Article
Tani Frankel · May 6, 2020 2m read

Open Exchange

SystemPerformance Utility (pka pButtons) API (and REST API) [and sample UI]

While reviewing our documentation for our ^pButtons (in IRIS renamed as ^SystemPerformance) performance monitoring utility, a customer told me: "I understand all of this, but I wish it could be simpler… easier to define profiles, manage them etc.".

After this session I thought it would be a nice exercise to try and provide some easier human interface for this.

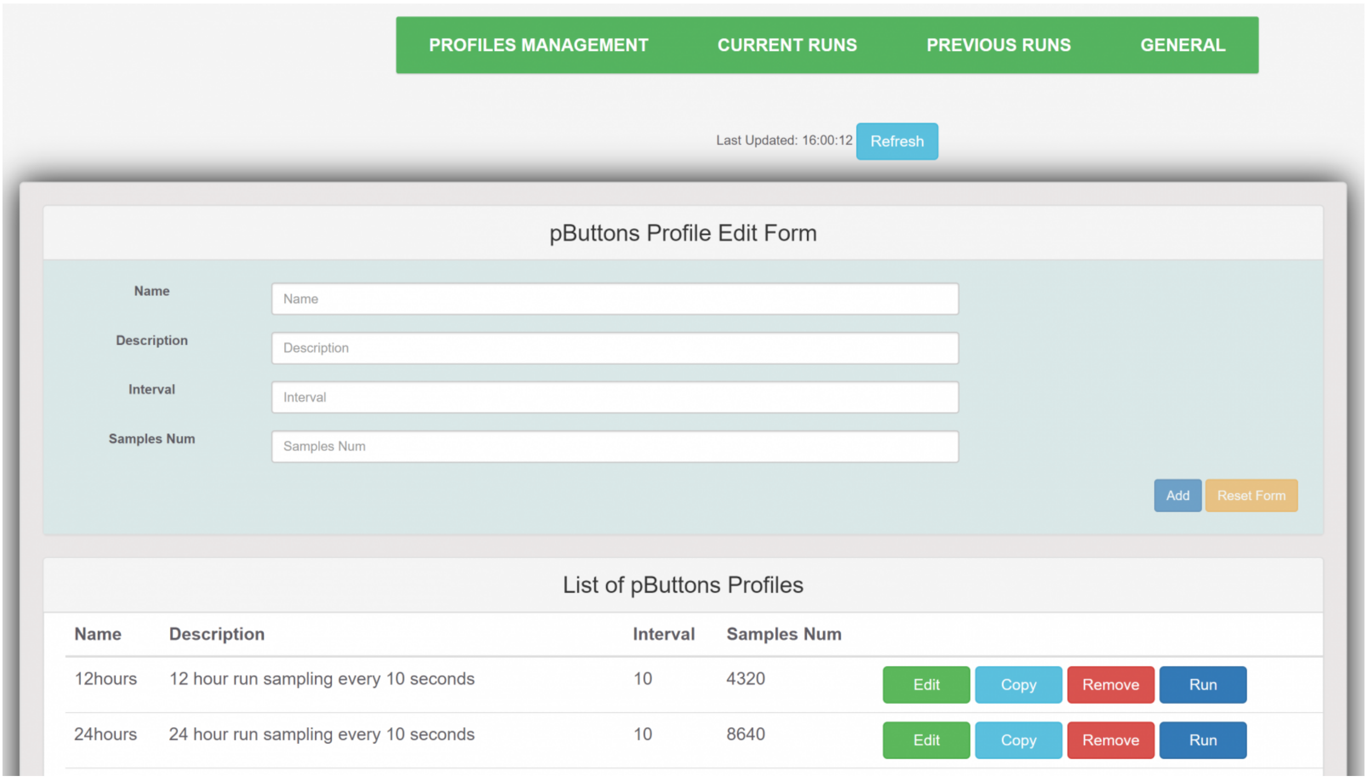
The first step in this was to wrap a class-based API to the existing pButtons routine.

I was also able to add some more "features" like showing what profiles are currently running, their time remaining to run, previously running processes and more.

The next step was to add on top of this API, a REST API class.

With this artifact (a pButtons REST API) in hand, one can go ahead and build a modern UI on top of that.

For example -



Last Updated: 16:00:12

pButtons Profile Edit Form

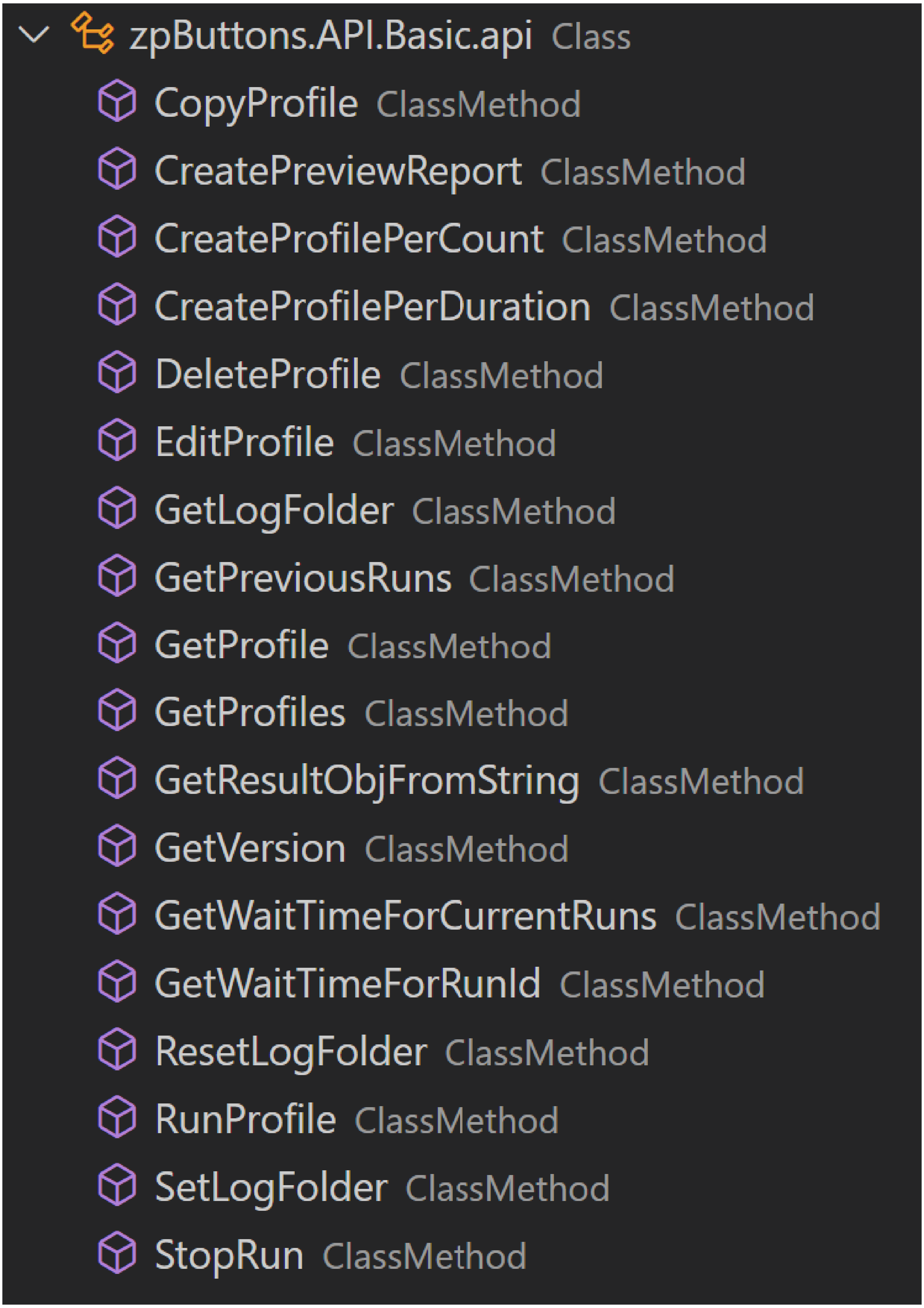
Name	Name
Description	Description
Interval	Interval
Samples Num	Samples Num

List of pButtons Profiles

Name	Description	Interval	Samples Num
12hours	12 hour run sampling every 10 seconds	10	4320
24hours	24 hour run sampling every 10 seconds	10	8640

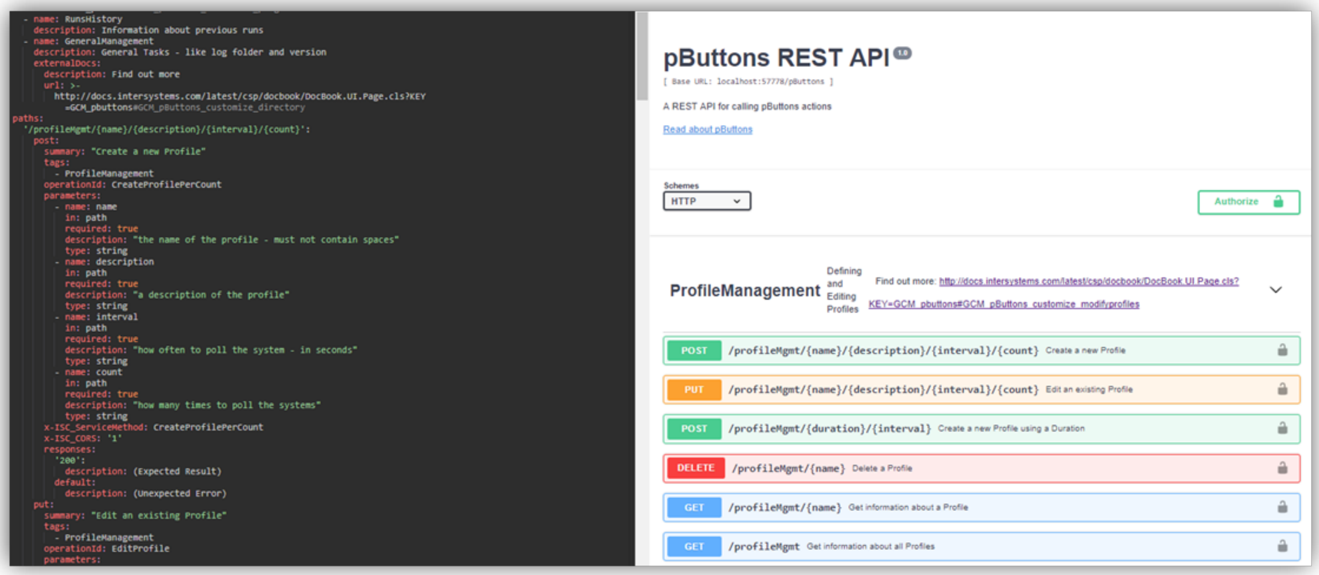
So I'm sharing with you here a few steps of the way:

Two classes which are the "Basic" API –


zpButtons.API.Basic.api Class
CopyProfile ClassMethod
CreatePreviewReport ClassMethod
CreateProfilePerCount ClassMethod
CreateProfilePerDuration ClassMethod
DeleteProfile ClassMethod
EditProfile ClassMethod
GetLogFolder ClassMethod
GetPreviousRuns ClassMethod
GetProfile ClassMethod
GetProfiles ClassMethod
GetResultObjFromString ClassMethod
GetVersion ClassMethod
GetWaitTimeForCurrentRuns ClassMethod
GetWaitTimeForRunId ClassMethod
ResetLogFolder ClassMethod
RunProfile ClassMethod
SetLogFolder ClassMethod
StopRun ClassMethod

As well as the REST API (including some Unit Test classes to test these).

A Swagger JSON for the REST API. To build this I used what was then (back in 2017) a not yet released capability in InterSystems IRIS for REST management. Building on top of the basic Swagger JSON provided by the InterSystems IRIS I added further information.



```yaml
  - name: RunsHistory
    description: Information about previous runs
  - name: GeneralManagement
    description: General Tasks - like log folder and version
    externalDocs:
      description: Find out more
      url: >-
        http://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY
          =GCM_pbuttons#GCM_pButtons_customize_directory
paths:
  '/profileMgmt/{name}/{description}/{interval}/{count}':
    post:
      summary: "Create a new Profile"
      tags:
        - ProfileManagement
      operationId: CreateProfilePerCount
      parameters:
        - name: name
          in: path
          required: true
          description: "the name of the profile - must not contain spaces"
          type: string
        - name: description
          in: path
          required: true
          description: "a description of the profile"
          type: string
        - name: interval
          in: path
          required: true
          description: "how often to poll the system - in seconds"
          type: string
        - name: count
          in: path
          required: true
          description: "how many times to poll the systems"
          type: string
      x-ISC_ServiceMethod: CreateProfilePerCount
      x-ISC_CORS: '1'
      responses:
        '200':
          description: (Expected Result)
        default:
          description: (Unexpected Error)
    put:
      summary: "Edit an existing Profile"
      tags:
        - ProfileManagement
      operationId: EditProfile
      parameters:
```

pButtons REST API 1.0
[Base URL: localhost:57778/pButtons]

A REST API for calling pButtons actions

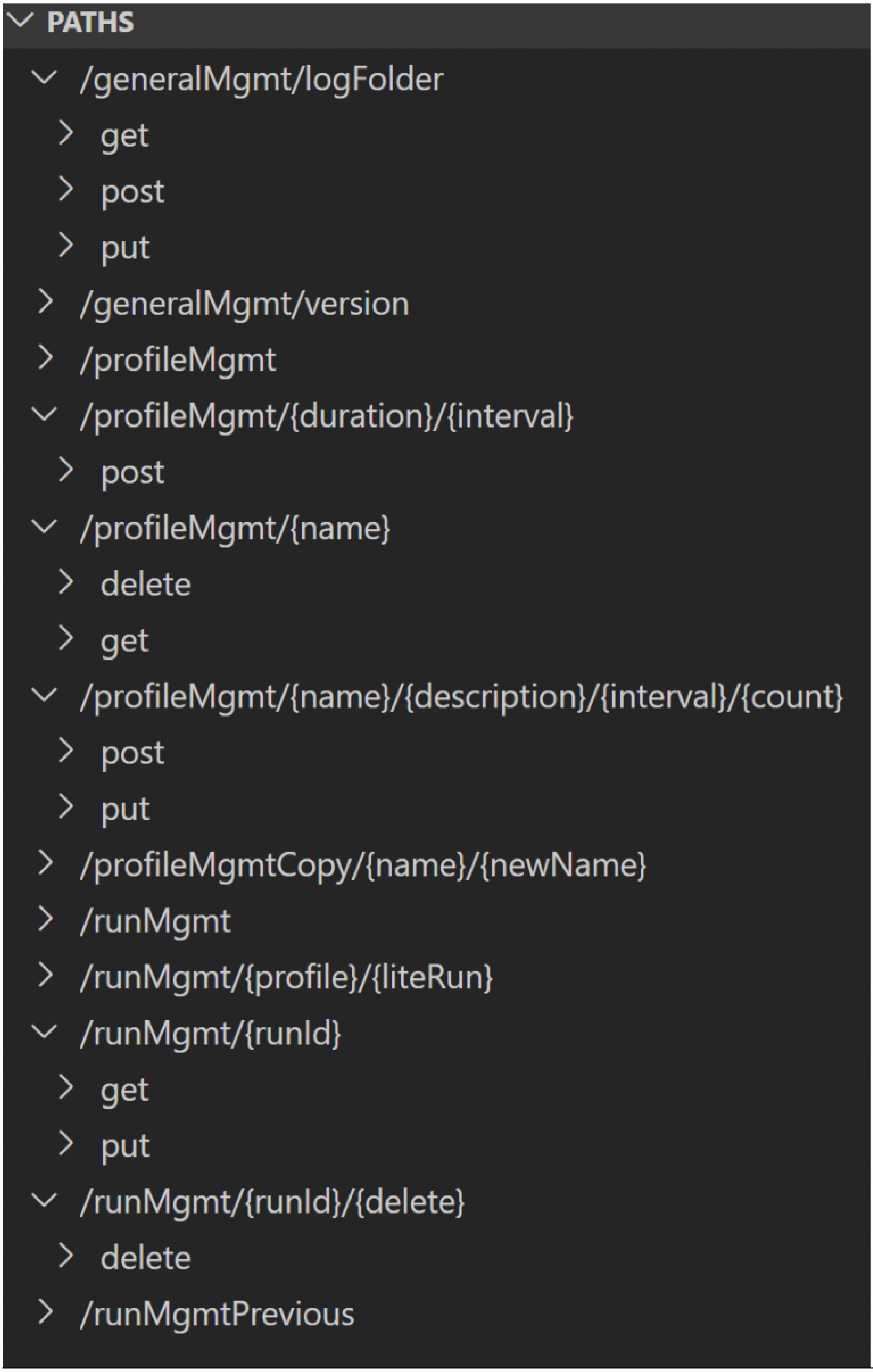
Read about pButtons

Schemes: HTTP

Authorize

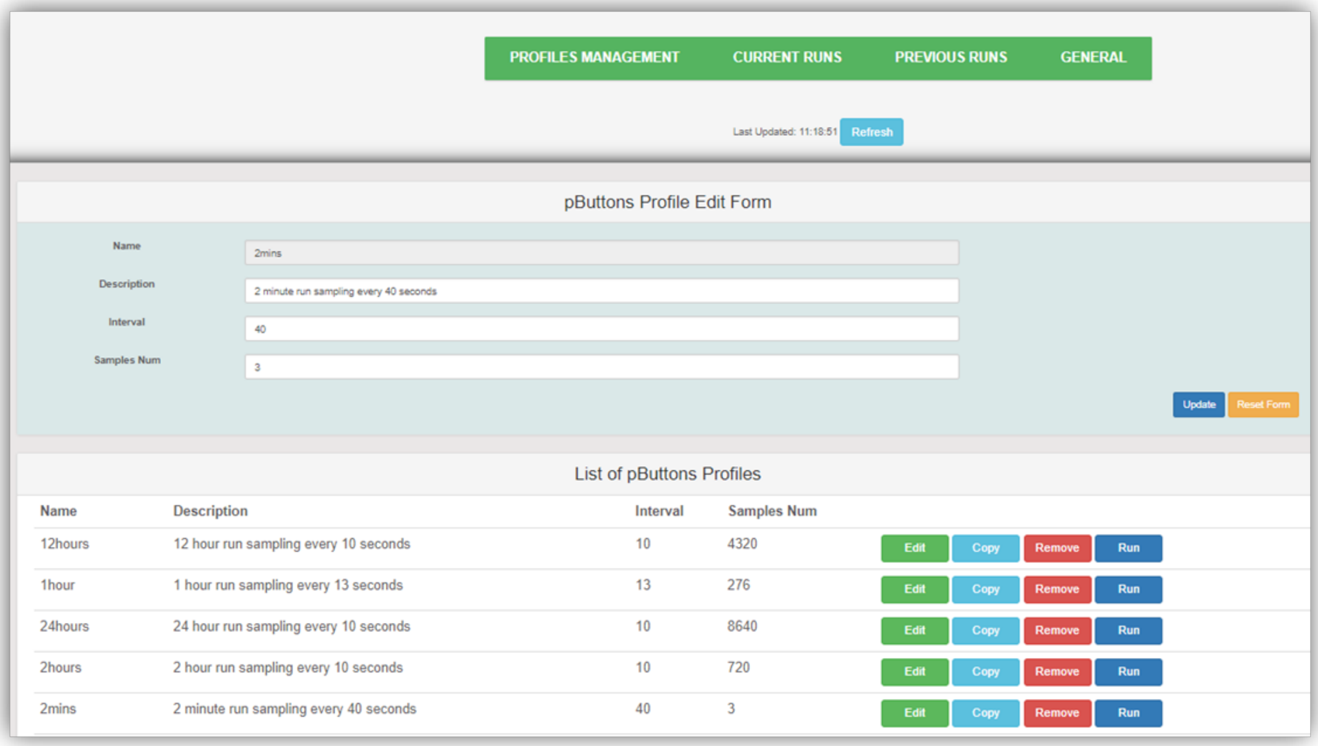
ProfileManagement Defining and Editing Profiles Find out more: http://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=GCM_pbuttons#GCM_pButtons_customize_modifyprofiles

POST	/profileMgmt/{name}/{description}/{interval}/{count}	Create a new Profile
PUT	/profileMgmt/{name}/{description}/{interval}/{count}	Edit an existing Profile
POST	/profileMgmt/{duration}/{interval}	Create a new Profile using a Duration
DELETE	/profileMgmt/{name}	Delete a Profile
GET	/profileMgmt/{name}	Get information about a Profile
GET	/profileMgmt	Get information about all Profiles



```
PATHS
  /generalMgmt/logFolder
    get
    post
    put
  /generalMgmt/version
  /profileMgmt
  /profileMgmt/{duration}/{interval}
    post
  /profileMgmt/{name}
    delete
    get
  /profileMgmt/{name}/{description}/{interval}/{count}
    post
    put
  /profileMgmtCopy/{name}/{newName}
  /runMgmt
  /runMgmt/{profile}/{liteRun}
  /runMgmt/{runId}
    get
    put
  /runMgmt/{runId}/{delete}
    delete
  /runMgmtPrevious
```

As well as a simple angular UI (based on this http://websystique.com/angularjs/angularjs-crud-application-using-ngresource/)



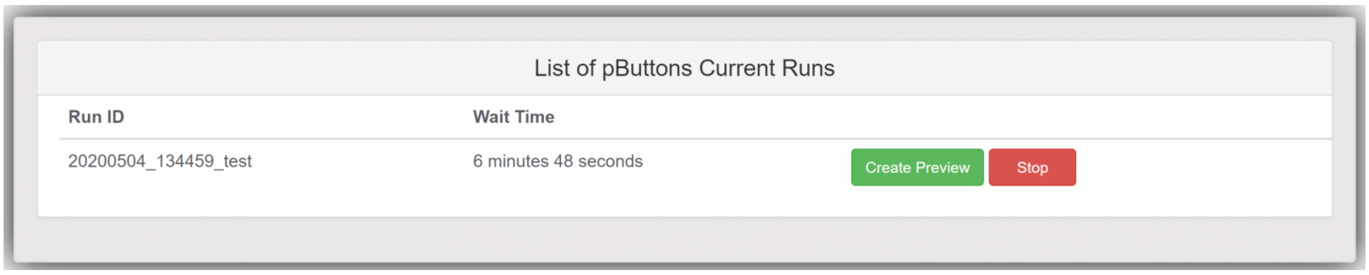
Last Updated: 11:18:51

pButtons Profile Edit Form

Name	2mins
Description	2 minute run sampling every 40 seconds
Interval	40
Samples Num	3

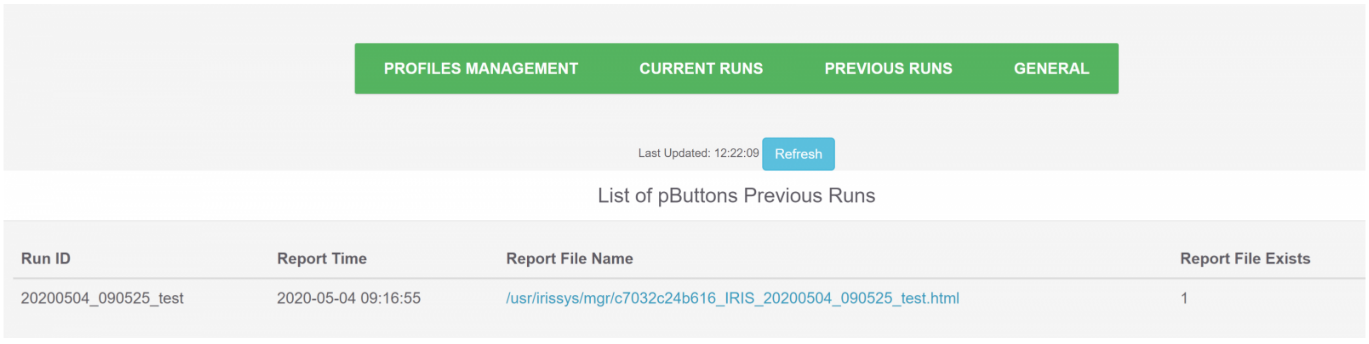
List of pButtons Profiles

Name	Description	Interval	Samples Num
12hours	12 hour run sampling every 10 seconds	10	4320
1hour	1 hour run sampling every 13 seconds	13	276
24hours	24 hour run sampling every 10 seconds	10	8640
2hours	2 hour run sampling every 10 seconds	10	720
2mins	2 minute run sampling every 40 seconds	40	3



List of pButtons Current Runs

Run ID	Wait Time
20200504_134459_test	6 minutes 48 seconds



Last Updated: 12:22:09

List of pButtons Previous Runs

Run ID	Report Time	Report File Name	Report File Exists
20200504_090525_test	2020-05-04 09:16:55	/usr/irissys/mgr/c7032c24b616_IRIS_20200504_090525_test.html	1

A few important notes:

- Most of the "Basic" API methods use documented and supported entry points to the pButtons/SystemPerformance routine. But some methods access the internal structures managed by the pButtons utility. These are NOT documented or supported and these methods could stop working upon upgrade without notice.
- This code by any means is not intended as a "best-practice" example for building a REST-based

Angular app with InterSystems IRIS. The UI part was just given as an example/"teaser" and a sample starting point [It's not complete for example - the "General section" (e.g. a place-holder for Log folder location management) was not implemented; and the Refresh of the contents doesn't work fully across the board and some other "known issues..."]

- The initial code for this was written quite some time ago - so:
    - (a) I came back to this now to make sure it runs with IRIS (and had to make some naming changes).
    - (b) The REST API did not use the spec-first approach - I did initially play with the (back then) IRIS-beta's support for generating a Swagger, and now used that to make this into a spec-first approach.
    - (c) This might not use the latest features available today.
    - (d) I added now also support for this running within a Docker container (with the relevant Dockerfile etc.).
    - (e) I also added support for this being installed as a ZPM package.

#Angular #API #Best Practices #Performance #REST API #Caché #Ensemble #HealthShare #InterSystems IRIS #InterSystems IRIS for Health
Check the related application on InterSystems Open Exchange

Source URL:https://community.intersystems.com/post/systemperformance-utility-pka-pbuttons-api-and-rest-api-and-sample-ui