Jeff Keir (jeff.keir@mckesson.com)

Azure Storage Service Encryption

Contents

Final Project Summary
Azure Storage Service Encryption

Problem Statement:
According to breachlevelindex.com, as of 10-Feb 2018 over 9.2 billion data records have been lost or stolen since 2013 - a frequency today of 57 records per second[1]. Only 4% of those records were encrypted. Microsoft Azure demonstrates commitment to safeguarding all data by offering a multitude of data protection services. Azure Storage Service Encryption is one such solution. This problem set seeks to unpack how storage service encryption is enabled, how to verify that it is enabled, and demonstrate that data passing through storage service encryption interacts seamlessly with practical applications. In a publication from August 2017 Microsoft implied that blob, file, queue and table services would all be encrypted by SSE[2].

Overview of the Technology:
Azure Storage Service Encryption is a server-side "toggle" impacting the suite of existing services that leverage Azure storage. Once activated, all subsequent storage devices instantiated within the service domain are encrypted. Data written prior to activating the toggle are encrypted after a new read/write operation.

High Level Steps:
1) Instantiate a new storage service and visualize the toggle
2) Query the storage devices to visualize the encryption state
3) Toggle the encryption state
4) Inspect data within a storage device
5) Re-create the messaging service used in Azure Deep Dive homework 8 within a secure storage service and visualize the seamless interoperation

Code Source:
https://github.com/blumu/azure-content/blob/master/articles/event-hubs/event-hubs-archive-python.md

Hardware Used:
Windows 7 64b 16Gb RAM HP Zbook laptop

Software Used:
Azure Cloud Shell (Azure CLI 2.0 (bash))
Python 2.7.5 (https://www.python.org/downloads/)
VS Code 1.18.1 (https://code.visualstudio.com/download)

YouTube Links:
2 Min: https://youtu.be/f_FUk-OmsGE
15 Min: https://youtu.be/l2DjdfS3t9g

[1] http://breachlevelindex.com/

[2] https://azure.microsoft.com/en-us/blog/announcing-default-encryption-for-azure-blobs-files-table-and-queue-storage/

Problem Practicum.

Create new storage and visualize the encryption settings of the Azure Storage Service. Examine the default properties of blob, file, table, and queue storage types. Test the assertions made by Microsoft regarding the property setting, and the services protected by the property. Utilize the storage service in a programmatic application and visualize the encryption states throughout the operation(s).

Create a resource group:

```
az group create --name jnkhw15rg --location southcentralus
```

```
jeff_keir@Azure:~$ az group create --name jnkhw15rg --location southcentralus
{
  "id": "/subscriptions/19ce215b-e250-49ce-a40f-e3efd1217efc/resourceGroups/jnkhw15rg",
  "location": "southcentralus",
  "managedBy": null,
  "name": "jnkhw15rg",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
```

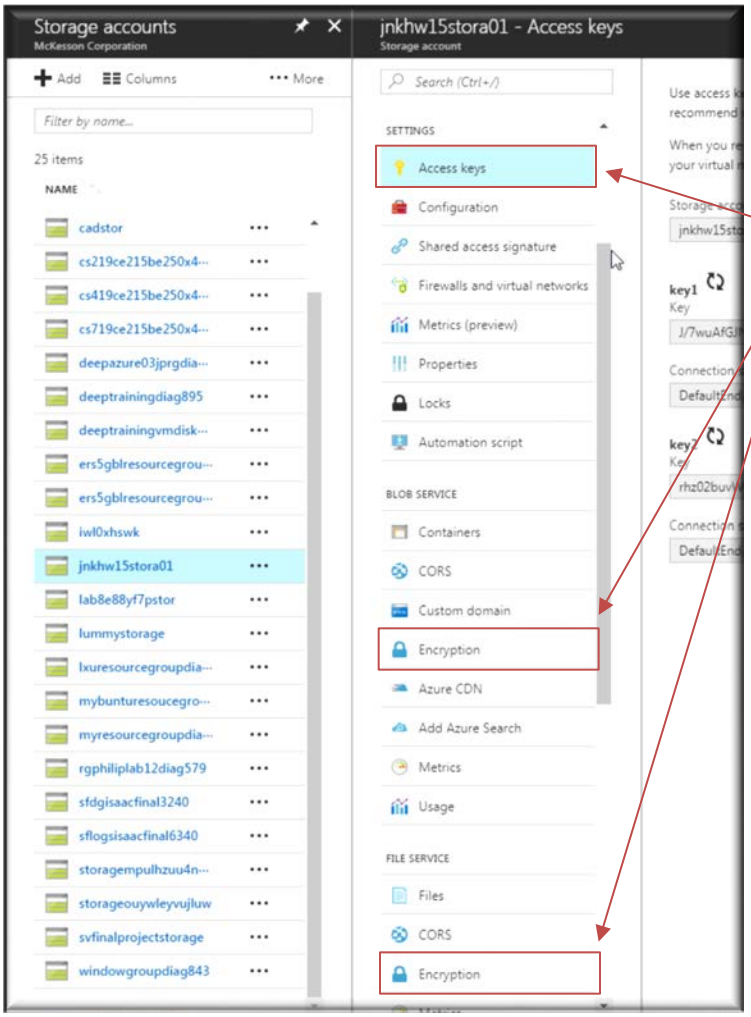
Create a storage account:

```
az storage account create --name jnkhw15stora01 --resource-group jnkhw15rg --
location southcentralus --sku standard_lrs --encryption blob
```

```
jeff_keir@Azure:~$ az storage account create --name jnkhw15stora01 --resource-group jnkhw15rg --location southcentralus --sku standard_lrs --encryption blob
{
  "accessTier": null,
  "creationTime": "2018-02-10T20:59:32.647143+00:00",
  "customDomain": null,
  "enableHttpsTrafficOnly": false,
  "encryption": {
    "keySource": "Microsoft.Storage",
    "keyVaultProperties": null,
    "services": {
      "blob": {
        "enabled": true,
        "lastEnabledTime": "2018-02-10T20:59:32.690738+00:00"
      },
      "file": {
        "enabled": true,
        "lastEnabledTime": "2018-02-10T20:59:32.690738+00:00"
      },
      "queue": null,
      "table": null
    }
  },
  "id": "/subscriptions/19ce215b-e250-49ce-a40f-e3efd1217efc/resourceGroups/jnkhw15rg/providers/Microsoft.Storage/storageAccounts/jnkhw15stora01",
  "identity": null,
  "kind": "Storage",
  "lastGeoFailoverTime": null,
  "location": "southcentralus",
  "name": "jnkhw15stora01",
  "networkRuleSet": {
    "bypass": "AzureServices",
    "defaultAction": "Allow",
    "ipRules": [],
    "virtualNetworkRules": []
  },
  "primaryEndpoints": {
    "blob": "https://jnkhw15stora01.blob.core.windows.net/",
    "file": "https://jnkhw15stora01.file.core.windows.net/",
    "queue": "https://jnkhw15stora01.queue.core.windows.net/",
    "table": "https://jnkhw15stora01.table.core.windows.net/"
  },
  "primaryLocation": "southcentralus",
  "provisioningState": "Succeeded",
  "resourceGroup": "jnkhw15rg",
  "secondaryEndpoints": null,
  "secondaryLocation": null,
  "sku": {
    "capabilities": null,
    "kind": null,
    "locations": null,
    "name": "Standard_LRS",
    "resourceType": null,
    "restrictions": null,
    "tier": "Standard"
  },
  "statusOfPrimary": "available",
  "statusOfSecondary": null,
  "tags": {
    "ccSubOwner": "ete5f57",
    "techOwner": "eqvknmx"
  },
  "type": "Microsoft.Storage/storageAccounts"
}
```

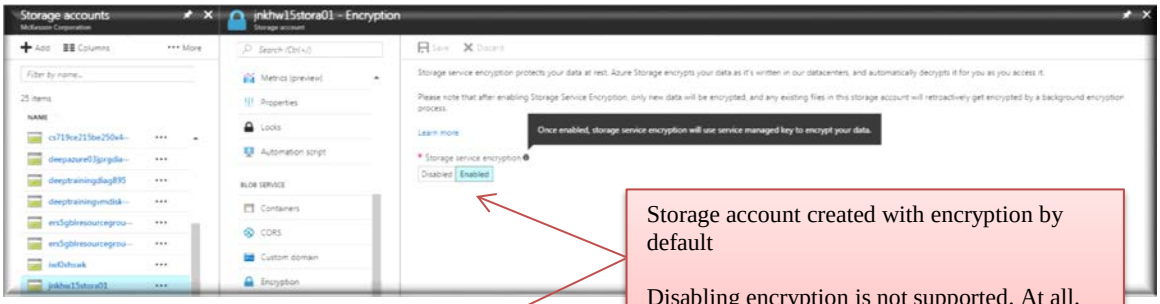
Notice for the blob creation that blob & file are enabled, table and queue are not by default (but can be added)

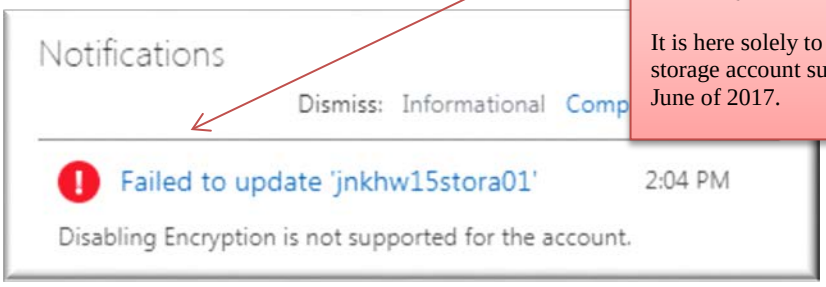
Examine the properties of the storage encryption from the Azure Portal



Encryption keys are automatically generated upon storage creation.

Within the storage account properties blade there are also blob and file service encryption sub-blades visible.



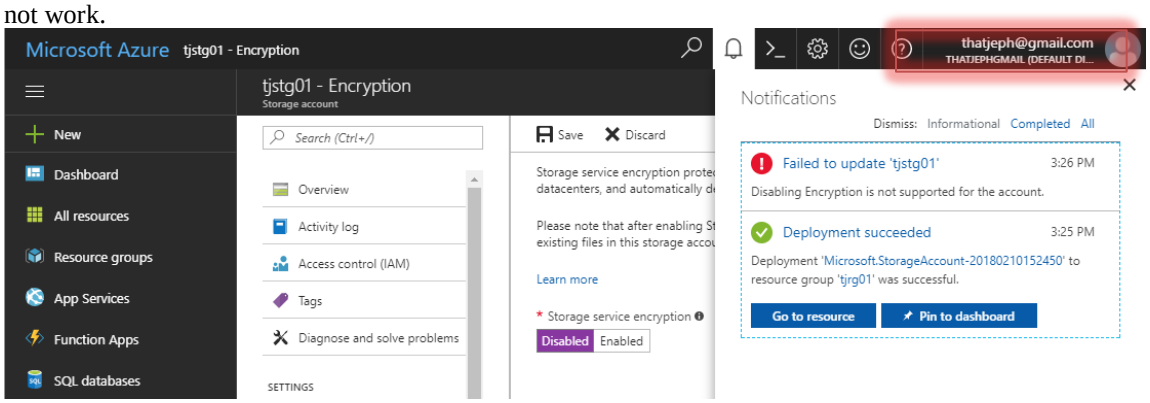


Storage account created with encryption by default

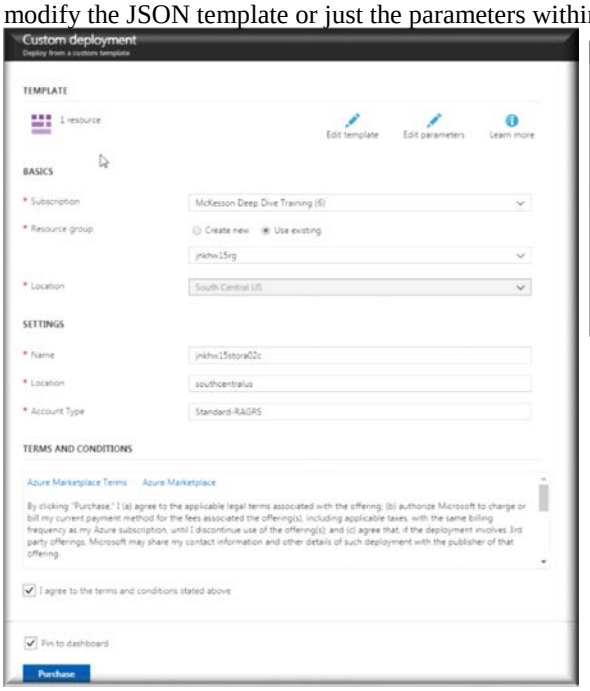
Disabling encryption is not supported. At all. Like, why is this here?

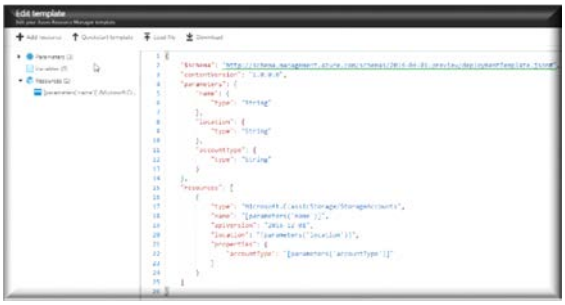
It is here solely to enable encryption on legacy storage account sub-services created prior to June of 2017.

To test whether disabling is a limitation of this trial account, I created a new trial account using a personal credit card. The results were identical to above – disabling RMStorage encryption in the portal simply does not work.

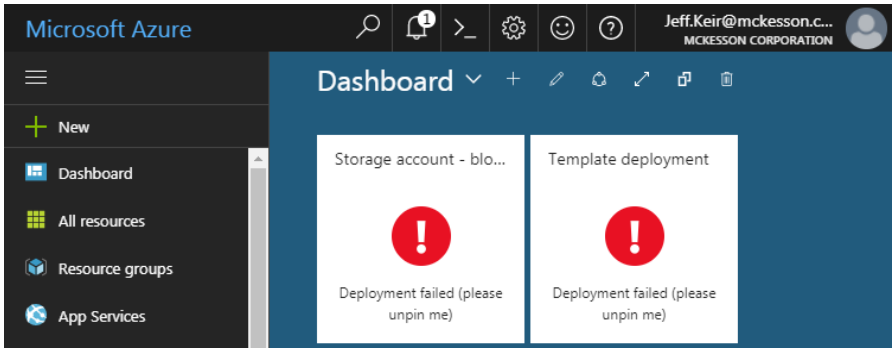


Consider a custom deployment to enable Classic Storage. It is possible to engage a custom deployment and modify the JSON template or just the parameters within the JSON template.

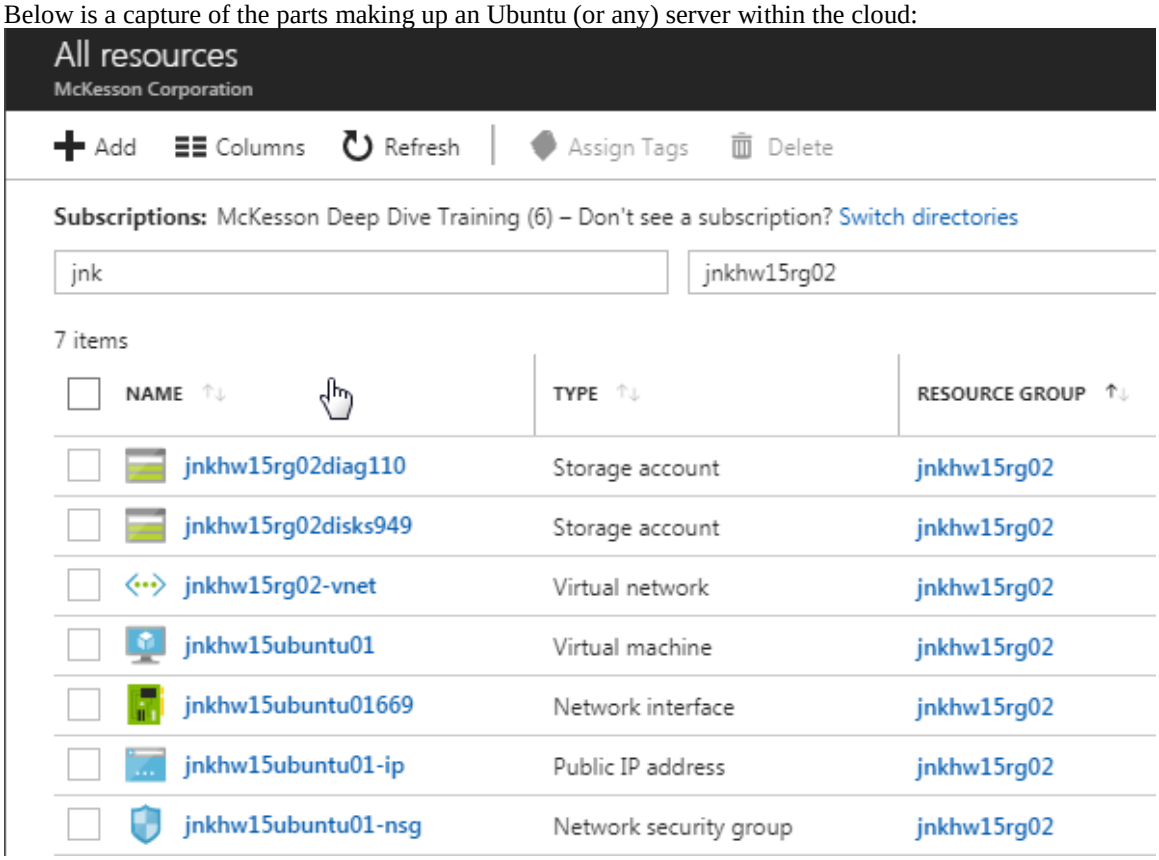




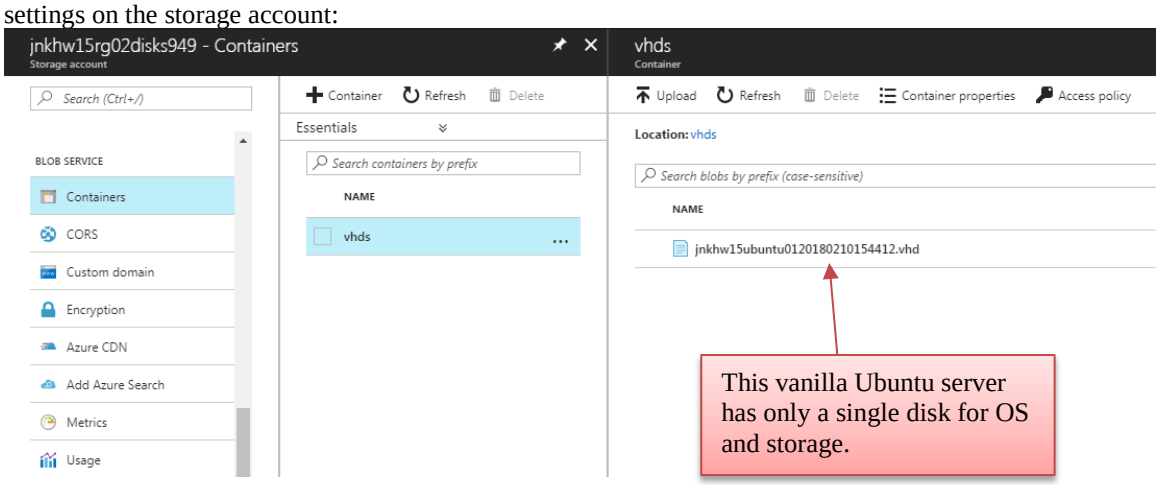
The results are lackluster.



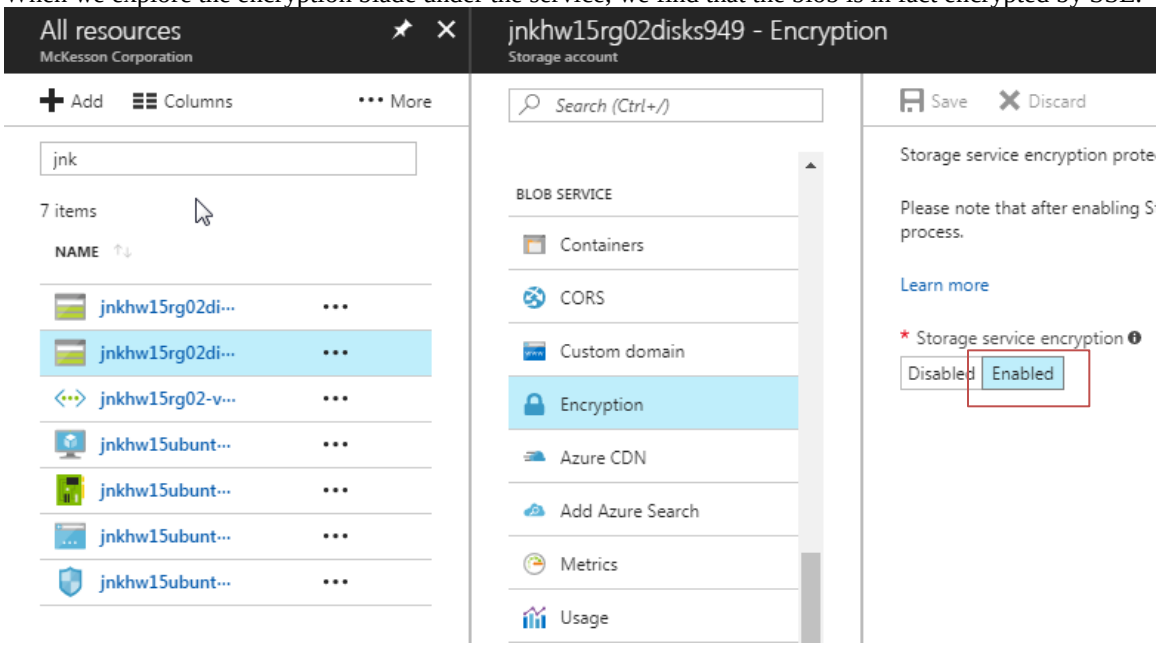
Dig deeper:
Instantiate an Ubuntu 16.04 server VM using *unmanaged* disks (according to Microsoft all managed disks are encrypted by default, unmanaged disks are not mentioned) and review the results in the resources pane. Below is a capture of the parts making up an Ubuntu (or any) server within the cloud:



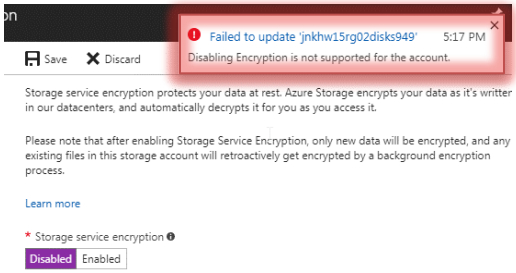
…disks949 are the germane object in this case, a device type of "storage account". Virtual hard disks (VHDs) are stored as a .vhd file within a blob container. We can visualize this by drilling into the blob settings on the storage account:



When we explore the encryption blade under the service, we find that the blob is in fact encrypted by SSE.



…and we are unable to disable it:

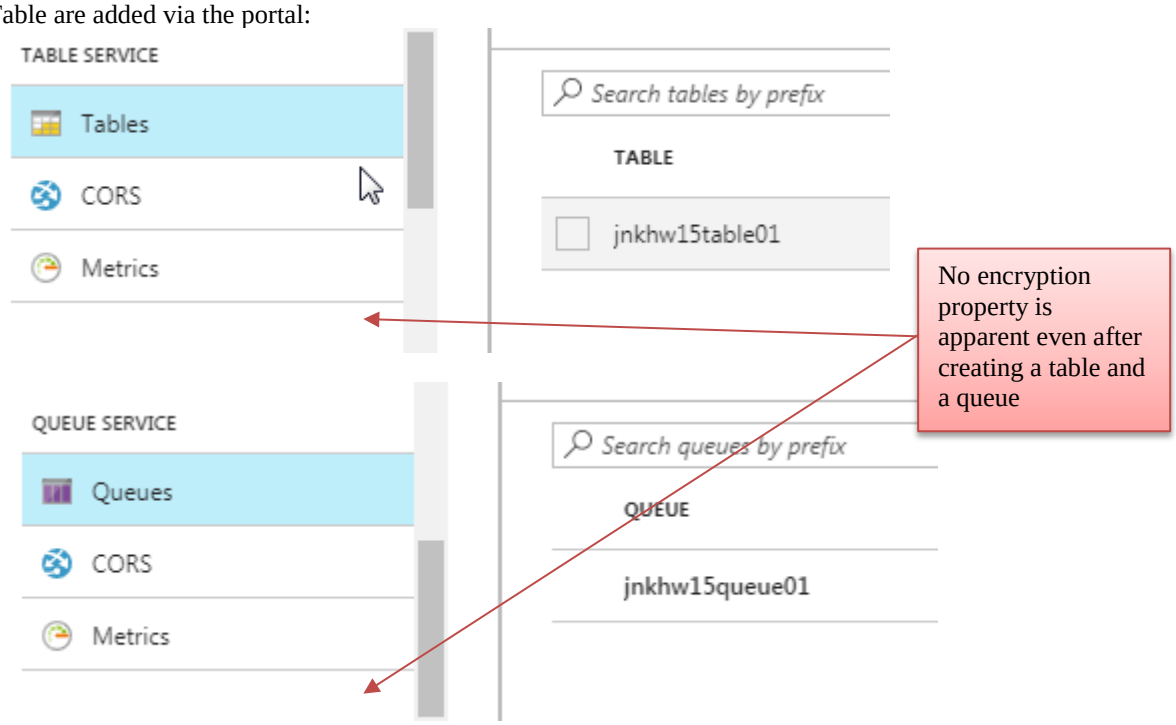


We now have two storage accounts created first deliberately using the CLI, and second created coincident to instantiating a server VM. In both cases the SSE encryption was enabled without question and is 'locked' against change via the portal.

What about the CLI?
Here we can see encryption as a property. Note that Queue and Table are "null" which was one of our tests for this problem – there is no encryption from SSE on these elements. At this stage these elements do not exist.

```
jeff_keir@Azure:~$ az storage account show --resource-group jnkhw15rg02 --name jnkhw15rg02disks949
{
  "accessTier": null,
  "creationTime": "2018-02-10T22:44:44.273338+00:00",
  "customDomain": null,
  "enableHttpsTrafficOnly": null,
  "encryption": {
    "keySource": "Microsoft.Storage",
    "keyVaultProperties": null,
    "services": {
      "blob": {
        "enabled": true,
        "lastEnabledTime": "2018-02-10T22:44:44.304571+00:00"
      },
      "file": {
        "enabled": true,
        "lastEnabledTime": "2018-02-10T22:44:44.304571+00:00"
      },
      "queue": null,
      "table": null
    }
  },
```

To understand if Queues and Tables instantiated after a storage account has been created, a Queue and a Table are added via the portal:



Re-visualizing in the CLI shows the encryption state for these storage types remains null even after creation within an encrypted storage service:

```
jeff_keir@Azure:~$ az storage account show --resource-group jnkhw15rg02 --name jnkhw15rg02disks949
{
  "accessTier": null,
  "creationTime": "2018-02-10T22:44:44.273338+00:00",
  "customDomain": null,
  "enableHttpsTrafficOnly": null,
  "encryption": {
    "keySource": "Microsoft.Storage",
    "keyVaultProperties": null,
    "services": {
      "blob": {
        "enabled": true,
        "lastEnabledTime": "2018-02-10T22:44:44.304571+00:00"
      },
      "file": {
        "enabled": true,
        "lastEnabledTime": "2018-02-10T22:44:44.304571+00:00"
      },
      "queue": null,
      "table": null
    }
  },
```

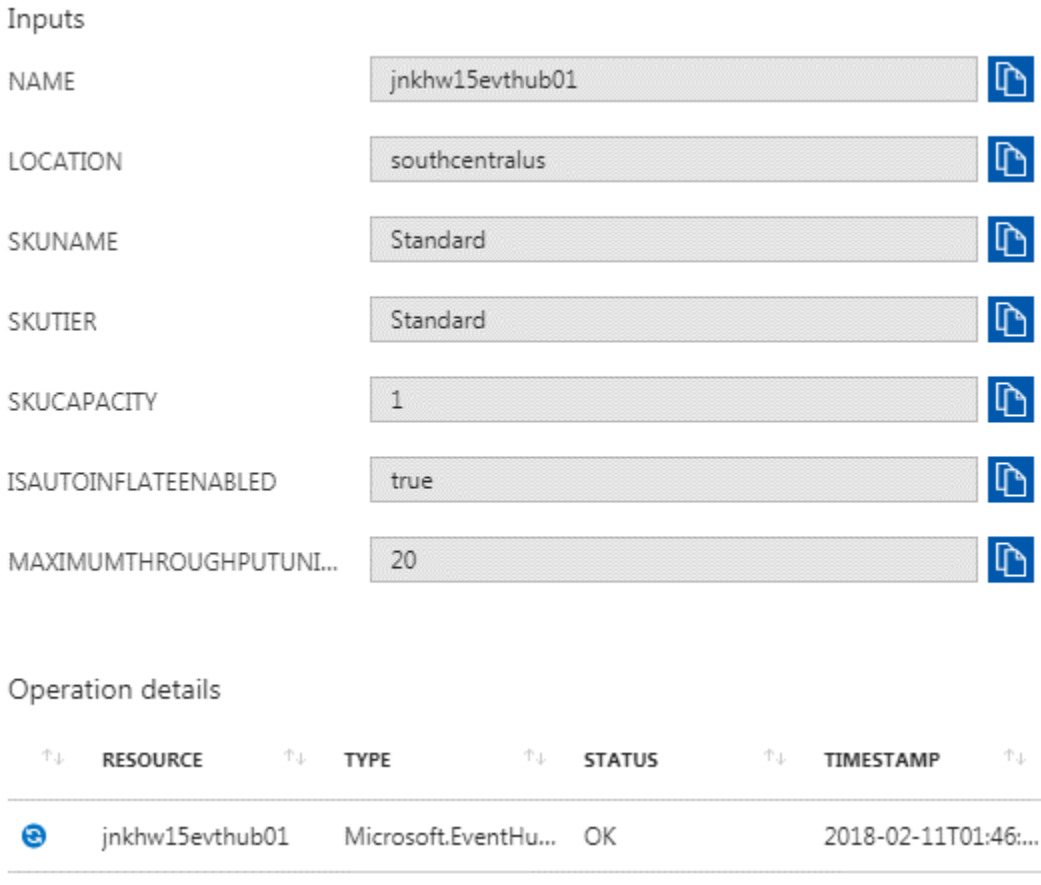
Within the CLI usage hints for storage we can see that there are some encryption elements we can change:

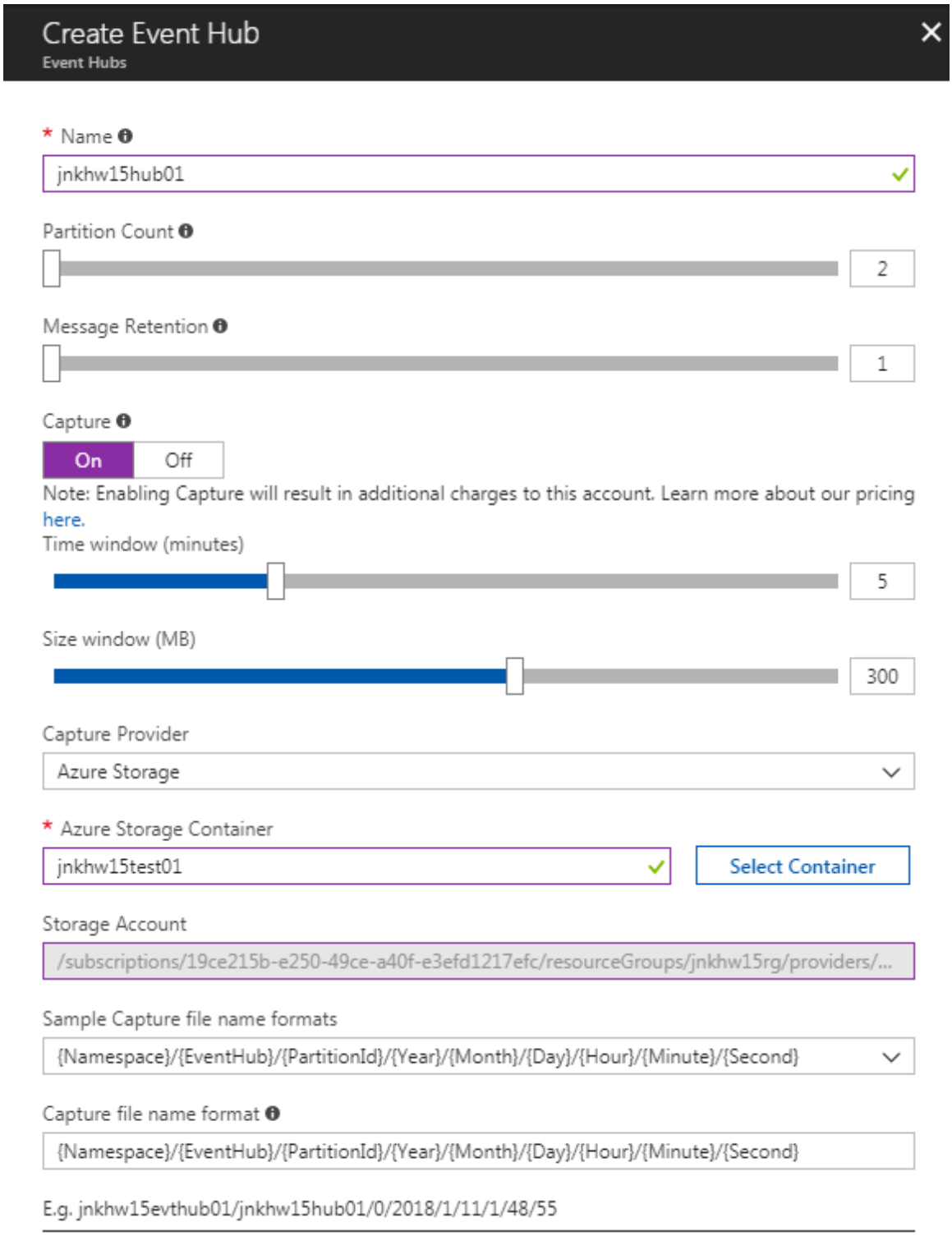
```
usage: az storage account [-h]
                          {check-name,create,delete,show,list,show-usage,show-connection-string,update,keys,generate-sas,network-rule}
                          ...
jeff_keir@Azure:~$ az storage account update
usage: az storage account update [-h] [--verbose] [--debug]
                                 [--output {json,jsonc,table,tsv}]
                                 [--query JMESPATH]
                                 [--resource-group RESOURCE_GROUP_NAME]
                                 [--name ACCOUNT_NAME]
                                 [--sku {Standard_LRS,Standard_GRS,Standard_RAGRS,Standard_ZRS,Premium_LRS}]
                                 [--tags [TAGS [TAGS ...]]]
                                 [--custom-domain CUSTOM_DOMAIN]
                                 [--use-subdomain {true,false}]
                                 [--encryption-services {blob,file,table,queue} [{blob,file,table,queue} ...]]
                                 [--encryption-key-source ENCRYPTION_KEY_SOURCE]
                                 [--encryption-key-vault-properties ENCRYPTION_KEY_VAULT_PROPERTIES]
                                 [--access-tier {Hot,Cool}]
                                 [--https-only [{true,false}]]
                                 [--assign-identity]
                                 [--bypass {None,Logging,Metrics,AzureServices} [{None,Logging,Metrics,AzureServices} ...]]
                                 [--default-action {Allow,Deny}]
                                 [--set KEY=VALUE [KEY=VALUE ...]]
                                 [--add LIST KEY=VALUE [LIST KEY=VALUE ...]]
                                 [--remove LIST INDEX [LIST INDEX ...]]
                                 [--ids RESOURCE_ID [RESOURCE_ID ...]]
az storage account update: error: (--resource-group --name | --ids) are required
```

Namely: service type, key source, and key vault properties. But there is no actual toggle.

Recreate the messaging services from Deep Dive Homework 09, demonstrate the encryption state, and that the base functionality is successful:

Namespace:




Create Event Hub
Event Hubs
* Name
jnkhw15hub01
Partition Count
2
Message Retention
1
Capture
On
Off
Note: Enabling Capture will result in additional charges to this account. Learn more about our pricing here.
Time window (minutes)
5
Size window (MB)
300
Capture Provider
Azure Storage
* Azure Storage Container
jnkhw15test01
Select Container
Storage Account
/subscriptions/19ce215b-e250-49ce-a40f-e3efd1217efc/resourceGroups/jnkhw15rg/providers/...
Sample Capture file name formats
{Namespace}/{EventHub}/{PartitionId}/{Year}/{Month}/{Day}/{Hour}/{Minute}/{Second}
Capture file name format
{Namespace}/{EventHub}/{PartitionId}/{Year}/{Month}/{Day}/{Hour}/{Minute}/{Second}
E.g. jnkhw15evthub01/jnkhw15hub01/0/2018/1/11/1/48/55

Code – the following code snippets are taken from https://github.com/blumu/azure-content/blob/master/articles/event-hubs/event-hubs-archive-python.md:

This script sends 20 messages to the storage service

```python
import uuid
import datetime
import random
import json
from azure.servicebus import ServiceBusService

sbs = ServiceBusService(service_namespace='jnkhw15evthub01', shared_access_key_name='RootManageSharedAccessKey', shared_access_key_value='oZT//BAPVlfbxFkfhikt3J11DRbHCRe3A9VBOVh0tVQ=')
devices = []
for x in range(0, 10):
    devices.append(str(uuid.uuid4()))

for y in range(0,20):
    for dev in devices:
        reading = {'id': dev, 'timestamp': str(datetime.datetime.utcnow()), 'uv': random.random(), 'temperature': random.randint(70, 100), 'humidity': random.randint(70, 100)}
        s = json.dumps(reading)
        sbs.send_event('jnkhw15hub01', s)
    print(y)
```

This script calls upon the storage service to process messages

```python
import os
import string
import json
import avro.schema
from avro.datafile import DataFileReader, DataFileWriter
from avro.io import DatumReader, DatumWriter
from azure.storage.blob import BlockBlobService

def processBlob(filename):
    reader = DataFileReader(open(filename, 'rb'), DatumReader())
    dict = {}
    for reading in reader:
        parsed_json = json.loads(reading["Body"])
        if not 'id' in parsed_json:
            return
        if not dict.has_key(parsed_json['id']):
            list = []
            dict[parsed_json['id']] = list
        else:
            list = dict[parsed_json['id']]
            list.append(parsed_json)
    reader.close()
    for device in dict.keys():
        deviceFile = open(device + '.csv', "a")
        for r in dict[device]:
            deviceFile.write(", ".join([str(r[x]) for x in r.keys()])+'\n')

def startProcessing(accountName, key, container):
    print 'Processor started using path: ' + os.getcwd()
    block_blob_service = BlockBlobService(account_name=accountName, account_key=key)
    generator = block_blob_service.list_blobs(container)
    for blob in generator:
        if blob.properties.content_length > 508:
            print('Downloaded a non empty blob: ' + blob.name)
            cleanName = string.replace(blob.name, '/', '_')
            block_blob_service.get_blob_to_path(container, blob.name, cleanName)
            processBlob(cleanName)
            os.remove(cleanName)
        block_blob_service.delete_blob(container, blob.name)
startProcessing('jnkhw15stora01', 'J/7wuAfGJNxIv5IrtkJvDo2MyvNrMR/Bj3NQjTfrtgyLXQavGSQL31U94vaHaumtyIZEeLgFnjPoyp83n5mn5w==', 'jnkhw15test01')
```

The practical application of these code samples is simply to see that data is indeed written into an encrypted storage blob “jnkhw15hub01”, subsequently extracted and manipulated, the results written to blob “jnkhw15test01” and then a copy to the local machine and readable.

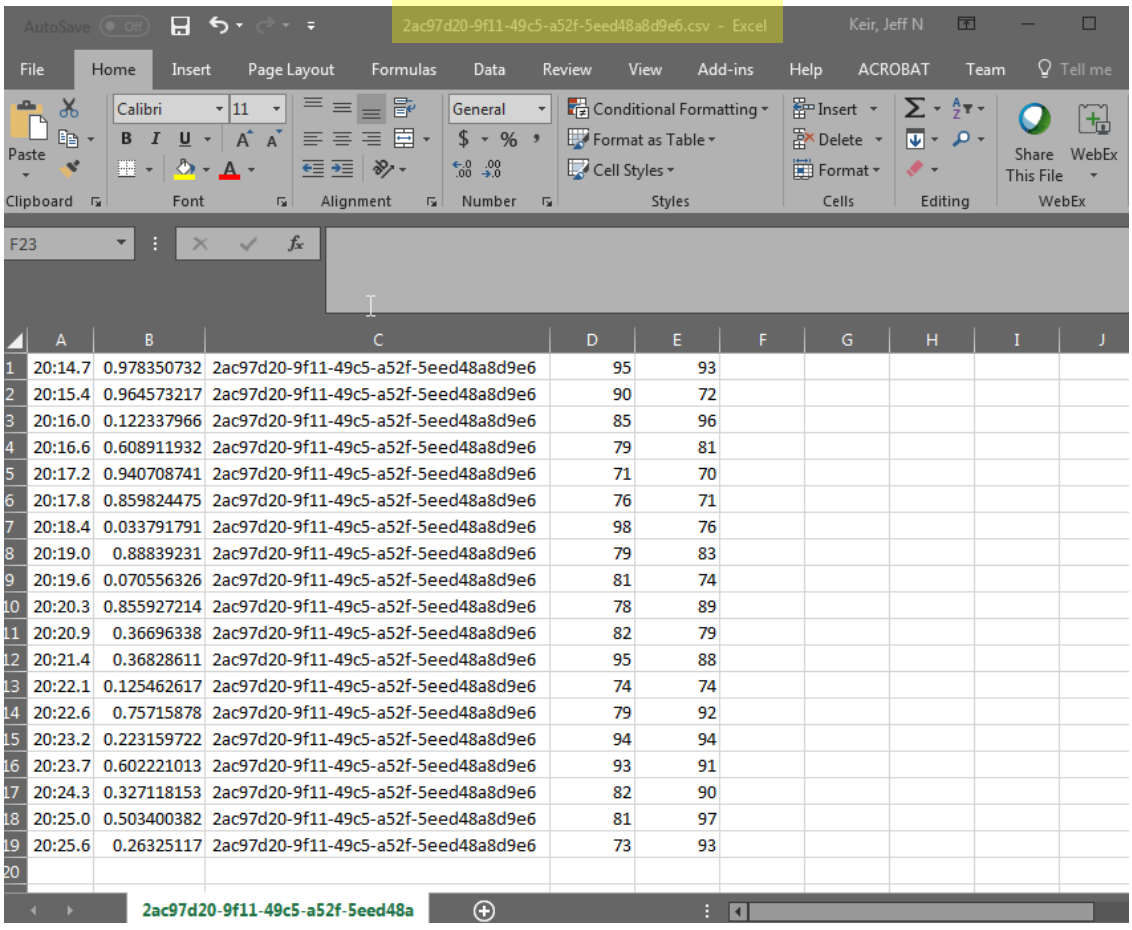
```
(C:\ProgramData\Anaconda2) C:\hw15>python hw15sender.py
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

```
(C:\ProgramData\Anaconda2) C:\hw15>python hw15capture.py
Processor started using path: C:\hw15
Downloaded a non empty blob: jnkhw15evthub01/jnkhw15hub01/0/2018/02/11/02/24/56.
avro
Downloaded a non empty blob: jnkhw15evthub01/jnkhw15hub01/1/2018/02/11/02/24/55.
avro
```

```
(C:\ProgramData\Anaconda2) C:\hw15>dir
 Volume in drive C has no label.
 Volume Serial Number is A604-3594

 Directory of C:\hw15

02/10/2018  07:29 PM    <DIR>          .
02/10/2018  07:29 PM    <DIR>          ..
02/10/2018  07:15 PM    <DIR>          .ipynb_checkpoints
02/10/2018  07:29 PM             1,714 247d5b5d-ea27-4e71-a84a-821704c63b32.csv
02/10/2018  07:29 PM             1,709 2ac97d20-9f11-49c5-a52f-5eed48a8d9e6.csv
02/10/2018  07:29 PM             1,713 2b8f752b-f068-4df2-aa20-0045e8ddc9af.csv
02/10/2018  07:29 PM             1,717 4203f181-c25e-4f9f-b8c4-bcb637cbd693.csv
02/10/2018  07:29 PM             1,710 8238a721-87c8-45c4-8809-1fd5f2f33640.csv
02/10/2018  07:29 PM             1,711 a770bedb-6e1c-4390-a127-722eccde2bf7.csv
02/10/2018  07:29 PM             1,710 ac24d994-fd84-4791-942f-3136729af571.csv
02/10/2018  07:29 PM             1,711 b5a31fe6-9c60-4286-afc6-4e3eb0a4b412.csv
02/10/2018  07:29 PM             1,713 b9132b07-b8c0-4df3-a868-0c4a564c280b.csv
02/10/2018  07:29 PM             1,707 cd79277c-0422-4581-baa6-9227f5010bc5.csv
02/10/2018  07:22 PM             1,669 hw15capture.py
02/10/2018  07:13 PM               682 hw15sender.py
02/10/2018  07:14 PM                 0 untitled
02/10/2018  07:27 PM             4,123 Untitled.ipynb
              14 File(s)         23,589 bytes
               3 Dir(s)  475,152,011,264 bytes free
```



	A	B	C	D	E	F	G	H	I	J
1	20:14.7	0.978350732	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	95	93					
2	20:15.4	0.964573217	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	90	72					
3	20:16.0	0.122337966	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	85	96					
4	20:16.6	0.608911932	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	79	81					
5	20:17.2	0.940708741	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	71	70					
6	20:17.8	0.859824475	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	76	71					
7	20:18.4	0.033791791	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	98	76					
8	20:19.0	0.88839231	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	79	83					
9	20:19.6	0.070556326	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	81	74					
10	20:20.3	0.855927214	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	78	89					
11	20:20.9	0.36696338	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	82	79					
12	20:21.4	0.36828611	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	95	88					
13	20:22.1	0.125462617	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	74	74					
14	20:22.6	0.75715878	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	79	92					
15	20:23.2	0.223159722	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	94	94					
16	20:23.7	0.602221013	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	93	91					
17	20:24.3	0.327118153	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	82	90					
18	20:25.0	0.503400382	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	81	97					
19	20:25.6	0.26325117	2ac97d20-9f11-49c5-a52f-5eed48a8d9e6	73	93					
20										

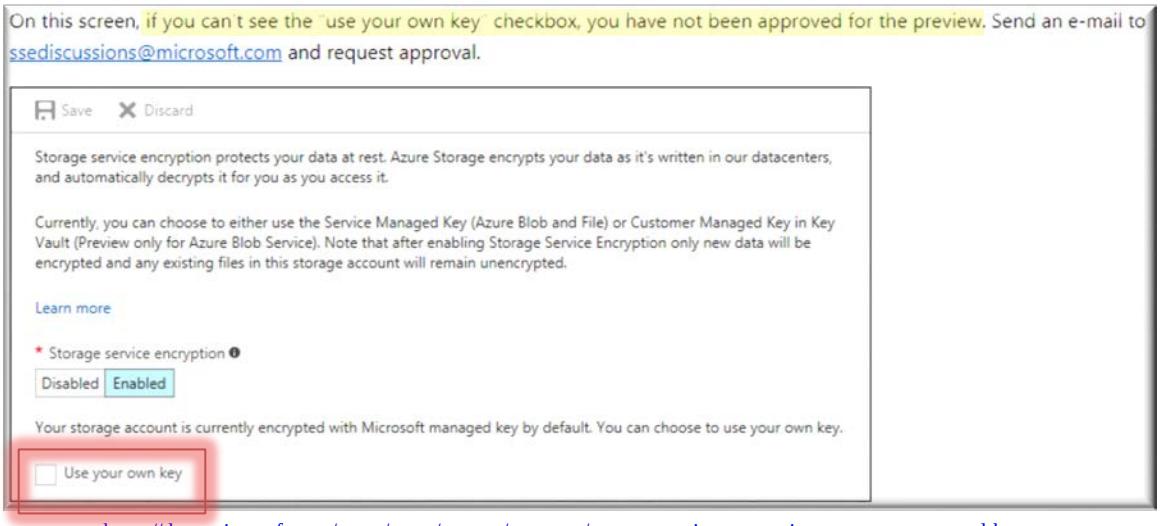
Conclusion:

It is evident that encryption is enabled, as Microsoft claims, on Resource Manager Storage by default. Furthermore, it is easy to see the settings either in the portal or via the CLI.

What is not evident is any mechanism by which the administrator could deactivate the service. Though the CLI and APIs are referenced frequently in the documentation, no actual syntax is provided to toggle the encryption state. Relatedly, there is little given information on the amount of disk-access-time added to operations by the existence of Storage Service Encryption. Practical experience indicates that the overhead is small, though one imagines a proportionate increase in wait time as the size of a given data-set increases.

As recently as 10 months ago it may have been possible to instantiate unencrypted storage and toggle the setting, but today it appears impossible.

All indications are that Microsoft is still working on the tools that allow developers to manage encryption across the storage service *manually*. The foundation for this conclusion is inferred from a reference to the preview of customer-managed storage service encryption on the document page:



https://docs.microsoft.com/en-us/azure/storage/common/storage-service-encryption-customer-managed-keys