Article
Yuri Marx · Mar 30, 2022 9m read

Open Exchange

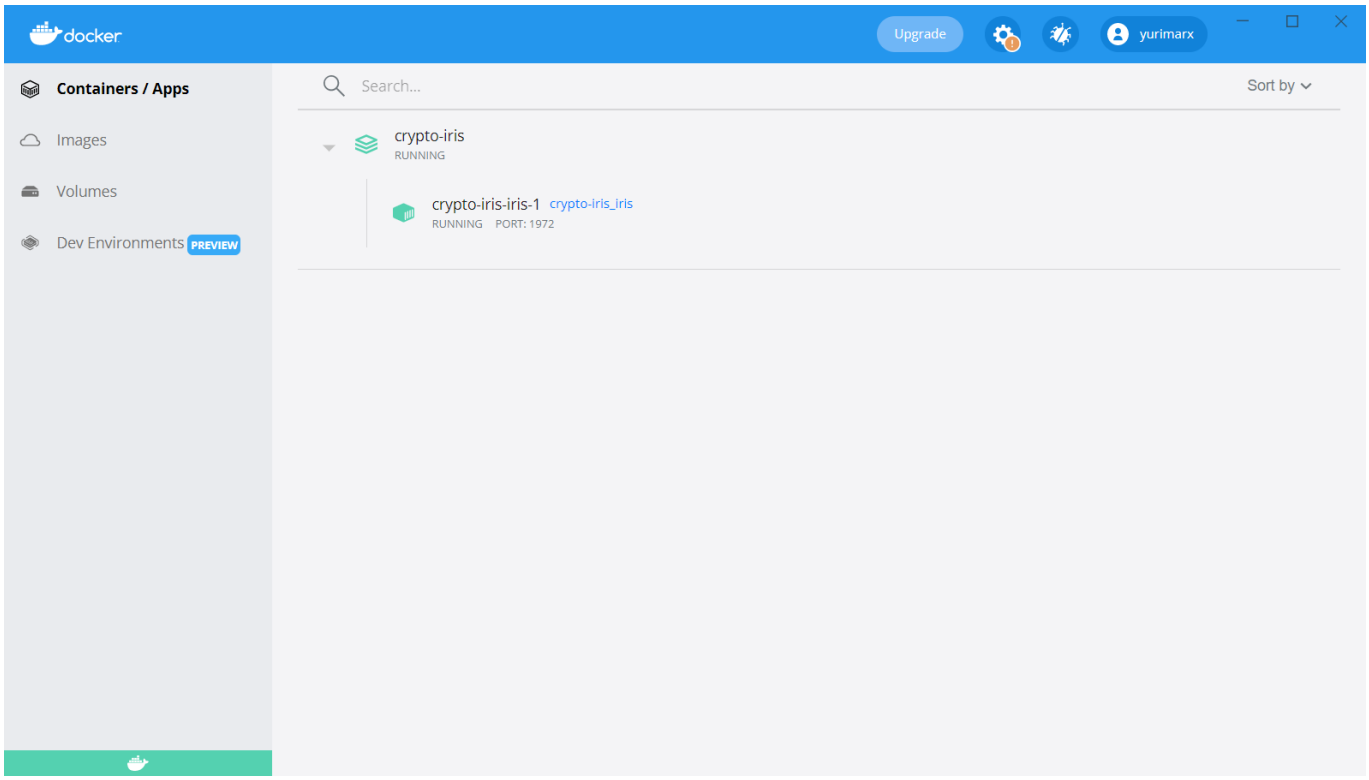
3DES support

There are several ways of classifying cryptographic algorithms: **1) Secret Key Cryptography (SKC)** - Uses a single key for both encryption and decryption. It is also called symmetric encryption. Primarily, it was used for privacy and confidentiality; **2) Public Key Cryptography (PKC)** - Uses one key for encryption and another one for decryption. It is also called asymmetric encryption. Initially, it was utilised for authentication, non-repudiation, and key exchange; **3) Hash Functions** - Uses a mathematical transformation to irreversibly "encrypt" information, providing a digital fingerprint. Originally, it was employed for message integrity. The InterSystems IRIS supports encryption algorithms in all the above-mentioned categories. However, the 3DES (Triple DES) algorithm, many popular and based on SKC, is not supported by the %SYSTEM.Encryption class. The IRIS support for Embedded Python allows using the Python language to support 3DES through the Python package pyDes (https://pypi.org/project/pyDes/). In this article, we will demonstrate to you how it works.

Get the 3DES sample application

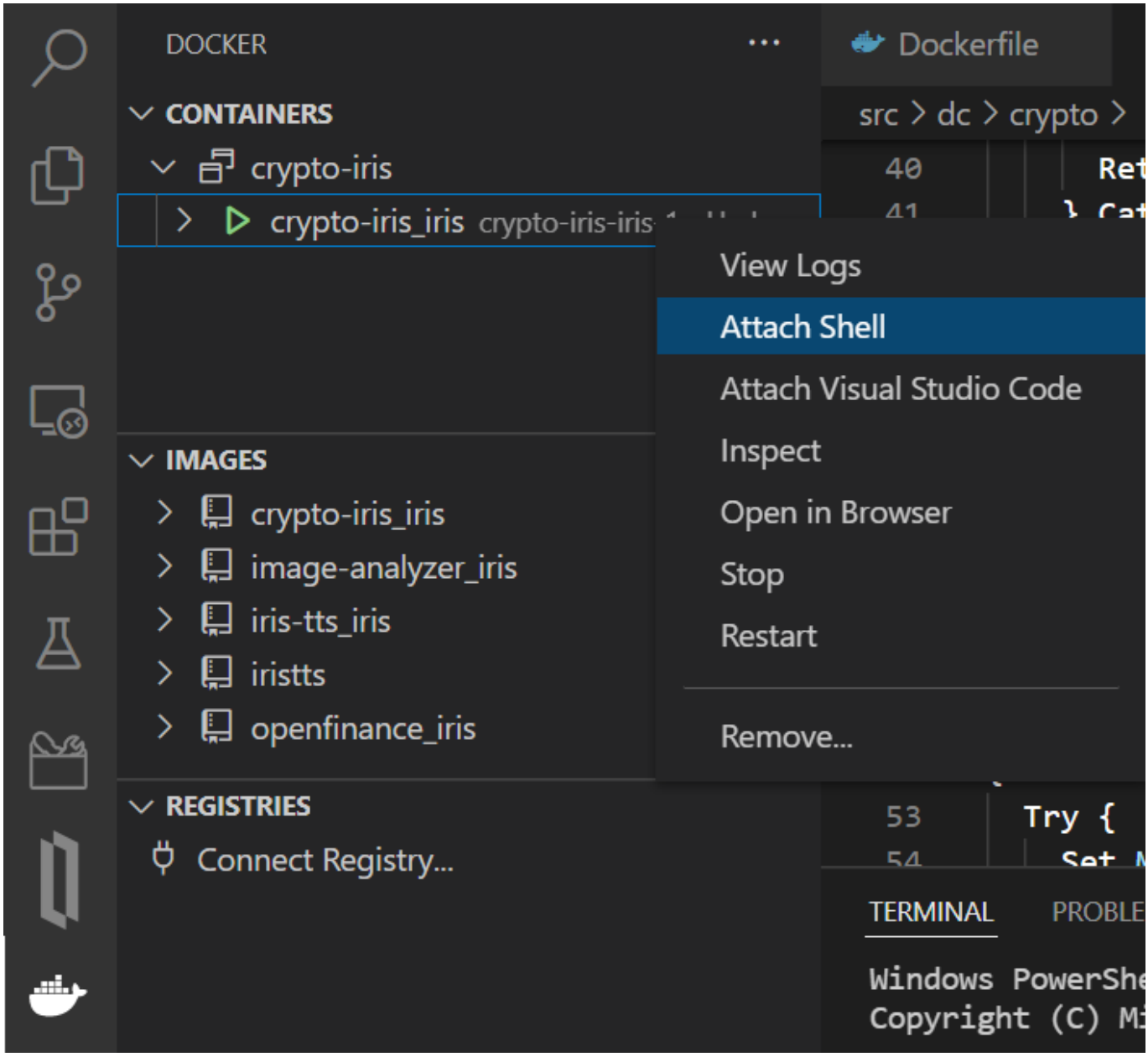
To get the sample and run it, follow these steps:

1. Go to the https://openexchange.intersystems.com/package/crypto-iris and click Download to go to the git repository.
2. Clone the project: git clone https://github.com/yurimarx/crypto-iris.git.
3. Go to the project folder crypto-iris.
4. Do the build: docker-compose build.
5. Execute the containers: docker-compose up -d.
6. Check in your docker desktop with the instances if everything is ok:



Try to encrypt some texts using IRIS Terminal

1. Attach shell to crypto-iris docker instance:



2. In terminal execute iris session iris:

3. Change to IRISAPP namespace:

```
TERMINAL   PROBLEMS   OUTPUT   DEBUG CONSOLE                                   powershell
> Executing task: docker exec -it d2b5a51b9a2504d54b7706aa8a9e2d8e42e60353577a769a9693b9e   Shell: crypto-iris-iris-1  Task
9db073ea8 bash <

irisowner@d2b5a51b9a25:/opt/irisbuild$ iris session iris

Node: d2b5a51b9a25, Instance: IRIS

USER>zn "IRISAPP"

IRISAPP>
```

4. Encrypt and Decrypt a message:

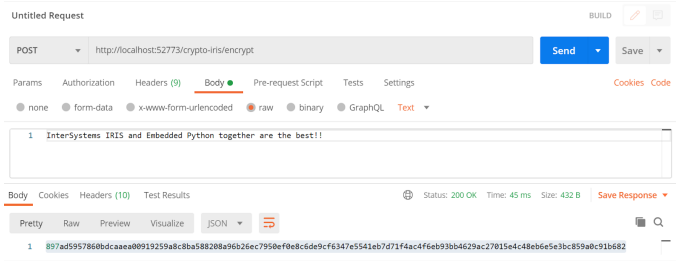
```
TERMINAL   PROBLEMS   OUTPUT   DEBUG CONSOLE                                   powershell
                                                                               Shell: crypto-iris-iris-1  Task
IRISAPP>

IRISAPP>

IRISAPP>write ##class(dc.crypto.CryptoService).Do3DESEncryptation("IRIS Data Platform")
9ebf9fa3a25bdeead6e1f598381991c2dcd1834b0f3649c0
IRISAPP>

IRISAPP>write ##class(dc.crypto.CryptoService).Do3DESDecryptation("9ebf9fa3a25bdeead6e1f5
98381991c2dcd1834b0f3649c0")
IRIS Data Platform
IRISAPP>
```

Try to encrypt some texts using ObjectScript CryptoService class

1. From any ObjectScript class call ##class(dc.crypto.CryptoService).Do3DESEncryptation("YOURMESSAGE") to encrypt.
2. From any ObjectScript class call ##class(dc.crypto.CryptoService).Do3DESDecryptation("YOURMESSAGE") to decrypt.

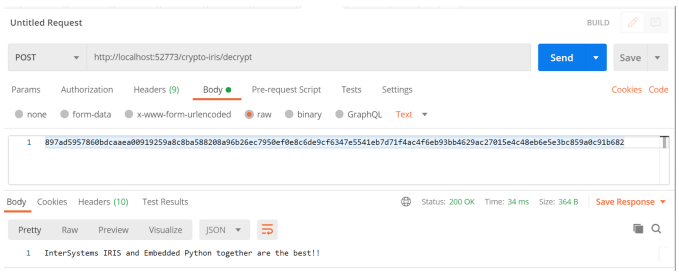
Try to encrypt and decrypt some texts using Postman

1. Go to your Postman (or another similar REST client) and configure the request as shown in this image to encrypt:



- Method: POST
- URL: http://localhost:52773/crypto-iris/encrypt
- Body: raw

2. Click send and get a text encrypted as a binary hex message
3. Go to your Postman (or another similar REST client) and configure the request as indicated in this image to decrypt:



- Method: POST
- URL: http://localhost:52773/crypto-iris/decrypt
- Body: raw

4. Click send and get a text decrypted as a string message

Python and ObjectScript code supporting 3DES

Docker File

The Dockerfile installs python and pyDes package, copies the source file, creates the SECRETKEY which will be used as the private key on encryption and runs IRIS.

Dockerfile

Class dc.crypto.CryptoService

This class implements Encrypt3DES and Decrypt3DES python methods to encrypt and decrypt strings. To do that, it imports triple<u>d</u>es, CBC and PAD<u>P</u>KCS5 to do string encryption and decryption. The binaascii package is imported to return the encrypted string as ASCII hexadecimal string, resolving possible Unicode problems.

CrytoService class

The triple<u>d</u>es python method returns the class to encrypt and decrypt texts. Note that the size of the key was limited to 24 positions and IV to 8.

Class dc.crypto.CryptoRESTApp

This class exposes encryption and decryption services as REST services.

CryptoRESTApp

Modes of operation supported with 3DES

Our sample used CBC mode, but the 3DES also operates the following modes:

ECB:	Electronic Code Book (ECB)	The most basic but also the weakest mode of operation. Each block of plaintext is encrypted independently of any other bloc
CBC:	Cipher-Block Chaining (CBC)	It is a mode of operation where each plaintext block gets XOR with the previous ciphertext block prior to encryption.
CFB:	Cipher FeedBack (CFB)	It is a mode of operation which turns the block cipher into a st cipher. Each byte of plaintext is XOR-ed with a byte taken fro a *keystream*: the result is the ciphertext.
OFB:	Output FeedBack (OFB)	It is another mode that leads to a stream cipher. Each byte of plaintext is XOR-ed with a byte taken from a *keystream*: the re is the ciphertext. The *keystream* is obtained by recursively encrypting the *Initialization Vector*.

CTR:	CounTer Mode (CTR)	This mode turns the block cipher into a stream cipher. Each b of plaintext is XOR-ed with a byte taken from a *keystream*: the result is the ciphertext. The *keystream* is generated by encryp a sequence of *counter blocks* with ECB.

More details on: https://pycryptodome.readthedocs.io/en/latest/src/cipher/des3.html
To get more information about Triple DES visit: https://en.wikipedia.org/wiki/TripleDES

#Security #InterSystems Ideas Portal #InterSystems IRIS
Check the related application on InterSystems Open Exchange

Source URL:https://community.intersystems.com/post/3des-support