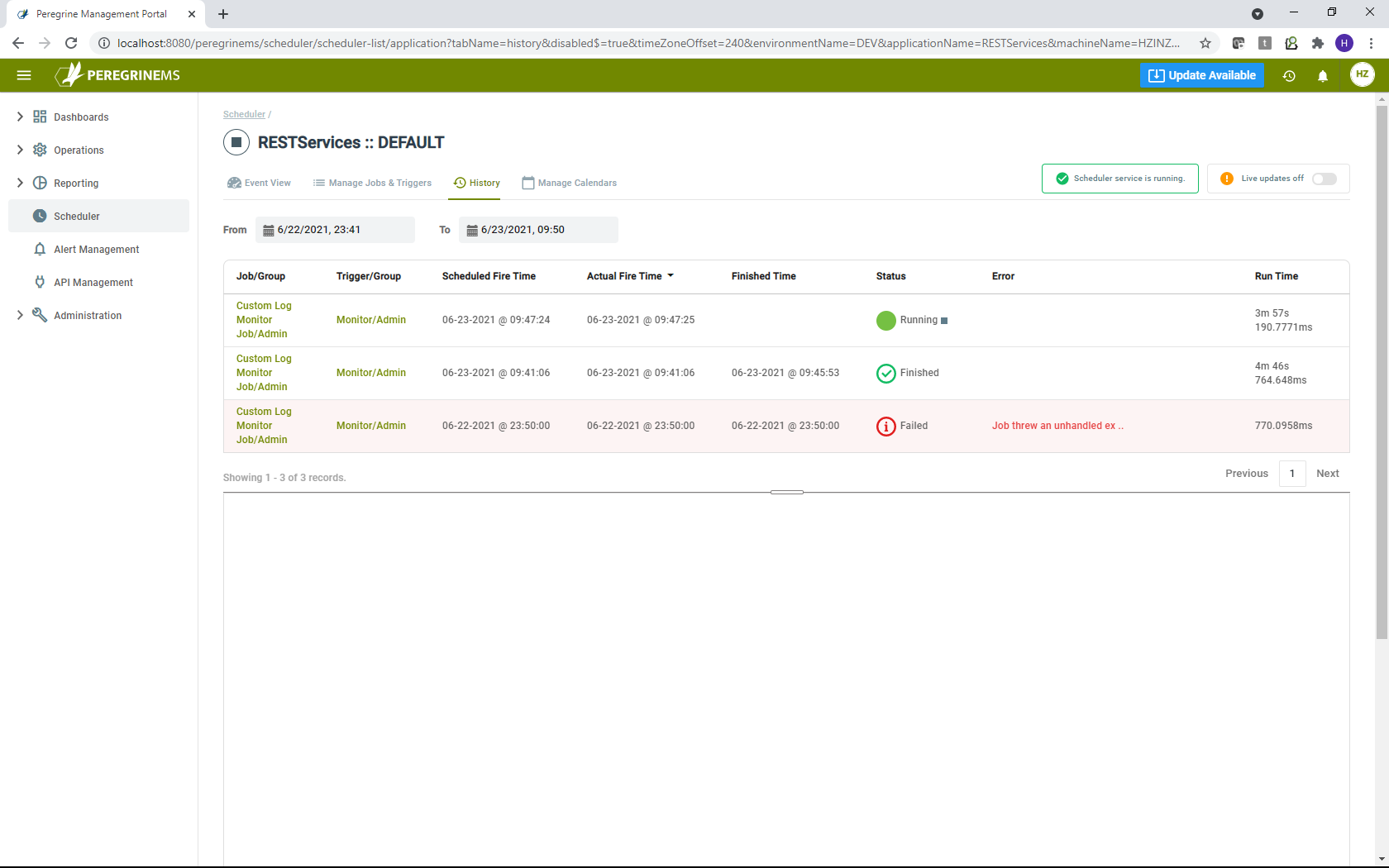


Add a trigger for the job as below. Note you can skip the calendar step. Adjust the time zone, start time, end time and frequency as needed.

## 

## Verifying Job Execution

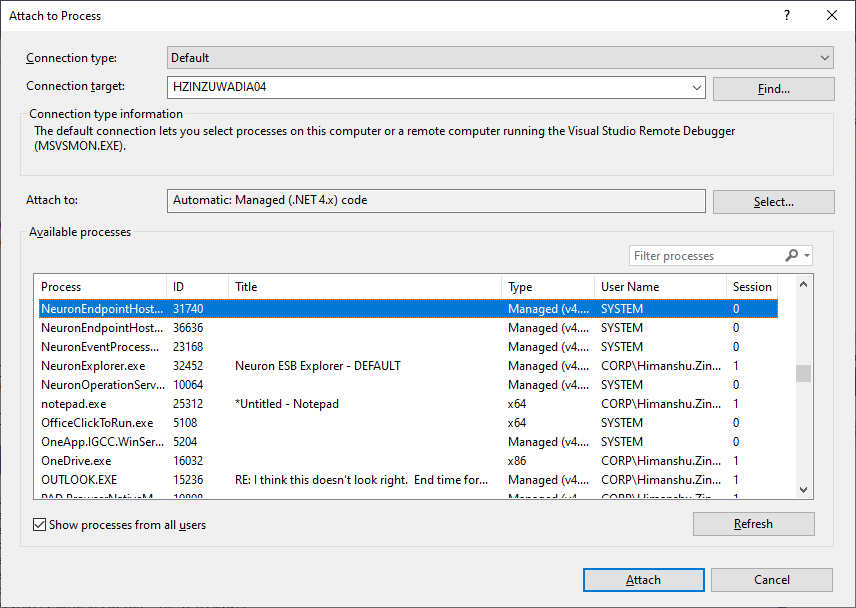
You can verify the job execution by looking at the Job History view. Clicking on a History Record shows details in the pane below. You can see job executions for successful completion, current execution and ones that threw exception.



## 

## Debugging

You can debug your custom job by placing a breakpoint near start of Execute method and attaching Visual Studio debugger to the Neuron ESB Scheduler Host.



You will need to find the Process Id of the Neuron Scheduler Endpoint host from the Neuron Explorer Endpoint Health as shown below.

