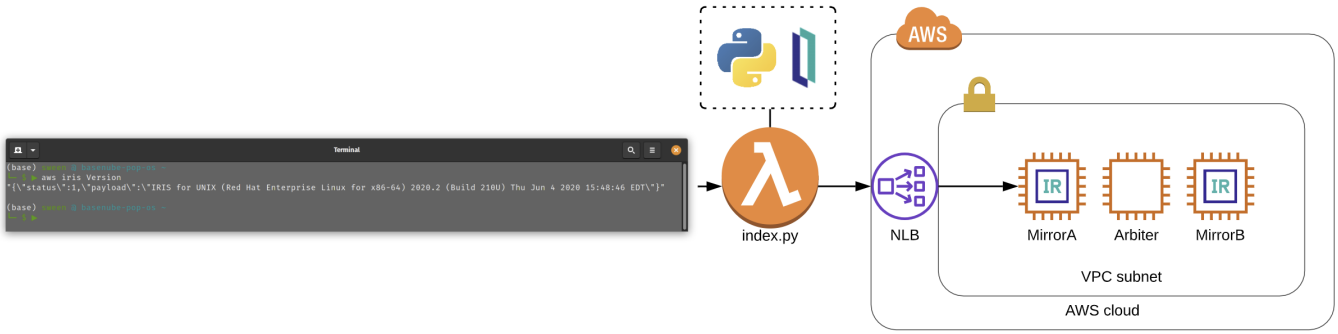
Article
Ron Sweeney · Dec 7, 2020 6m read

IRIS Python Native API in AWS Lambda

If you are looking for a slick way to integrate your IRIS solution in the Amazon Web Services ecosystem, server less application, or boto3 powered python script, using the IRIS Python Native API could be the way to go. You don't have to build out to far with a production implementation until you'll need to reach out and get something or set something in IRIS to make your application do its awesome sauce, so hopefully you will find value in this article and build something that matters or doesn't matter at all to anybody else but you as that is equally important.



If you are looking for some excuses to implement this:

- You need to hit the Pre Token Generation Trigger in Cognito to lookup and stuff the Patient ID context into the token for a SHMAHT on FHIR(R) based solution implementing an OAUTH2 workflow.
- You want to post provision iris tuning settings based on the Instance Type, Node Group, Toaster, or ECS Cluster you fired up to run IRIS on ring zero.
- You want to wow family and friends on Zoom with your management skills of IRIS without your shell ever leaving the AWS cli.

The Point

Here we are going to provision an AWS lambda function that talks to IRIS and provide some examples on how to provision it, and how to interact with it in various capacities in hopes we can all argue about it and **get it published to pip to make things easier.**

In a Hurry????

Check out the Stream

https://www.youtube.com/embed/mA0VzKOYhBk
*[This is an embedded link, but you cannot view embedded content directly on the site because you have declined the cookies necessary to access it. To view embedded content, you would need to accept all cookies in your **Cookies Settings**]*

In All Cases...

In order to participate in the fun, you'll need to cross off a few things for your flight plan.

Networking

You have IRIS running on anything you want, running in an AWS VPC with exactly two subnets and a security group that allows access to the superserver running IRIS... we use 1972 for nostalgic reasons and for the simple fact InterSystems took the time to register that port with IANA, and if you add the new port in /etc/services without doing so, you will suffer the same consequences as if you tore the tag off a new mattress. In our case, its a mirror set of ec2 instances with proper health checks around an AWS v2 network load balancer.

Imported Example Class

Somehow, someway, you have managed to create and import the a class in the root of this repository into the %SYS namespace on your IRIS instance. The below is an example class that drives the output above. If you are wondering why we need a class to import here, please see the note below where the recommended approach is to provision some wrapper classes for use through python.

> Note from the Docs:
> Although these methods can also be used with InterSystems classes defined in the Class Library, best practice is to call them indirectly, from within a user defined class or routine. Many class methods return only a status code, passing the actual results back in an argument (which cannot be accessed by the Native API). System defined functions (listed under ObjectScript Functions in the ObjectScript Reference) cannot be called directly.

Example Class:

```
Class ZDEMO.IRIS.Lambda.Operations Extends %Persistent
{
  ClassMethod Version() As %String
  {
   Set tSC = 0

   Set tVersion = $ZV
   if ( tVersion '="" ) { set tSC = $$$OK }

   Set jsonret = {}
   Set jsonret.status = tSC
   Set jsonret.payload = tVersion

   Quit jsonret.%ToJSON()
  }
}
```

Note above here that I have decided to work based off convention here and always return a JSON object as the response, that also allows me to return the status and possibly fill the gap from returning some things by reference.

AWS Access

Get some IAM access keys that will allow you to provision and invoke the Lambda Function we are going to change the world with.

Pre-Flight check:

```
IRIS            [ $$$OK ]
VPC             [ $$$OK ]
Subnets         [ $$$OK ]
Security Group  [ $$$OK ]
IAM Access      [ $$$OK ]
Imported Class  [ $$$OK ]
```

$$$OK, **Lesgo**.

Packing up the IRIS Native Python Api for use in Lambda Function

This is the part that would be fantastic if it were a pip package, especially if only for Linux, as AWS Lambda functions execute on Linux boxen. At the time of this commit, the api is not available via pip, but we are resourceful and can roll our own.

```
mkdir iris_native_lambda
cd iris_native_lambda
wget https://github.com/intersystems/quickstarts-python/raw/master/Solutions/nativeAPI_wheel/irisnative-1.0.0-cp34-abi3-linux_x86_64.whl
unzip nativeAPI_wheel/irisnative-1.0.0-cp34-abi3-linux_x86_64.whl
```

Create connection.config

Example:
connection.config

Create your handler, index.py or use the one in the examples folder, in the demo github repository. Note the demo version uses both environment variables and the use of an external file for the IRIS connectivity information.

Example:
index.py

Now zip it up for use:

```
zip -r9 ../iris_native_lambda.zip *
```

Create an S3 bucket, and upload the function zip to it.

```
cd ..
aws s3 mb s3://iris-native-bucket
s3 sync iris_native_lambda.zip s3://iris-native-bucket
```

This concludes the packaging of the api and handler for use as an AWS Lambda Function.

Now, either click the console to death to create the function, or use something like Cloudformation to get the job done:

```
  IRISAPIFunction:
    Type: "AWS::Lambda::Function"
```

```
    DependsOn:
      - IRISSG
      - VPC
    Properties:
      Environment:
        Variables:
          IRISHOST: "172.31.0.10"
          IRISPORT: "1972"
          NAMESPACE: "%SYS"
          USERNAME: "intersystems"
          PASSWORD: "lovetheyneighbor"
      Code:
        S3Bucket: iris-native-bucket
        S3Key: iris_native_lambda.zip
      Description: "IRIS Native Python API Function"
      FunctionName: iris-native-lambda
      Handler: "index.lambda_handler"
      MemorySize: 128
      Role: "arn:aws:iam::8675309:role/BeKindtoOneAnother"
      Runtime: "python3.7"
      Timeout: 30
      VpcConfig:
        SubnetIds:
          - !GetAtt
            - SubnetPrivate1
            - Outputs.SubnetId
          - !GetAtt
            - SubnetPrivate2
            - Outputs.SubnetId
        SecurityGroupIds:
          - !Ref IRISSG
```

Now that was A LOT, but now you can get all crazy and call IRIS through the the lambda function with Python and change the world.

Execution!

The way the above is implemented, the function is expected to be passed in an event object that is somewhat structured for re-use, you can see the idea in the example event object below:

```
    {
      "method": "Version",
      # important, if method requires no args, enforce "none"
      "args": "none"
      # example method with args, comma seperated
      # "args": "thing1, thing2"
    }
```

now you can if you have a toleration for command line examples, take a peak at the execution below using the AWS CLI:

```
(base) sween @ basenube-pop-os ~/Desktop/BASENUBE
?? $ &#x25b6; aws lambda invoke --function-name iris-native-lambda --payload '{"metho
```

```
d":"Version","args":"none"}' --invocation-type RequestResponse --cli-binary-
format raw-in-base64-out --region us-east-2 --profile default /dev/stdout
{{\"status\":1,\"payload\":\"IRIS for UNIX (Red Hat Enterprise Linux for x86-64) 2020
.2 (Build 210U) Thu Jun 4 2020 15:48:46 EDT\"}"
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
```

Now if we take it step further, the AWS CLI supports aliases, so create yourself one and you can play games with total integration into your aws command coolness. Here is an example cli alias:

```
?? $ &#x25b6; cat ~/.aws/cli/alias
[toplevel]
whoami = sts get-caller-identity

iris =
  !f() {
   aws lambda invoke \
     --function-name iris-native-lambda \
     --payload \
     "{\"method\":\""${1}"\",\"args\":\"none\"}" \
     --invocation-type RequestResponse \
     --log-type None \
     --cli-binary-format raw-in-base64-out \
     gar.json > /dev/null
     cat gar.json
     echo
     echo
  }; f
```

....and now you can just do...

Stay safe!-technical arguments welcome!

#integration-required #Python #InterSystems IRIS #InterSystems IRIS for Health

---

Source URL:https://community.intersystems.com/post/iris-python-native-api-aws-lambda