Article
Tomohiro Iwamoto · Dec 24, 2020 6m read

How to encrypt (HTTPS) communications with the InterSystems IRIS management portal while running on AWS

About this article:

In InterSystems IRIS, the default form of access to the management portal is HTTP, which means that if the client is in the office and the server is in the cloud, many clients probably desire to encrypt their traffic in some way.

Thus, we would like to show you some ways to encrypt your traffic to and from the IRIS management portal (or various REST services) running on AWS.

> This article uses the IRIS built-in apache server for access. It should not be used for benchmarking purposes or as a method of access from production environments applications.
> It is designed to encrypt access for development, operation verification, and quick management and with a small number of people.

The best solution would be to prepare a domain name and an SSL server certificate issued by a central certification authority. However, in the case of the aforementioned applications, it would not be easy in terms of cost.

Therefore, the following certificates are assumed to be used:

- Self-signed (a Japanese so-called “ ore ore ” certificate)
- A certificate issued by a self-built certification authority (a Japanese so-called “ ore ore ” Certification Authority)

We additionally assume the following running environment:

The PC environment	
O/S	Windows 10
Browser	Chrome/FireFox/Edge
IDE	vscode+ObjectScript Extension
Unused port number in the local PC	8888
The secret key of the key pair used during the creation of the EC2 Instance	aws-secret.pem
AWS Environment	
IRIS host's public hostname	ec2-54-250-169-xxx.ap-northeast-1.compute.amazonaws.com
IRIS webserver port number	52773
O/S	Ubuntu 18.04LTS
O/S User name	ubuntu

Direct Access

1) Using Port Forwarding

This is the easiest way.

In the security group, the port for SSH (22) must be allowed inbound from the Internet.

```
C:\Users\xxxx>ssh -i aws-secret.pem -L 8888:localhost:52773 ubuntu@ec2-54-250-169-xxx
.ap-northeast-1.compute.amazonaws.com
```

While running this command from the PC, you can access the IRIS host through the link below.

http://localhost:8888/csp/sys/%25CSP.Portal.Home.zen

Since you are logged in with ssh, you can use it for terminal operations such as those performed during the development process (starting and stopping IRIS, starting an IRIS session, etc.).

This method is also useful for the super-server port (51773), so it can encrypt the communication with Studio.

Note: For Windows, if you do not place the private key for ssh (aws-secret.pem) in %USERPROFILE%\, you will get an error.

```
C:\Users\xxxx>dir %USERPROFILE%\aws-secret.pem
2020/07/14  17:10             1,692 aws-secret.pem
           1 File(s)          1,692 bytes
           0 Dir(s)  100,576,694,272 bytes free
```

The vscode settings (settings.json) should look like the one below.

```
{
    "objectscript.conn": {
        "host": "localhost",
        "https": false,
        "port": 8888,
        "ns": "USER",
        "username": "xxx",
        "password": "xxx",
        "active": true
    }
}
```

2) Using a Reverse Proxy with SSL configuration

In this case, you deploy a self-certified apache or Nginx with a reverse proxy configuration and access the IRIS host through it.

In the security group, a port for HTTPS (443) must be allowed inbound from the Internet.

A script to configure apache and Nginx is available in the link. This will allow you to access the IRIS host from your browser through the link below.

https://ec2-54-250-169-xxx.ap-northeast-1.compute.amazonaws.com/csp/sys/...

The vscode settings (settings.json) should look like the following:

```
{
    "objectscript.conn": {
        "host": "ec2-54-250-169-xxx.ap-northeast-1.compute.amazonaws.com",
```

```
        "https": true,
        "port": 443,
        "ns": "USER",
        "username": "xxx",
        "password": "xxx",
        "active": true
    }
}
```

Via Bastion host

Suppose you are uncomfortable about letting user data, EC2 instance of SSH (including code) or HTTPS ports be published on the Internet, despite being for verification purposes. In that case, you can use a Bastion Host.

The security group must allow inbound TCP traffic between the Bastion Host and the IRIS host.

It is assumed that the execution environment is as follow:

AWS Environment	
IRIS host's public hostname	none
Internal IP address of the IRIS host	10.0.1.81
Public hostname of the Bastion host	ec2-54-250-169-yyy.ap-northeast-1.compute.amazonaws.com

1) Using Port Forwarding

In the security group, the port for SSH (22) must be allowed inbound to the Internet.

The following commands are executed against the Bastion host:

```
C:\Users\xxxx>ssh -i aws-secret.pem -L 8888:10.0.1.81:52773 ubuntu@ec2-54-250-169-yyy
.ap-northeast-1.compute.amazonaws.com
```

Ditto.

2) Using a Reverse Proxy with SSL configuration

In the security group, the HTTPS port (443) must be allowed inbound to the Internet.

Do the same task (deploy apache/Nginx) on the Bastion host.

Change the destination URL to the internal IP address of the IRIS host: 10.0.1.81.

```
ProxyRequests Off
ProxyPass / http://10.0.1.81:52773/
ProxyPassReverse / http://10.0.1.81:52773/
```

Ditto. (Except that now using the bastion hostname as URL)

Via AWS/ALB

It is not something people typically do, but you can apply a self-certification to AWS/ALB. In this case, the ALB will be SSL-terminated, saving you the trouble of preparing a separate SSL-enabled apache.

Since creating an ALB requires at least two AZs, I have used mirrored DMs created by ICM (InterSystems Cloud Manager).

(For more information about ICM, see How to Configure an IRIS Cluster with ICM)

default.json (excerpt)

```
{
  "Zone": "ap-northeast-1a,ap-northeast-1c",
  "Mirror": "true",
}
```

definitions.json

```
[
    {
        "Role": "DM",
        "Count": "2",
        "MirrorMap": "primary,backup",
        "ZoneMap": "0,1"
    }
]
```

Use the certificate files you created in setup.sh.

Import these files to ACM (AWS Certificate Manager).
Certificate: contents of the server.crt
Certificate of the Private key: contents of the server.key
Certificate Chain: contents of the inca.pem
Tip) If you have access to awscli, uncomment the last line of setup.sh, then it will be registered automatically.

Create a new ALB with the following settings:

Step 1: Configure Load Balancer
Name: anything
Scheme: For Internet
IP address type: ipv4
Listeners: https (port: 443)
Availability Zones: ap-northeast-1a,ap-northeast-1c

Step 2: Configuration of Security Settings
Selecting a default certificate: Selecting a certificate from ACM
Certificate Name: (Select the certificate you have just imported into ACM)

Step 3: Configure Security Groups
Security group settings: Create a new security group →that Allows only HTTPS (port:443).

Step 4: Configure Routing
Target group: New target group
Name: anything
Target Type: Instance
Protocol: HTTP
Port: 52773

Health Checks
Protocol: http
Path: /csp/bin/Mirrorstatus.cxw
Health Check Advanced Settings
Port: Overwrite →52773

Step 5: Register Targets
"Add to registered" the EC2 Instance you've just provisioned.

After the ALB status become active, you can use HTTPS with DNS name of the ALB.

> https://[ALB DNS Name]/csp/sys/exp/%25CSP.UI.Portal.SQL.Home.zen

#AWS #Development Environment #Management Portal #Security #InterSystems IRIS #InterSystems IRIS for Health

---

Source URL:https://community.intersystems.com/post/how-encrypt-https-communications-intersystems-iris-management-portal-while-running-aws