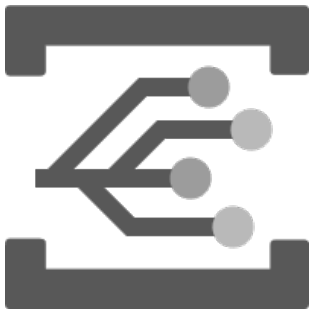
Particular Software

Deep Dive with Azure Service Bus

@danielmarbach

Eventing



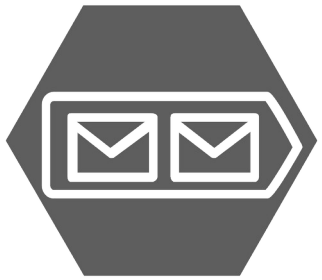
Event Grid

Cross cloud reactive eventing

Event Hubs

Big data streaming

Messaging



Storage Queues

Simple task queues

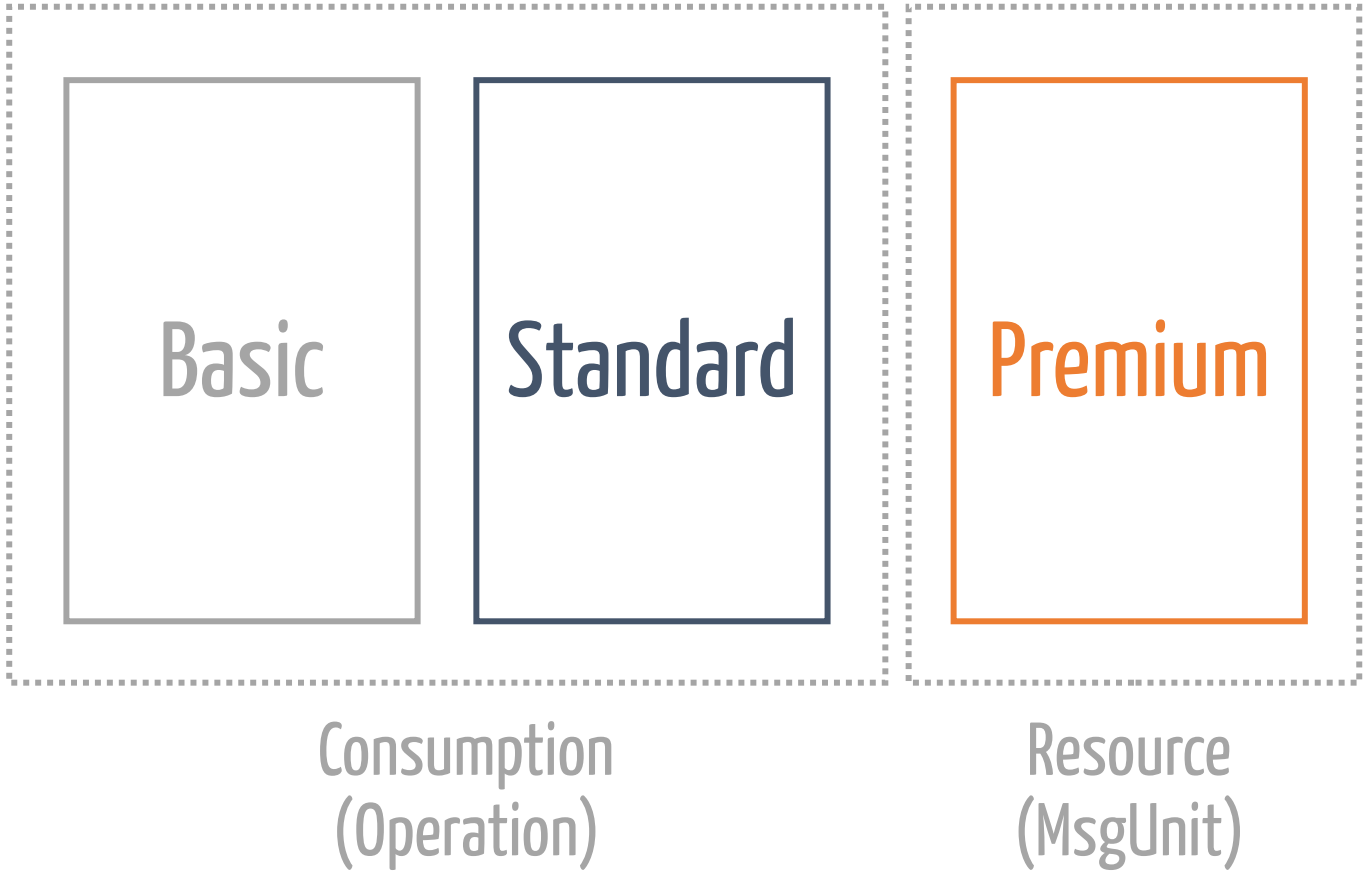
Service Bus

Enterprise messaging

Namespace

Unit of Isolation

Tiers


Basic
Standard
Premium
Consumption
(Operation)
Resource
(MsgUnit)

nuget Packages Upload Statistics Documentation Downloads Blog Sign in

azure service bus

468 packages returned for azure service bus

Filter

Azure.Messaging.ServiceBus by: azure-sdk Microsoft

24,600,513 total downloads last updated a month ago Latest version: 7.11.0-beta.1

Azure Service Bus ServiceBus .NET AMQP windowsazureofficial azureofficial

Azure Service Bus is a fully managed enterprise integration message broker. Service Bus can decouple applications and services. Service Bus offers a reliable and secure platform for asynchronous transfer of data and state. This client library allows for both sending and receiving messages using... More information

Azure.ResourceManager.ServiceBus by: azure-sdk Microsoft

9,457 total downloads last updated 3 months ago Latest version: 1.0.0

azure management arm resource manager azure.resourcemanager.servicebus

Azure management client SDK for Azure resource provider Microsoft.ServiceBus. This is a stable version. This version uses a next-generation code generator that introduces important breaking changes, but also new features (such as intuitive authentication, custom HTTP pipeline, distributed... More information

Microsoft.Azure.WebJobs.Extensions.ServiceBus by: azure-sdk Microsoft

17,731,598 total downloads last updated 2 months ago Latest version: 5.7.0

Microsoft Azure WebJobs SDK ServiceBus Extension

Microsoft.Azure.ServiceBus.EventProcessorHost by: azure-sdk Microsoft nugetservicebus

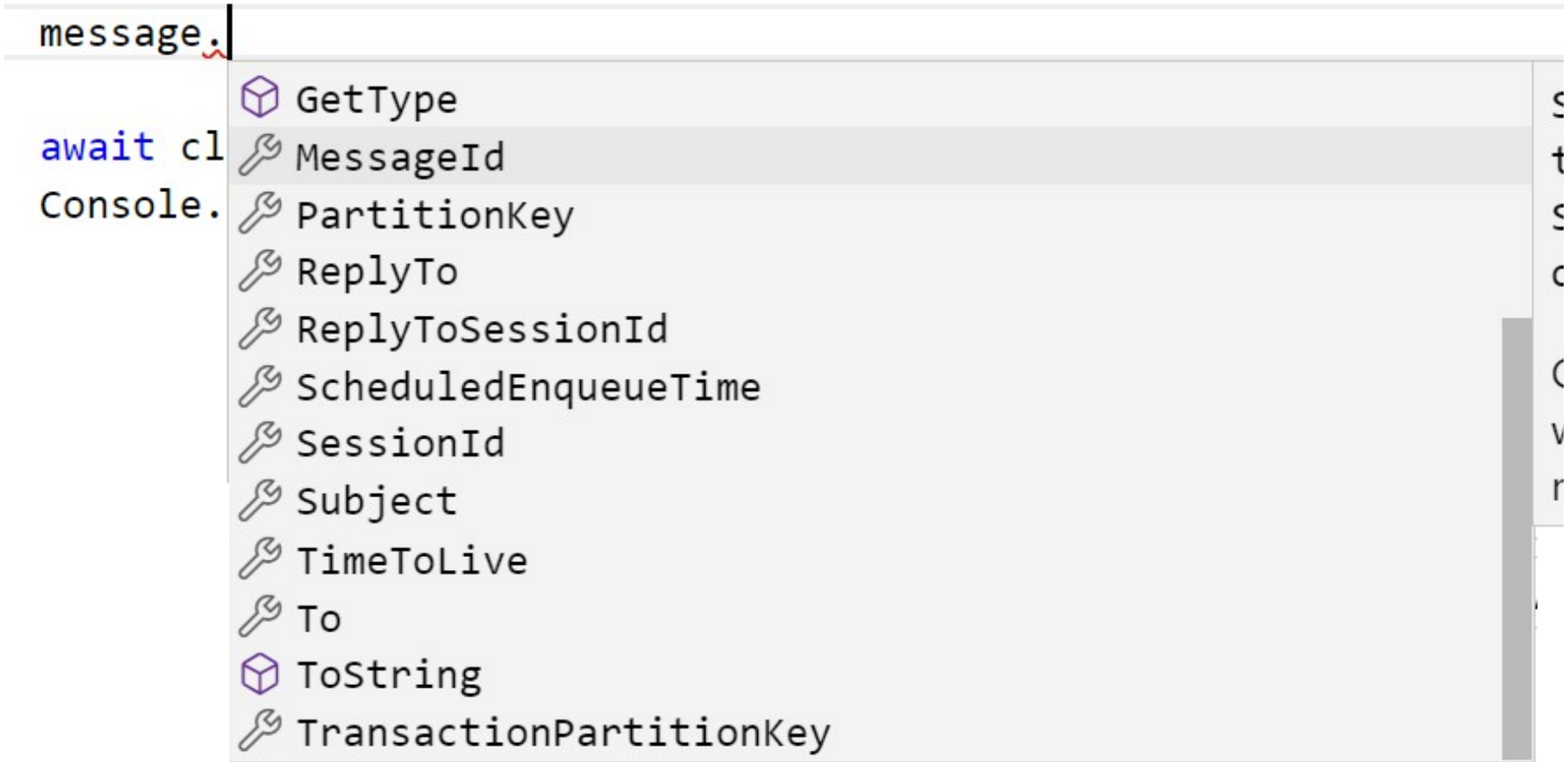
7,856,539 total downloads last updated 6/8/2021 Latest version: 5.0.1

ServiceBus Microsoft Azure AppFabric Messaging PubSub Publish Subscribe Queue Topic More tags

Use this for EventHub distributed partition processing. For more information please visit http://go.microsoft.com/fwlink/?LinkID=403957 Please note that a newer package Azure.Messaging.EventHubs.Processor is available as of February 2020. While this package will continue to receive critical... More information

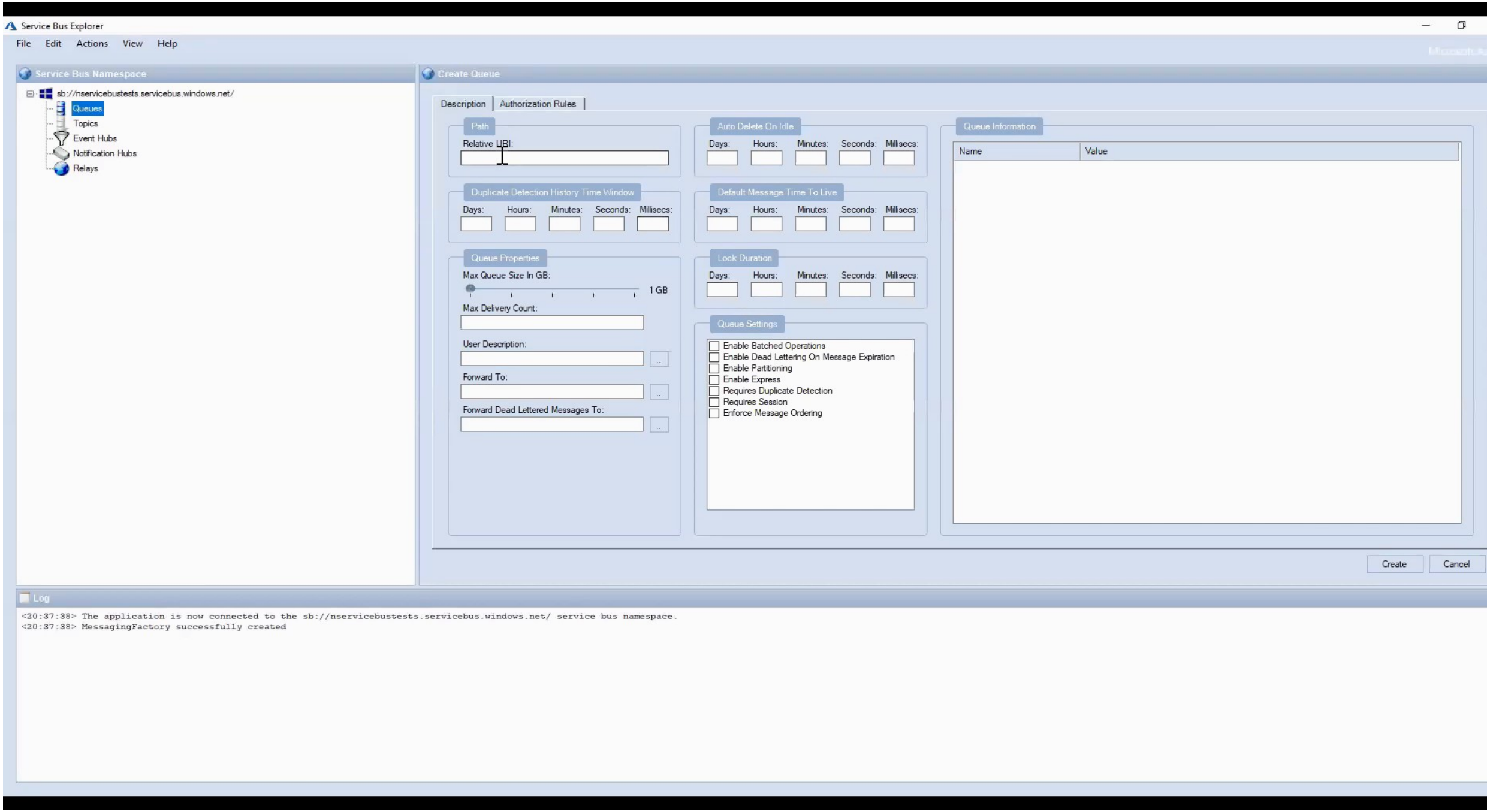
Send

```
await using var serviceBusClient = new ServiceBusClient(connectionString);
await using var client = serviceBusClient.CreateSender(destination);

var message = new ServiceBusMessage("Deep Dive");
<PackageReference Include="Azure.Messaging.ServiceBus" />
message.ApplicationProperties.Add("TenantId", "MyTenantId");

await client.SendMessageAsync(message);
```


message.
await cl
Console.
GetType
MessageId
PartitionKey
ReplyTo
ReplyToSessionId
ScheduledEnqueueTime
SessionId
Subject
TimeToLive
To
ToString
TransactionPartitionKey

<1> cmd - dotnet r...

```
c:\p\AzureServiceBus.DeepDive\Send
> dotnet run

Unhandled Exception: Microsoft.Azure.ServiceBus.MessagingEntityNotFoundException: Put token failed. status-code: 404, status-d
 The messaging entity 'sb://nservicebustests.servicebus.windows.net/queue' could not be found. TrackingId:3884f2d2-5841-4a05-8
cd570_G5, SystemTracker:nservicebustests.servicebus.windows.net:queue, Timestamp:2019-06-14T18:38:14.
   at Microsoft.Azure.ServiceBus.Core.MessageSender.OnSendAsync(IList`1 messageList) in C:\source\azure-service-bus-dotnet\src
Azure.ServiceBus\Core\MessageSender.cs:line 567
   at Microsoft.Azure.ServiceBus.RetryPolicy.RunOperation(Func`1 operation, TimeSpan operationTimeout) in C:\source\azure-serv
net\src\Microsoft.Azure.ServiceBus\RetryPolicy.cs:line 85
   at Microsoft.Azure.ServiceBus.RetryPolicy.RunOperation(Func`1 operation, TimeSpan operationTimeout) in C:\source\azure-serv
net\src\Microsoft.Azure.ServiceBus\RetryPolicy.cs:line 107
   at Microsoft.Azure.ServiceBus.Core.MessageSender.SendAsync(IList`1 messageList) in C:\source\azure-service-bus-dotnet\src\M
ure.ServiceBus\Core\MessageSender.cs:line 257
   at Send.Program.Main(String[] args) in c:\p\AzureServiceBus.DeepDive\Send\Program.cs:line 28
   at Send.Program.Main(String[] args) in c:\p\AzureServiceBus.DeepDive\Send\Program.cs:line 33
   at Send.Program.<Main>(String[] args)
```

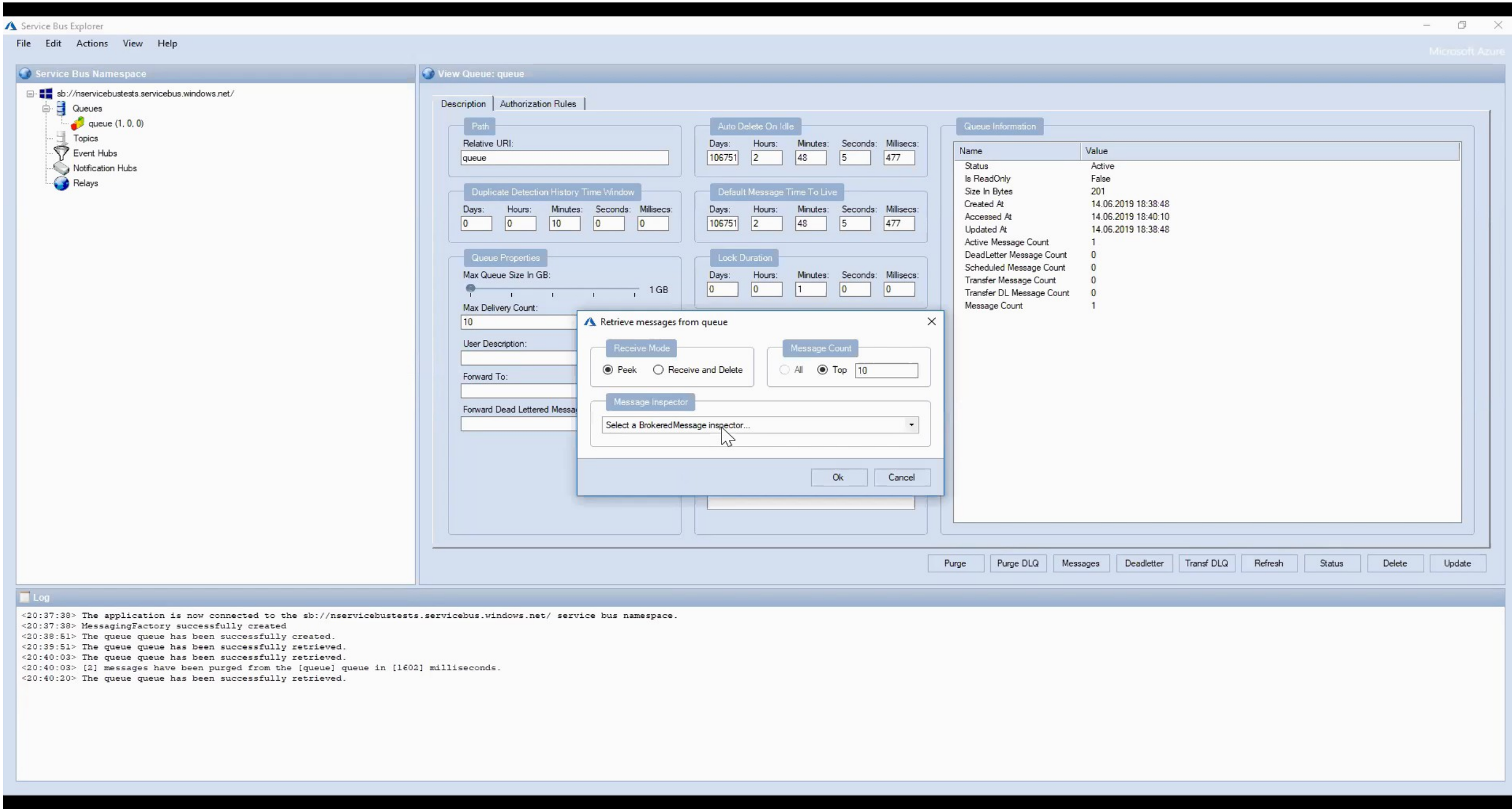

Service Bus Explorer
File
Edit
Actions
View
Help
Service Bus Namespace
sb://nservicebustests.servicebus.windows.net/
Queues
Topics
Event Hubs
Notification Hubs
Relays
Create Queue
Description
Authorization Rules
Path
Relative URI:
Auto Delete On Idle
Days:
Hours:
Minutes:
Seconds:
Millisecs:
Queue Information
Name
Value
Duplicate Detection History Time Window
Default Message Time To Live
Queue Properties
Max Queue Size In GB:
1 GB
Max Delivery Count:
User Description:
Forward To:
Forward Dead Lettered Messages To:
Lock Duration
Queue Settings
Enable Batched Operations
Enable Dead Lettering On Message Expiration
Enable Partitioning
Enable Express
Requires Duplicate Detection
Requires Session
Enforce Message Ordering
Create
Cancel
Log
<20:37:38> The application is now connected to the sb://nservicebustests.servicebus.windows.net/ service bus namespace.
<20:37:38> MessagingFactory successfully created

cmd - dotnet run

<1> cmd - dotnet r...

Search

```
c:\p\AzureServiceBus.DeepDive\Send
> dotnet run
```


Service Bus Explorer
File Edit Actions View Help
Microsoft Azure
Service Bus Namespace
sb://nservicebustests.servicebus.windows.net/
Queues
queue (1, 0, 0)
Topics
Event Hubs
Notification Hubs
Relays
View Queue: queue
Description
Authorization Rules
Path
Relative URI:
queue
Auto Delete On Idle
Days: Hours: Minutes: Seconds: Millisecs:
106751 2 48 5 477
Duplicate Detection History Time Window
Days: Hours: Minutes: Seconds: Millisecs:
0 0 10 0 0
Default Message Time To Live
Days: Hours: Minutes: Seconds: Millisecs:
106751 2 48 5 477
Queue Properties
Max Queue Size In GB:
1 GB
Max Delivery Count:
10
User Description:
Forward To:
Forward Dead Lettered Messa
Lock Duration
Days: Hours: Minutes: Seconds: Millisecs:
0 0 1 0 0
Queue Information
Name Value
Status Active
Is ReadOnly False
Size In Bytes 201
Created At 14.06.2019 18:38:48
Accessed At 14.06.2019 18:40:10
Updated At 14.06.2019 18:38:48
Active Message Count 1
DeadLetter Message Count 0
Scheduled Message Count 0
Transfer Message Count 0
Transfer DL Message Count 0
Message Count 1
Retrieve messages from queue
Receive Mode
Peek
Receive and Delete
Message Count
All
Top
10
Message Inspector
Select a BrokeredMessage inspector...
Ok
Cancel
Purge
Purge DLQ
Messages
Deadletter
Transf DLQ
Refresh
Status
Delete
Update
Log
<20:37:38> The application is now connected to the sb://nservicebustests.servicebus.windows.net/ service bus namespace.
<20:37:38> MessagingFactory successfully created
<20:38:51> The queue queue has been successfully created.
<20:39:51> The queue queue has been successfully retrieved.
<20:40:03> The queue queue has been successfully retrieved.
<20:40:03> [2] messages have been purged from the [queue] queue in [1602] milliseconds.
<20:40:20> The queue queue has been successfully retrieved.

Management

```
var client = new ServiceBusAdministrationClient(connectionString);
await client.CreateQueueAsync(destination);
```

Receive

```
await using var sender = serviceBusClient.CreateSender(destination);
await sender.SendMessageAsync(new ServiceBusMessage("Deep Dive"));
```

```
var receiverOptions = new ServiceBusReceiverOptions
{
    ReceiveMode = ServiceBusReceiveMode.PeekLock,
    PrefetchCount = 10,
    SubQueue = SubQueue.None
};

await using var receiver = serviceBusClient
           .CreateReceiver(destination, receiverOptions);

await foreach (var message in receiver.ReceiveMessagesAsync())
{
    // consume message
}
```

```
var processorOptions = new ServiceBusProcessorOptions
{
    AutoCompleteMessages = false,
    MaxConcurrentCalls = 1,
    MaxAutoLockRenewalDuration = TimeSpan.FromMinutes(10),
    ReceiveMode = ServiceBusReceiveMode.PeekLock,
    PrefetchCount = 10
};

await using var receiver = serviceBusClient.
          CreateProcessor(destination, processorOptions);
```

```
receiver.ProcessMessageAsync += async messageEventArgs =>
{
    var message = messageEventArgs.Message;

    await Out.WriteLineAsync(
        $"Received message with '{message.MessageId}’

and content '{UTF8.GetString(message.Body)}'");

    // throw new InvalidOperationException();

    await messageEventArgs.CompleteMessageAsync(message);

};
```

```
receiver.ProcessErrorAsync += async errorEventArgs =>
{
    await Out.WriteLineAsync($"Exception:
                {errorEventArgs.Exception}");
    await Out.WriteLineAsync($"FullyQualifiedNamespace:

{errorEventArgs.FullyQualifiedNamespace}");
    await Out.WriteLineAsync($"ErrorSource:
                {errorEventArgs.ErrorSource}");
    await Out.WriteLineAsync($"EntityPath:
                {errorEventArgs.EntityPath}");
};
```

```
await receiver.StartProcessingAsync();

await untilStopRequired;

await receiver.StopProcessingAsync();
```

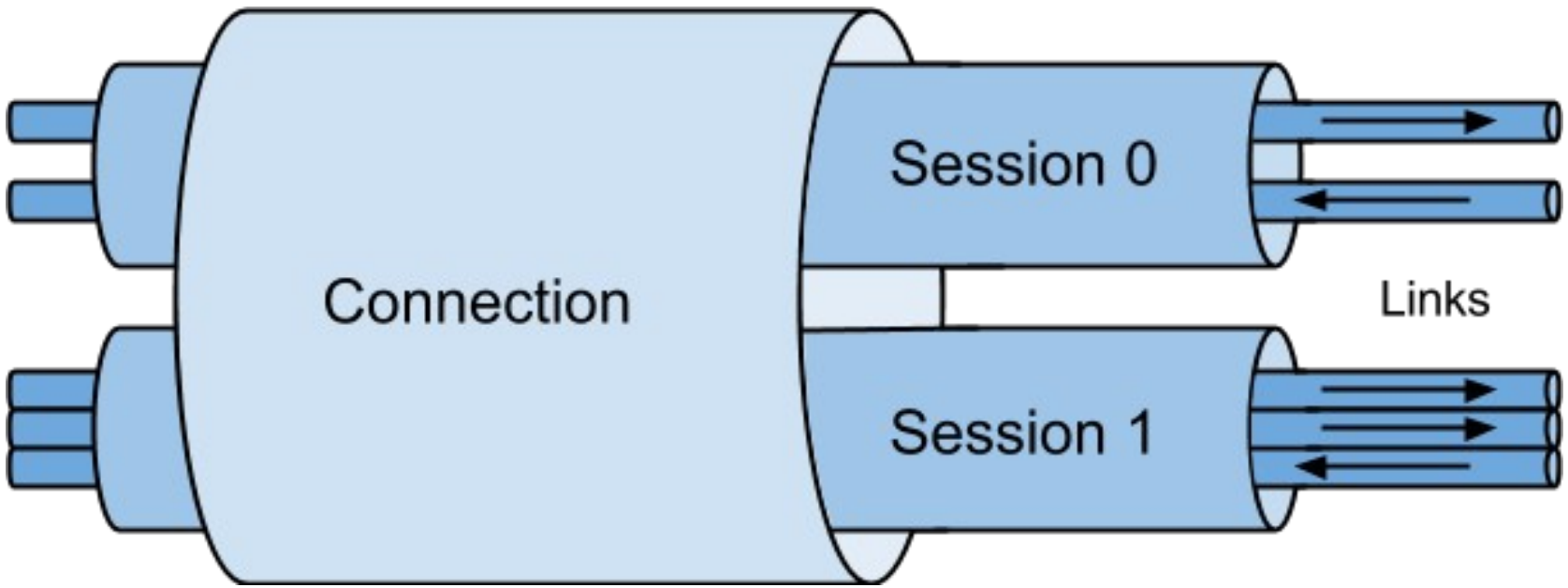
cmd

<1> cmd

Search

```
c:\p\AzureServiceBus.DeepDive\Receive
> dotnet run
Message sent
Received message with 'fe63aaa948444a08aa3c3c13b413c8cc' and content 'Deep Dive'

c:\p\AzureServiceBus.DeepDive\Receive
>
```

Connections


Connection
Session 0
Session 1
Links

```
await using var serviceBusClient =
            new ServiceBusClient(connectionString);

await using var connectionSharingSender =
   serviceBusClient.CreateSender(destination);
await connectionSharingSender
  .SendMessageAsync(new ServiceBusMessage("Deep Dive"));

await using var connectionSharingReceiver =
   serviceBusClient.CreateReceiver(destination);

await connectionSharingReceiver.ReceiveMessageAsync();
```

```
C:\Users\Daniel
> netstat -na | find "5671"
  TCP    192.168.1.14:52253     52.166.127.37:5671     ESTABLISHED
```

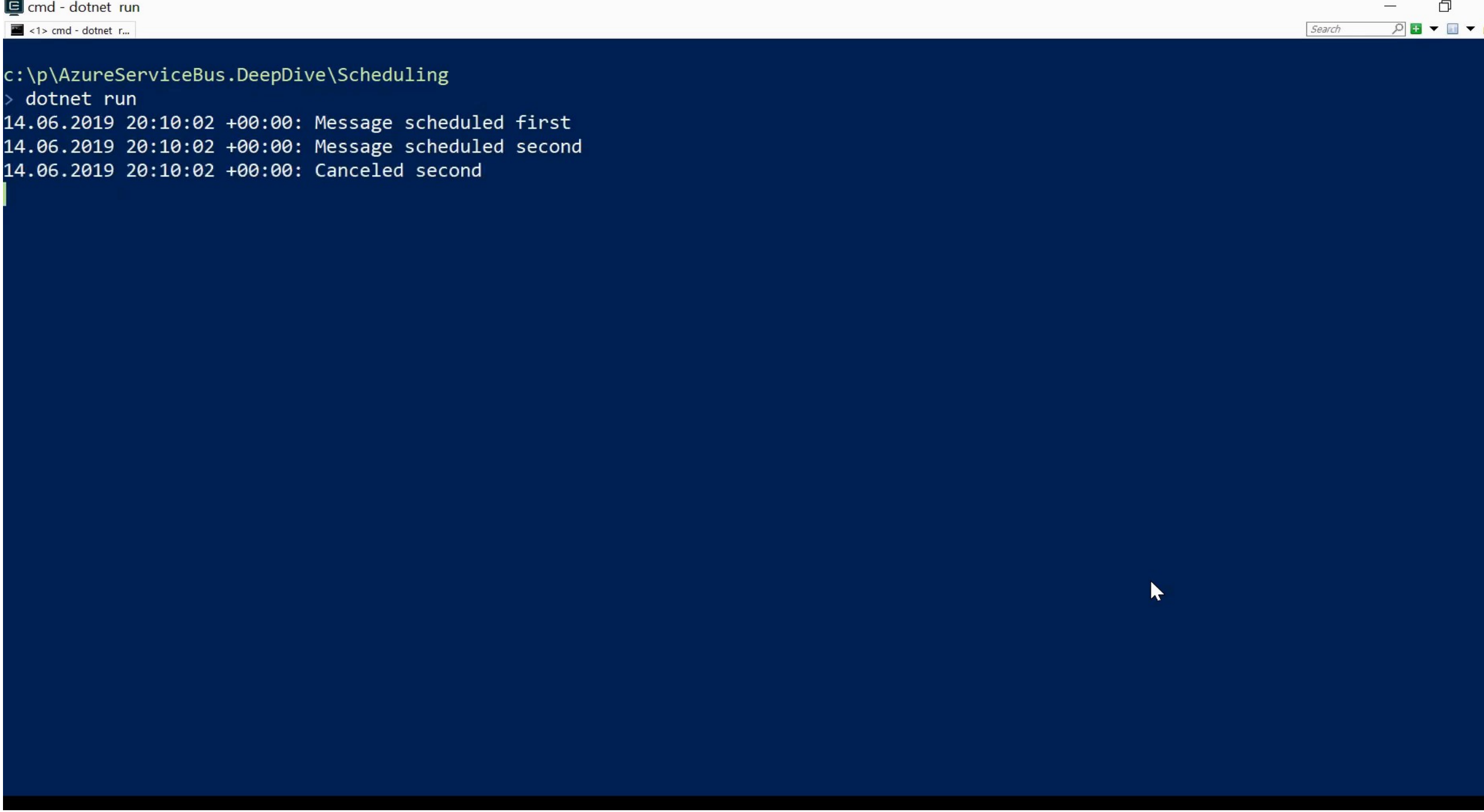
```
await using var senderServiceBusClient =
          new ServiceBusClient(connectionString);
await using var receiverServiceBusClient =
          new ServiceBusClient(connectionString);

await using var senderWithDedicatedConnection =
          senderServiceBusClient.CreateSender(destination);
await using var receiverWithDedicatedConnection =
     receiverServiceBusClient.CreateReceiver(destination);

await senderWithDedicatedConnection
     .SendMessageAsync(new ServiceBusMessage("Deep Dive"));
await receiverWithDedicatedConnection.ReceiveMessageAsync();
```

```
C:\Users\Daniel
> netstat -na | find "5671"
  TCP    192.168.1.14:52249     52.166.127.37:5671     ESTABLISHED
  TCP    192.168.1.14:52250     52.166.127.37:5671     ESTABLISHED
```

Scheduling

```
await using var sender = serviceBusClient.CreateSender(destination);

var due = DateTimeOffset.UtcNow.AddSeconds(10);
await sender.ScheduleMessageAsync(
      new ServiceBusMessage($"Deep Dive + {due}"), due);
```

```
var sequenceId = await sender.ScheduleMessageAsync(
            new ServiceBusMessage($"Deep Dive + {due}"), due);

await sender.CancelScheduledMessageAsync(sequenceId);
```

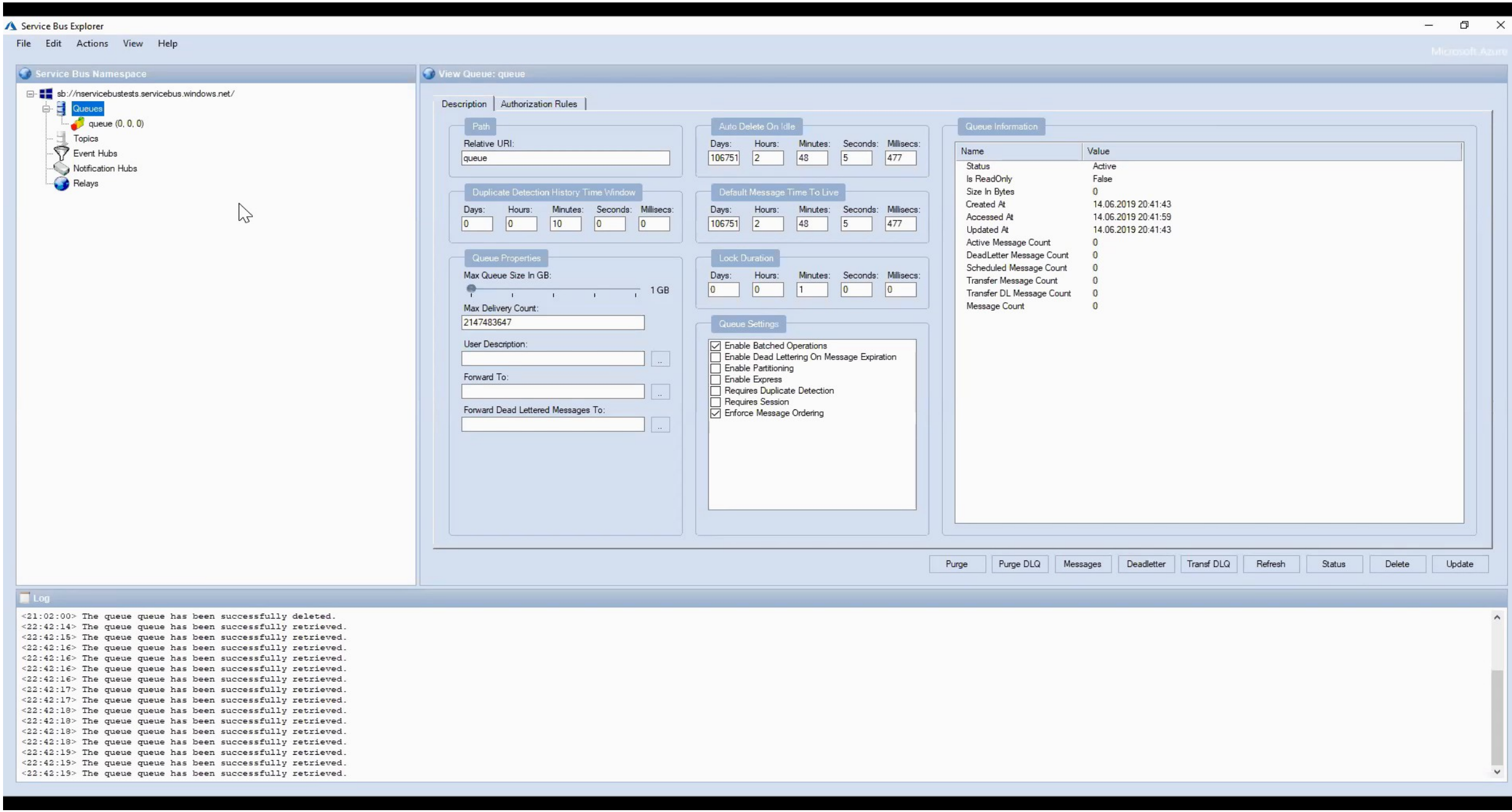

cmd - dotnet run
<1> cmd - dotnet r...
Search
c:\p\AzureServiceBus.DeepDive\Scheduling
> dotnet run
14.06.2019 20:10:02 +00:00: Message scheduled first
14.06.2019 20:10:02 +00:00: Message scheduled second
14.06.2019 20:10:02 +00:00: Canceled second

Expiry

```
await using var sender = serviceBusClient.CreateSender(destination);

var message = new ServiceBusMessage("Half life")
{
    TimeToLive = TimeSpan.FromSeconds(10)
};

await sender.SendMessageAsync(message);

await Prepare.SimulateActiveReceiver(serviceBusClient, destination);
```

```
c:\p\AzureServiceBus.DeepDive\Expiry
> dotnet run
Sent message
```

Where does the message go?


Service Bus Explorer
File
Edit
Actions
View
Help
Service Bus Namespace
sb://nservicebustests.servicebus.windows.net/
Queues
queue (0, 0, 0)
Topics
Event Hubs
Notification Hubs
Relays
View Queue: queue
Description
Authorization Rules
Path
Relative URI:
queue
Auto Delete On Idle
Days: 106751
Hours: 2
Minutes: 48
Seconds: 5
Millisecs: 477
Duplicate Detection History Time Window
Days: 0
Hours: 0
Minutes: 10
Seconds: 0
Millisecs: 0
Default Message Time To Live
Days: 106751
Hours: 2
Minutes: 48
Seconds: 5
Millisecs: 477
Queue Properties
Max Queue Size In GB:
1 GB
Max Delivery Count:
2147483647
User Description:
Forward To:
Forward Dead Lettered Messages To:
Lock Duration
Days: 0
Hours: 0
Minutes: 1
Seconds: 0
Millisecs: 0
Queue Settings
Enable Batched Operations
Enable Dead Lettering On Message Expiration
Enable Partitioning
Enable Express
Requires Duplicate Detection
Requires Session
Enforce Message Ordering
Queue Information
Name
Value
Status Active
Is ReadOnly False
Size In Bytes 0
Created At 14.06.2019 20:41:43
Accessed At 14.06.2019 20:41:59
Updated At 14.06.2019 20:41:43
Active Message Count 0
DeadLetter Message Count 0
Scheduled Message Count 0
Transfer Message Count 0
Transfer DL Message Count 0
Message Count 0
Purge
Purge DLQ
Messages
Deadletter
Transf DLQ
Refresh
Status
Delete
Update
Log
<21:02:00> The queue queue has been successfully deleted.
<22:42:14> The queue queue has been successfully retrieved.
<22:42:15> The queue queue has been successfully retrieved.
<22:42:16> The queue queue has been successfully retrieved.
<22:42:16> The queue queue has been successfully retrieved.
<22:42:16> The queue queue has been successfully retrieved.
<22:42:16> The queue queue has been successfully retrieved.
<22:42:17> The queue queue has been successfully retrieved.
<22:42:17> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:19> The queue queue has been successfully retrieved.
<22:42:19> The queue queue has been successfully retrieved.
<22:42:19> The queue queue has been successfully retrieved.

Deadlettering

```
var description = new CreateQueueOptions(destination)
{
    DeadLetteringOnMessageExpiration = true,
    MaxDeliveryCount = 1
};
await client.CreateQueueAsync(description);
```

```
var message = new ServiceBusMessage("Half life")
{
    TimeToLive = TimeSpan.FromSeconds(1)
};
await sender.SendMessageAsync(message);
```

```
var message = new ServiceBusMessage("Delivery Count");
await sender.SendMessageAsync(message);
```

```
var message = new ServiceBusMessage("Poor Soul");
message.ApplicationProperties.Add("Yehaa", "Why so happy?");
await sender.SendMessageAsync(message);
```

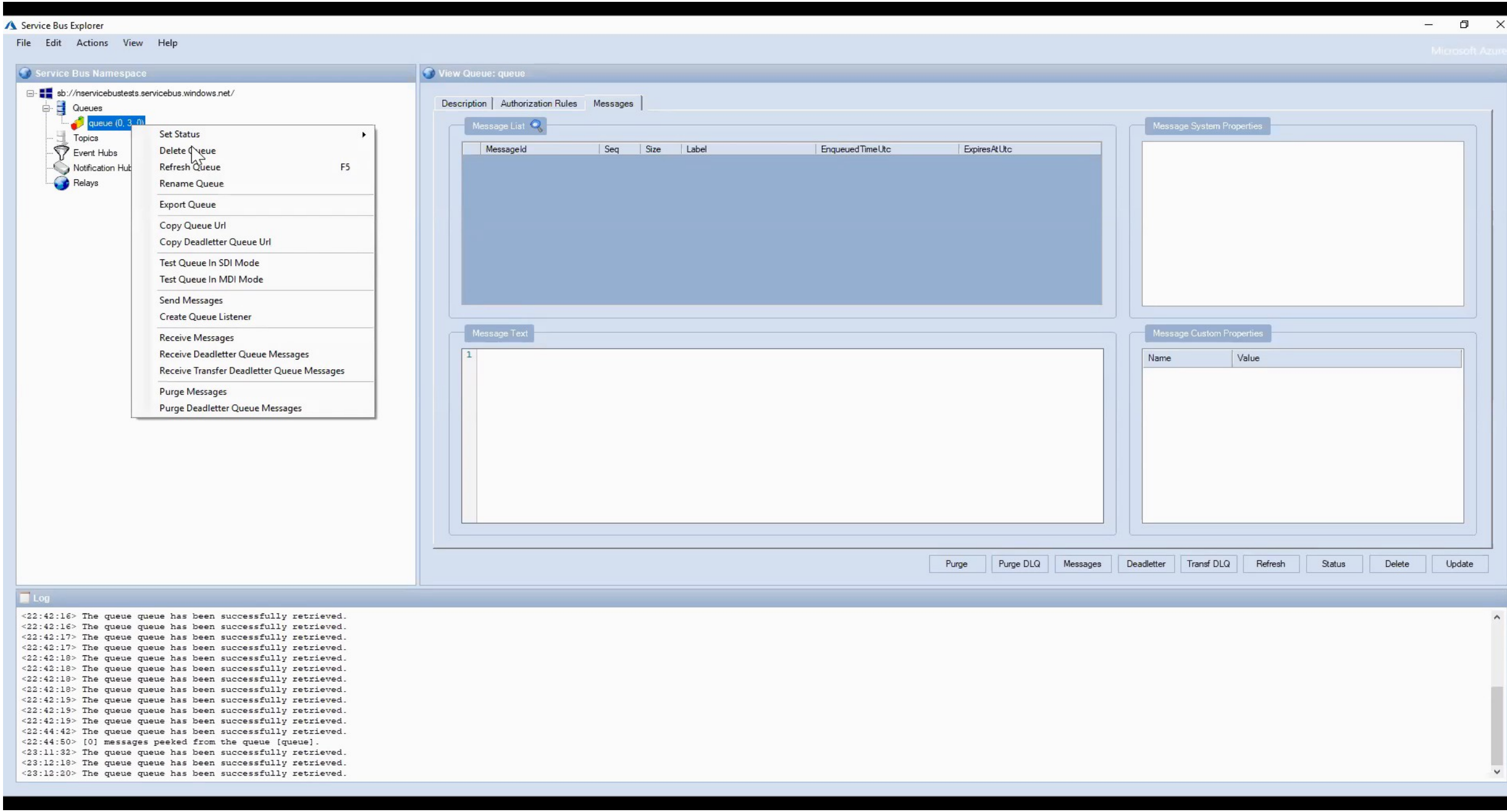
```
receiver.ProcessMessageAsync += async processMessageEventArgs =>
{
    var message = processMessageEventArgs.Message;
    switch (UTF8.GetString(message.Body)) {
        case "Half life":
            await processMessageEventArgs.AbandonMessageAsync(message);
            await Error.WriteLineAsync("Abandon half life message");
            break;

        case "Delivery Count":
            await Error.WriteLineAsync("Throwing delivery count message");
            throw new InvalidOperationException();

        case "Poor Soul":
            await Error.WriteLineAsync("Dead letter poor soul message");
            await processMessageEventArgs.DeadLetterMessageAsync(message,
                new Dictionary<string, object>
                {
                    {"Reason", "Because we can!"},
                    {"When", DateTimeOffset.UtcNow}
                });
            break;
    }
};
```

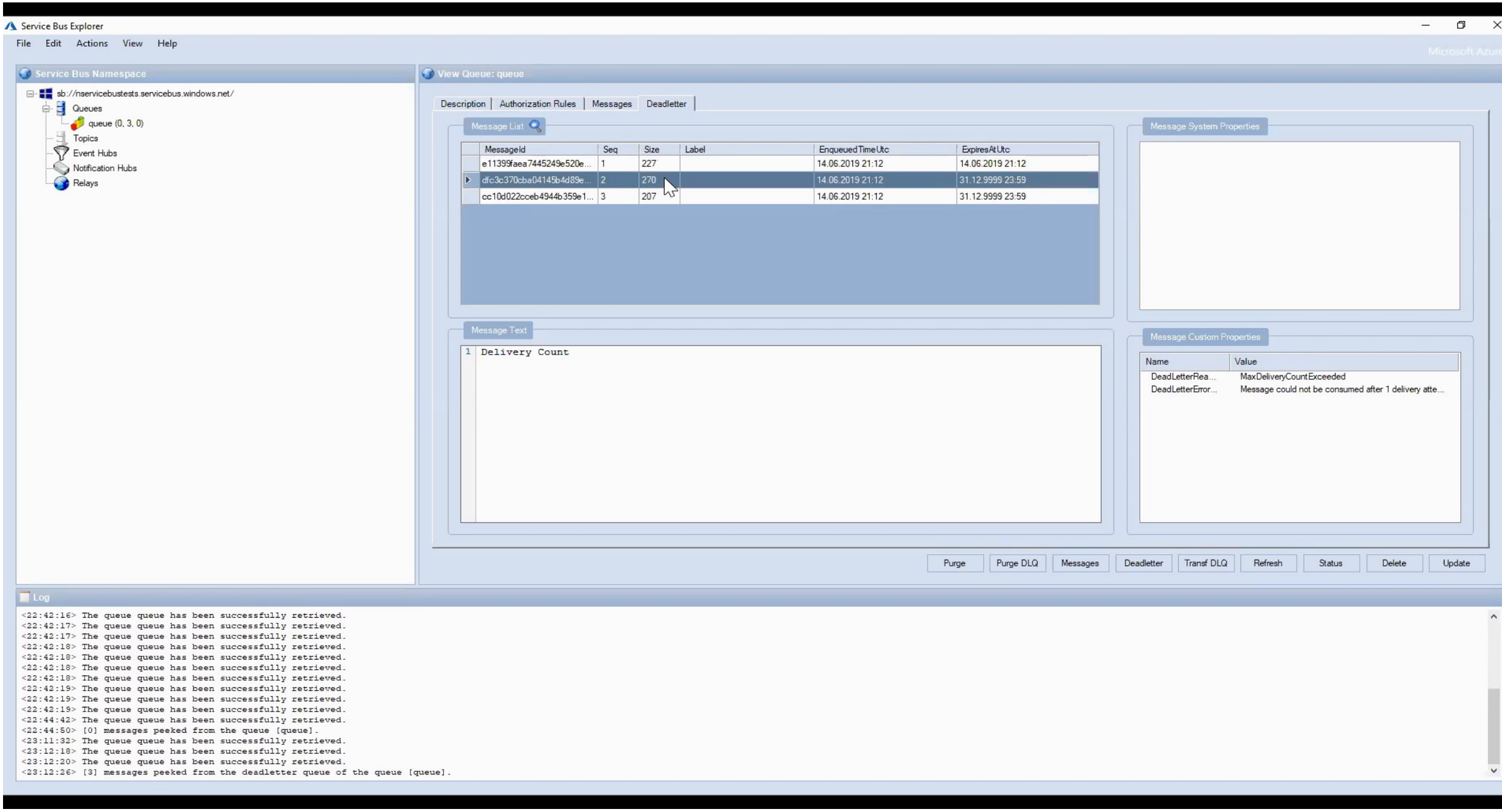
cmd - dotnet run

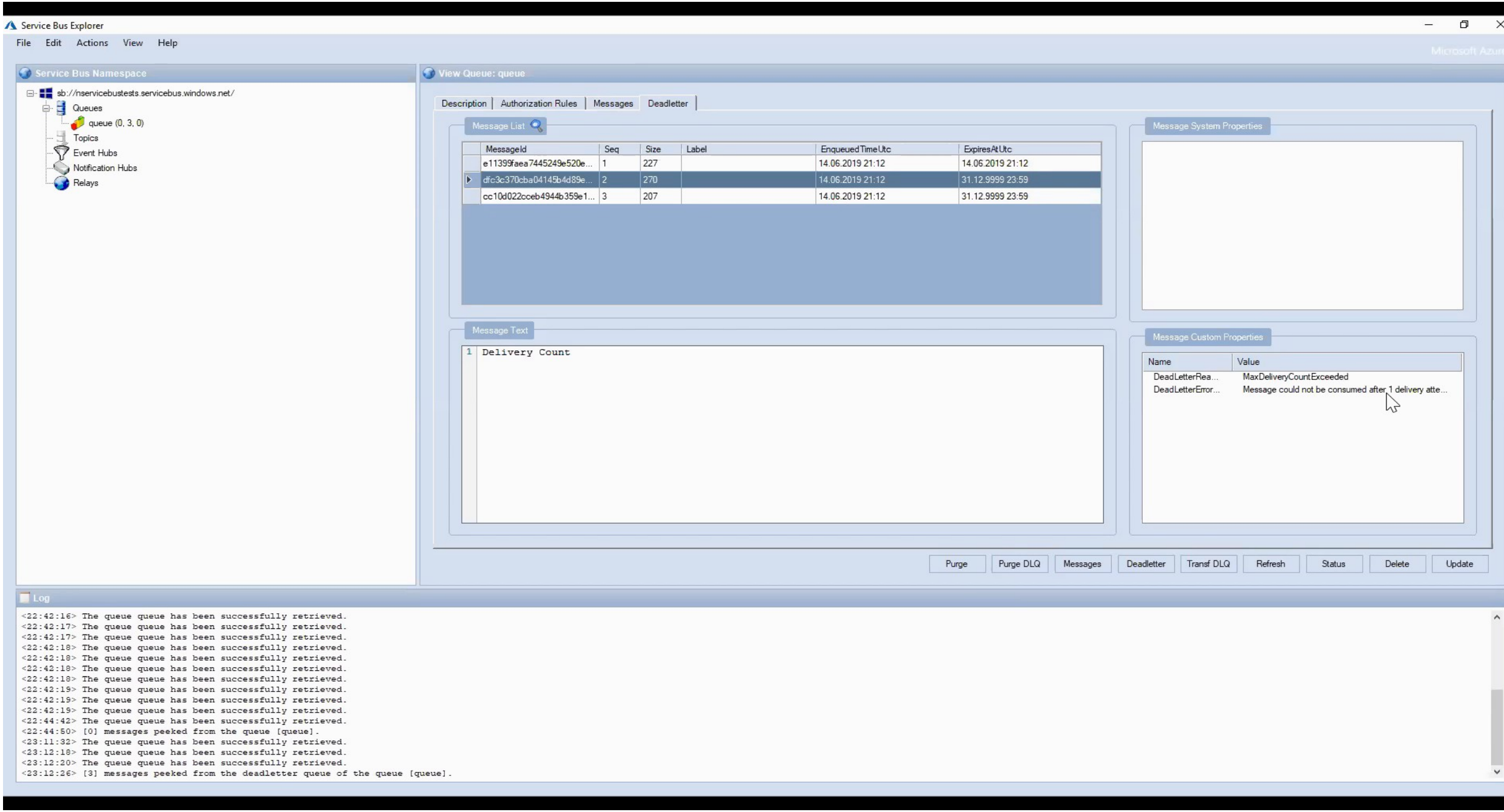
<1> cmd - dotnet r...

Search

```
c:\p\AzureServiceBus.DeepDive\Deadlettering
> dotnet run
Sent half life message
Sent delivery count message
Sent poor soul message
Throwing delivery count message
Dead letter poor soul message
```


Service Bus Explorer
File
Edit
Actions
View
Help
Service Bus Namespace
sb://nservicebustests.servicebus.windows.net/
Queues
queue (0, 3, 0)
Topics
Event Hubs
Notification Hub
Relays
Set Status
Delete Queue
Refresh Queue
F5
Rename Queue
Export Queue
Copy Queue Url
Copy Deadletter Queue Url
Test Queue In SDI Mode
Test Queue In MDI Mode
Send Messages
Create Queue Listener
Receive Messages
Receive Deadletter Queue Messages
Receive Transfer Deadletter Queue Messages
Purge Messages
Purge Deadletter Queue Messages
View Queue: queue
Description
Authorization Rules
Messages
Message List
MessageId
Seq
Size
Label
EnqueuedTimeUtc
ExpiresAtUtc
Message System Properties
Message Text
1
Message Custom Properties
Name
Value
Purge
Purge DLQ
Messages
Deadletter
Transf DLQ
Refresh
Status
Delete
Update
Log
<22:42:16> The queue queue has been successfully retrieved.
<22:42:16> The queue queue has been successfully retrieved.
<22:42:17> The queue queue has been successfully retrieved.
<22:42:17> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:19> The queue queue has been successfully retrieved.
<22:42:19> The queue queue has been successfully retrieved.
<22:42:19> The queue queue has been successfully retrieved.
<22:44:42> The queue queue has been successfully retrieved.
<22:44:50> [0] messages peeked from the queue [queue].
<23:11:32> The queue queue has been successfully retrieved.
<23:12:18> The queue queue has been successfully retrieved.
<23:12:20> The queue queue has been successfully retrieved.


Service Bus Explorer
File
Edit
Actions
View
Help
Service Bus Namespace
sb://nservicebustests.servicebus.windows.net/
Queues
queue (0, 3, 0)
Topics
Event Hubs
Notification Hubs
Relays
View Queue: queue
Description
Authorization Rules
Messages
Deadletter
Message List
MessageId
Seq
Size
Label
EnqueuedTimeUtc
ExpiresAtUtc
e11399faea7445249e520e...
1
227
14.06.2019 21:12
14.06.2019 21:12
dfc3c370cba04145b4d89e...
2
270
14.06.2019 21:12
31.12.9999 23:59
cc10d022cceb4944b359e1...
3
207
14.06.2019 21:12
31.12.9999 23:59
Message System Properties
Message Text
1
Delivery Count
Message Custom Properties
Name
Value
DeadLetterRea...
MaxDeliveryCountExceeded
DeadLetterError...
Message could not be consumed after 1 delivery atte...
Purge
Purge DLQ
Messages
Deadletter
Transf DLQ
Refresh
Status
Delete
Update
Log
<22:42:16> The queue queue has been successfully retrieved.
<22:42:17> The queue queue has been successfully retrieved.
<22:42:17> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:19> The queue queue has been successfully retrieved.
<22:42:19> The queue queue has been successfully retrieved.
<22:42:19> The queue queue has been successfully retrieved.
<22:44:42> The queue queue has been successfully retrieved.
<22:44:50> [0] messages peeked from the queue [queue].
<23:11:32> The queue queue has been successfully retrieved.
<23:12:18> The queue queue has been successfully retrieved.
<23:12:20> The queue queue has been successfully retrieved.
<23:12:26> [3] messages peeked from the deadletter queue of the queue [queue].

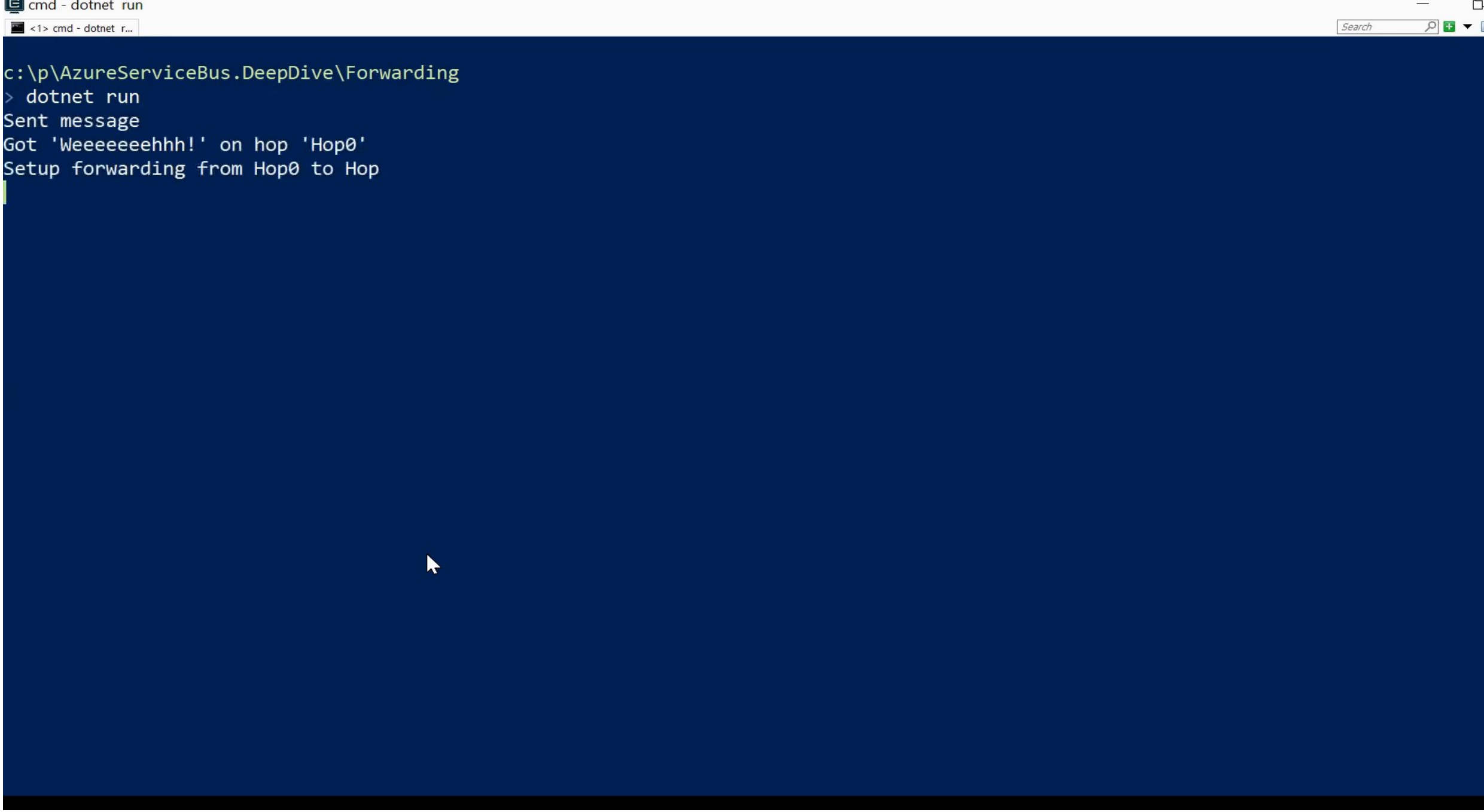

Service Bus Explorer
File
Edit
Actions
View
Help
Service Bus Namespace
sb://nservicebustests.servicebus.windows.net/
Queues
queue (0, 3, 0)
Topics
Event Hubs
Notification Hubs
Relays
View Queue: queue
Description
Authorization Rules
Messages
Deadletter
Message List
MessageId
Seq
Size
Label
EnqueuedTimeUtc
ExpiresAtUtc
e11399faea7445249e520e...
1
227
14.06.2019 21:12
14.06.2019 21:12
dfc3c370cba04145b4d89e...
2
270
14.06.2019 21:12
31.12.9999 23:59
cc10d022cceb4944b359e1...
3
207
14.06.2019 21:12
31.12.9999 23:59
Message System Properties
Message Text
1
Delivery Count
Message Custom Properties
Name
Value
DeadLetterRea...
MaxDeliveryCountExceeded
DeadLetterError...
Message could not be consumed after 1 delivery atte...
Purge
Purge DLQ
Messages
Deadletter
Transf DLQ
Refresh
Status
Delete
Update
Log
<22:42:16> The queue queue has been successfully retrieved.
<22:42:17> The queue queue has been successfully retrieved.
<22:42:17> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:18> The queue queue has been successfully retrieved.
<22:42:19> The queue queue has been successfully retrieved.
<22:42:19> The queue queue has been successfully retrieved.
<22:42:19> The queue queue has been successfully retrieved.
<22:44:42> The queue queue has been successfully retrieved.
<22:44:50> [0] messages peeked from the queue [queue].
<23:11:32> The queue queue has been successfully retrieved.
<23:12:18> The queue queue has been successfully retrieved.
<23:12:20> The queue queue has been successfully retrieved.
<23:12:26> [3] messages peeked from the deadletter queue of the queue [queue].

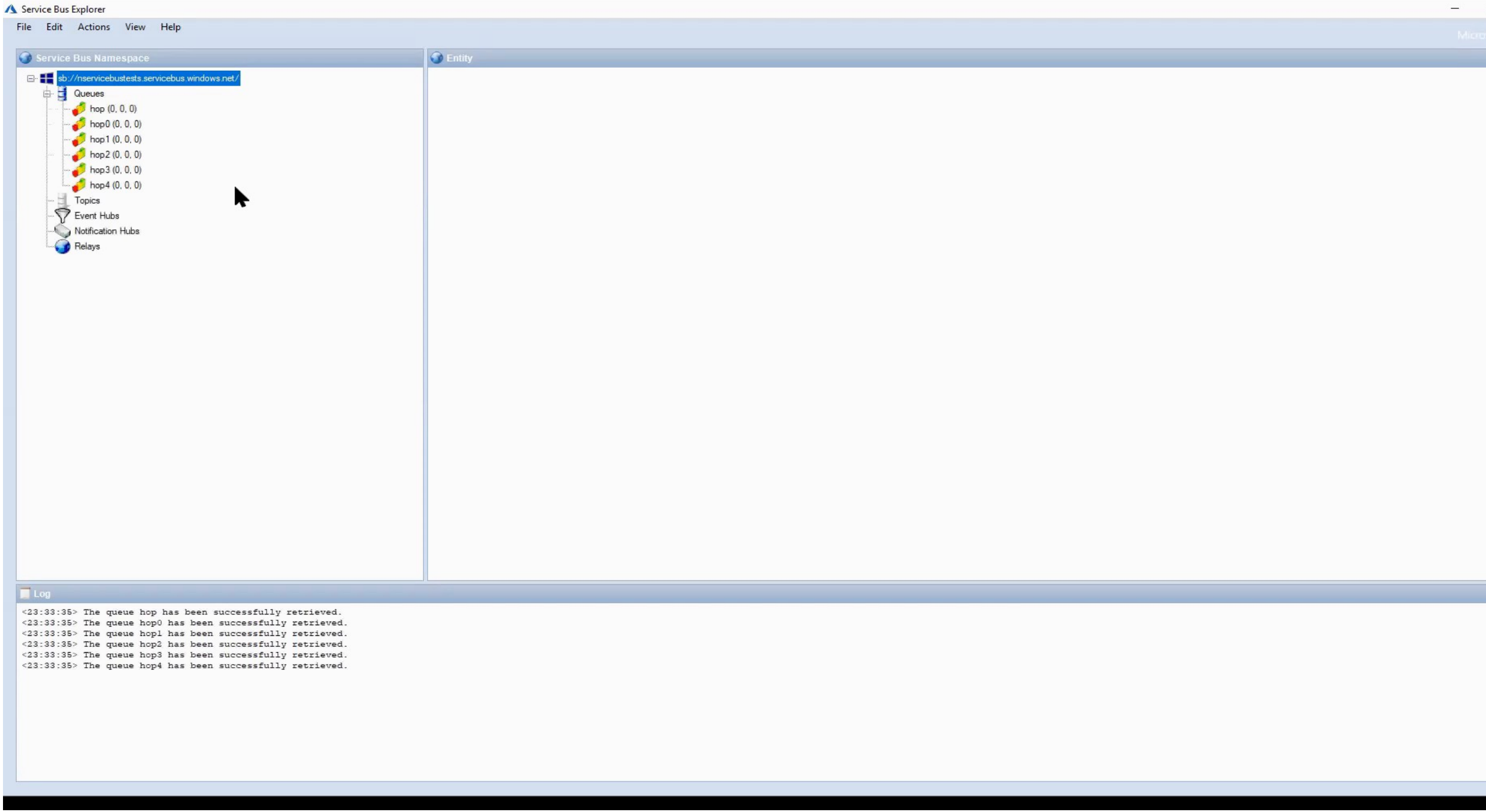
Forwarding

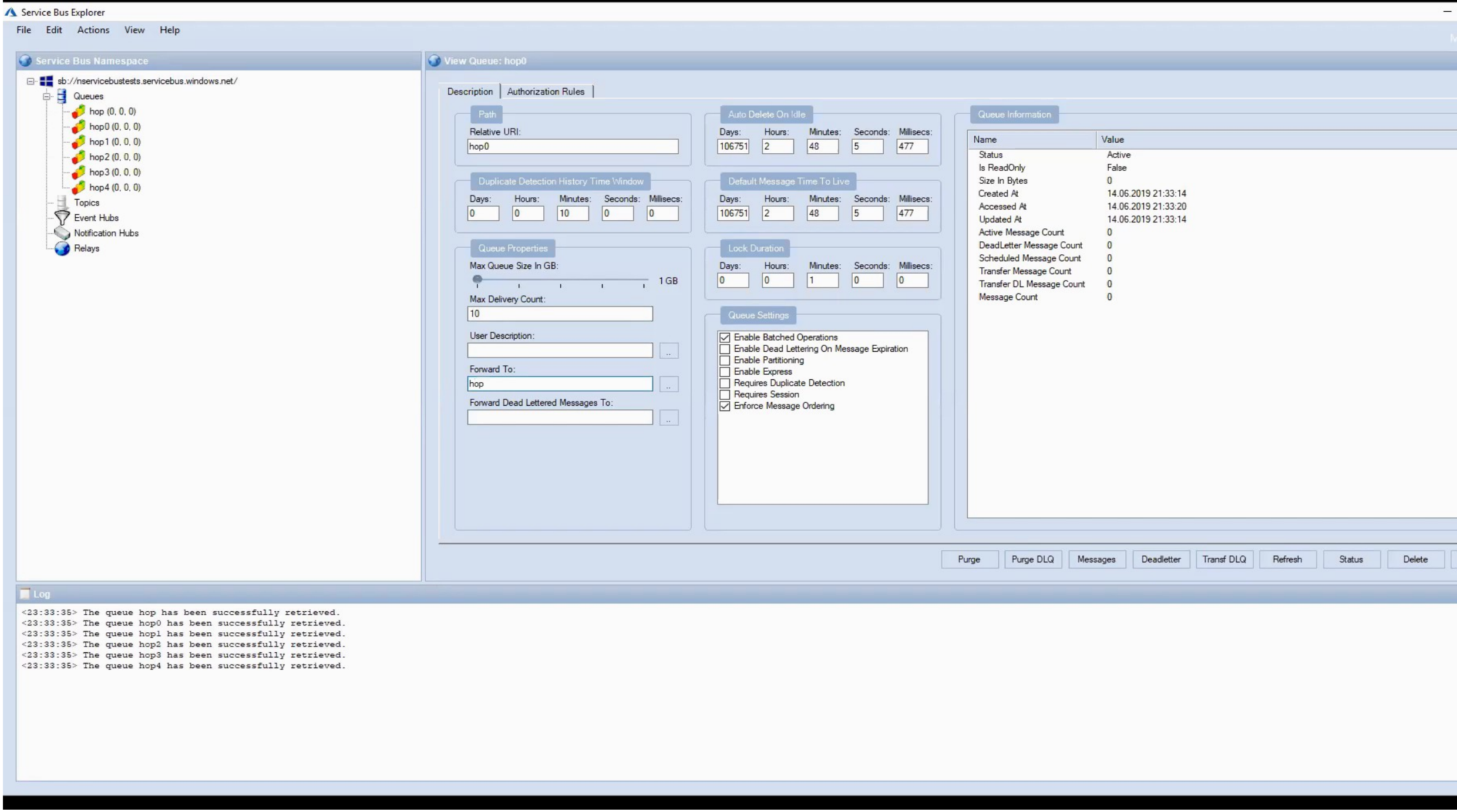

Hop4
Hop3
Hop2
Hop1
Hop0
Hop
ForwardTo

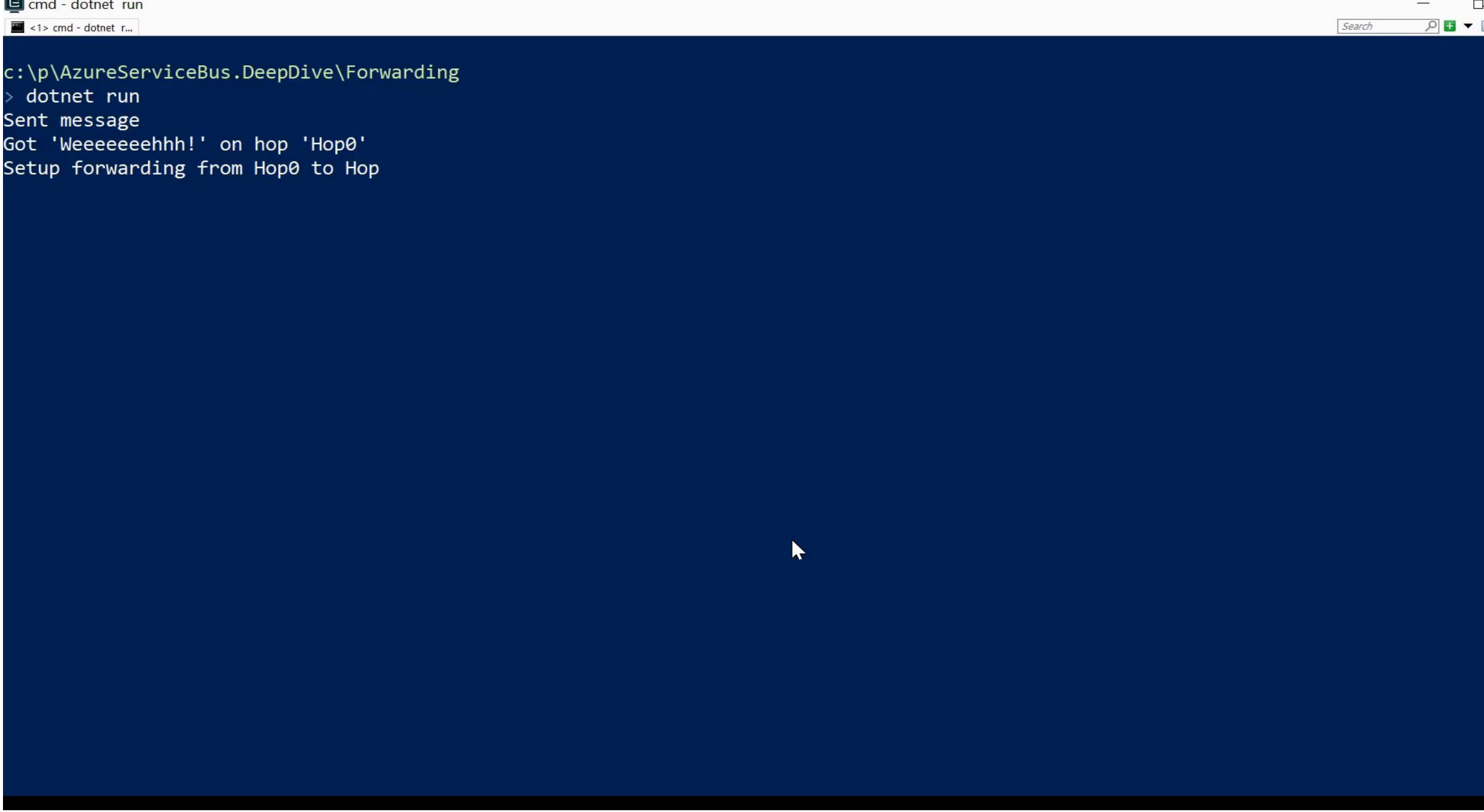
```
var description = new CreateQueueOptions("Hop");
await client.CreateQueueAsync(description);

description = new  CreateQueueOptions("Hop0");
await client.CreateQueueAsync(description);

description = new  CreateQueueOptions("Hop1")
{
    ForwardTo = "Hop0"
};
await client.CreateQueueAsync(description);

description = new  CreateQueueOptions("Hop2")
{
    ForwardTo = "Hop1"
};
await client.CreateQueueAsync(description);

description = new  CreateQueueOptions("Hop3")
{
    ForwardTo = "Hop2"
};
await client.CreateQueueAsync(description);

description = new  CreateQueueOptions("Hop4")
{
    ForwardTo = "Hop3"
};
await client.CreateQueueAsync(description);
```

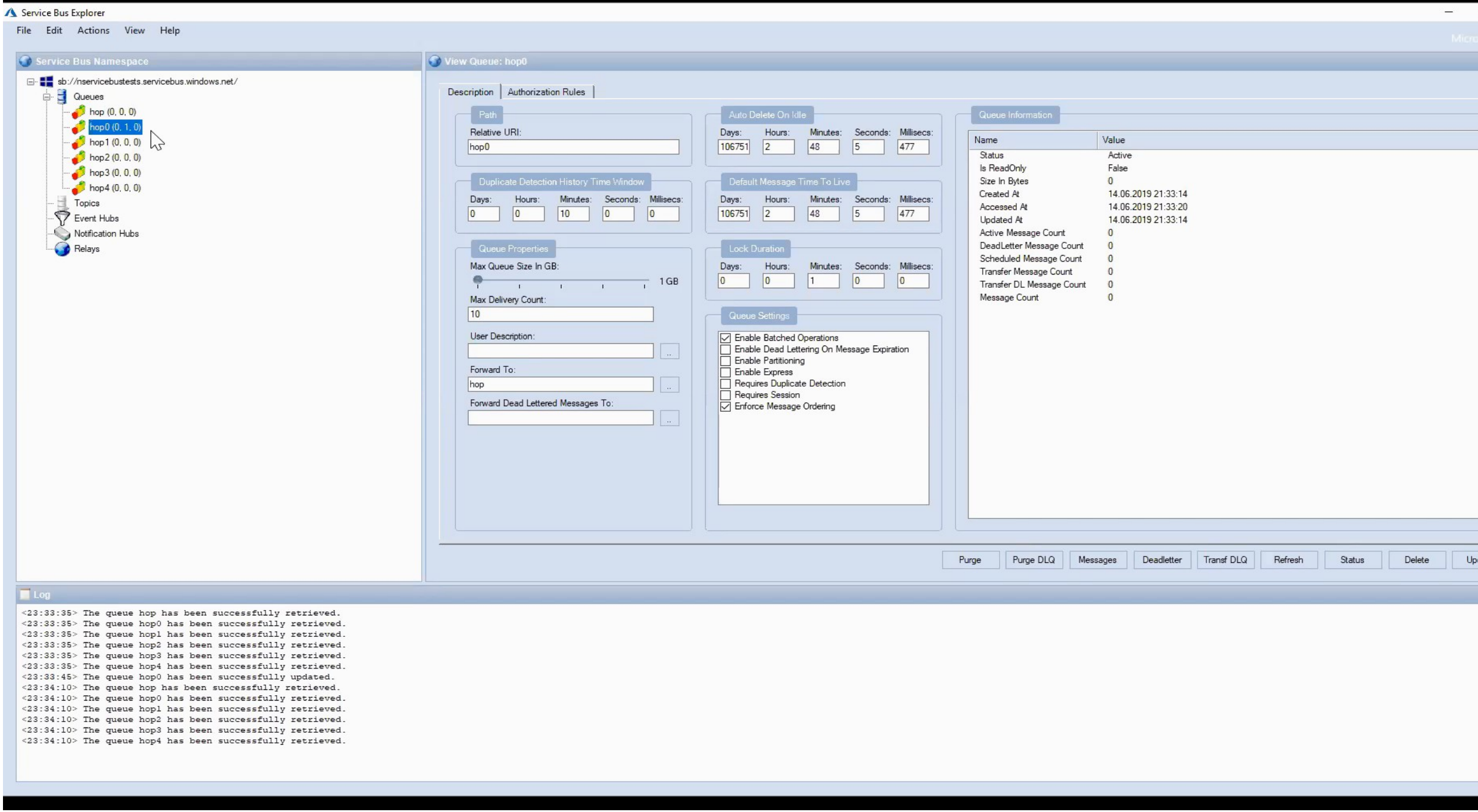
```
await using var serviceBusClient =
      new ServiceBusClient(connectionString);
var sender = serviceBusClient.CreateSender("Hop4");

var message = new ServiceBusMessage("Weeeeeeeehhh!");
await sender.SendMessageAsync(message);
```


cmd - dotnet run
<1> cmd - dotnet r...
Search
c:\p\AzureServiceBus.DeepDive\Forwarding
> dotnet run
Sent message
Got 'Weeeeeeehhh!' on hop 'Hop0'
Setup forwarding from Hop0 to Hop

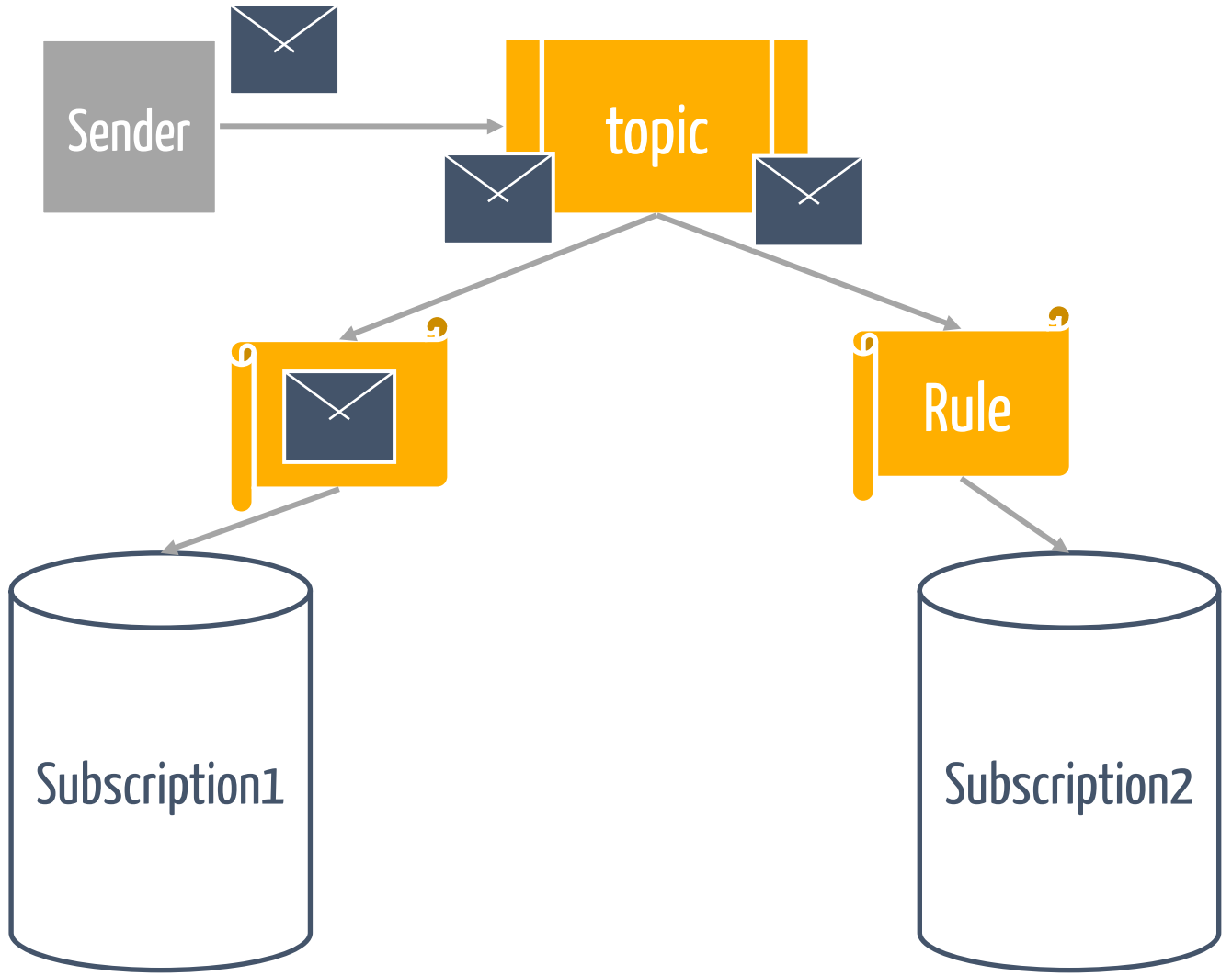

Service Bus Explorer
File
Edit
Actions
View
Help
Service Bus Namespace
sb://nservicebustests.servicebus.windows.net/
Queues
hop (0, 0, 0)
hop0 (0, 0, 0)
hop1 (0, 0, 0)
hop2 (0, 0, 0)
hop3 (0, 0, 0)
hop4 (0, 0, 0)
Topics
Event Hubs
Notification Hubs
Relays
Entity
Log
<23:33:35> The queue hop has been successfully retrieved.
<23:33:35> The queue hop0 has been successfully retrieved.
<23:33:35> The queue hop1 has been successfully retrieved.
<23:33:35> The queue hop2 has been successfully retrieved.
<23:33:35> The queue hop3 has been successfully retrieved.
<23:33:35> The queue hop4 has been successfully retrieved.

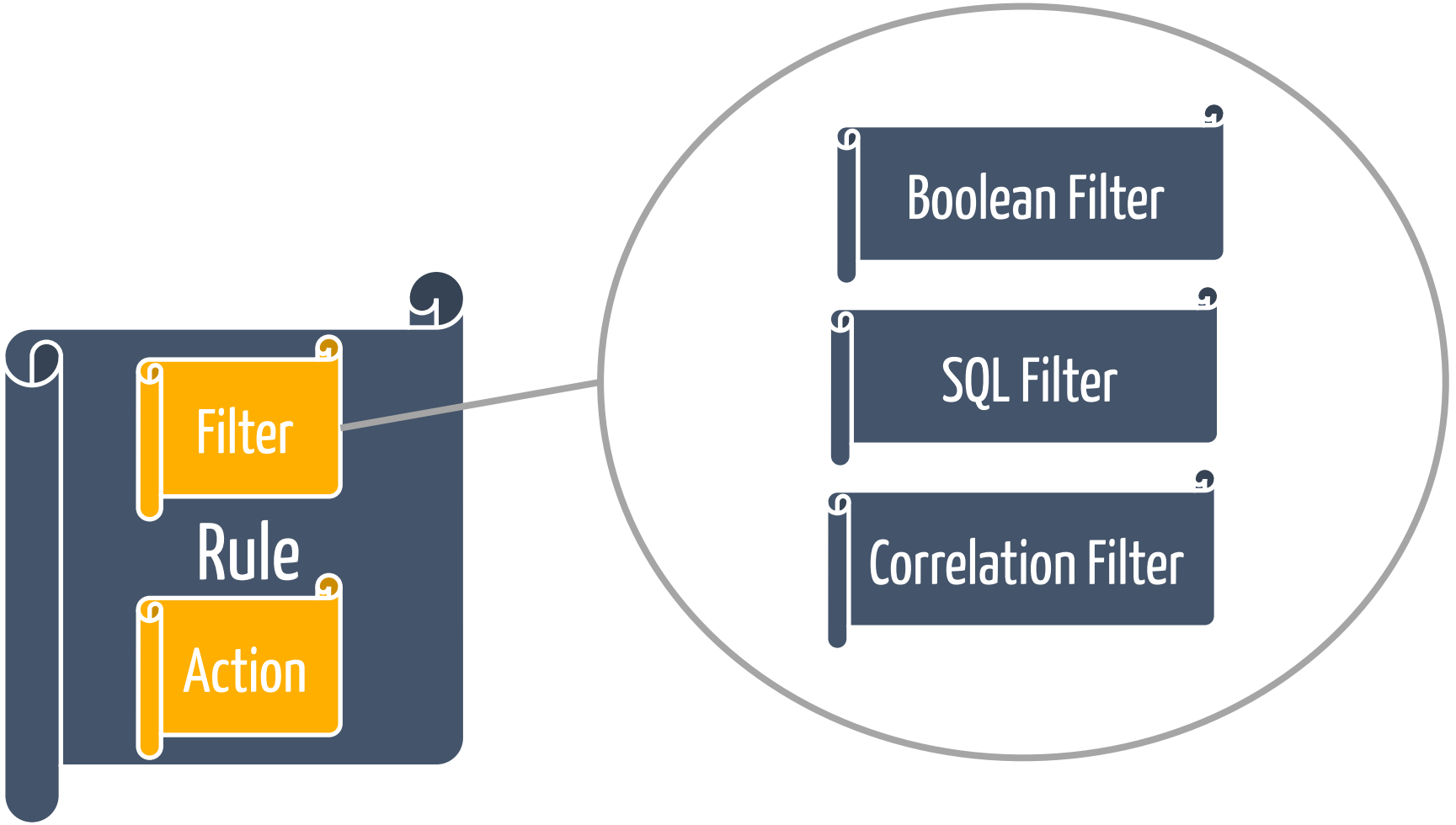

Service Bus Explorer
File
Edit
Actions
View
Help
Service Bus Namespace
sb://nservicebustests.servicebus.windows.net/
Queues
hop (0, 0, 0)
hop0 (0, 0, 0)
hop1 (0, 0, 0)
hop2 (0, 0, 0)
hop3 (0, 0, 0)
hop4 (0, 0, 0)
Topics
Event Hubs
Notification Hubs
Relays
View Queue: hop0
Description
Authorization Rules
Path
Relative URI:
hop0
Auto Delete On Idle
Days: 106751
Hours: 2
Minutes: 48
Seconds: 5
Millisecs: 477
Duplicate Detection History Time Window
Days: 0
Hours: 0
Minutes: 10
Seconds: 0
Millisecs: 0
Default Message Time To Live
Days: 106751
Hours: 2
Minutes: 48
Seconds: 5
Millisecs: 477
Queue Properties
Max Queue Size In GB:
1 GB
Max Delivery Count:
10
User Description:
Forward To:
hop
Forward Dead Lettered Messages To:
Lock Duration
Days: 0
Hours: 0
Minutes: 1
Seconds: 0
Millisecs: 0
Queue Settings
Enable Batched Operations
Enable Dead Lettering On Message Expiration
Enable Partitioning
Enable Express
Requires Duplicate Detection
Requires Session
Enforce Message Ordering
Queue Information
Name
Value
Status
Active
Is ReadOnly
False
Size In Bytes
0
Created At
14.06.2019 21:33:14
Accessed At
14.06.2019 21:33:20
Updated At
14.06.2019 21:33:14
Active Message Count
0
DeadLetter Message Count
0
Scheduled Message Count
0
Transfer Message Count
0
Transfer DL Message Count
0
Message Count
0
Purge
Purge DLQ
Messages
Deadletter
Transf DLQ
Refresh
Status
Delete
Log
<23:33:35> The queue hop has been successfully retrieved.
<23:33:35> The queue hop0 has been successfully retrieved.
<23:33:35> The queue hop1 has been successfully retrieved.
<23:33:35> The queue hop2 has been successfully retrieved.
<23:33:35> The queue hop3 has been successfully retrieved.
<23:33:35> The queue hop4 has been successfully retrieved.


cmd - dotnet run
<1> cmd - dotnet r...
Search
c:\p\AzureServiceBus.DeepDive\Forwarding
> dotnet run
Sent message
Got 'Weeeeeeehhh!' on hop 'Hop0'
Setup forwarding from Hop0 to Hop


Service Bus Explorer
File
Edit
Actions
View
Help
Service Bus Namespace
sb://nservicebustests.servicebus.windows.net/
Queues
hop (0, 0, 0)
hop0 (0, 1, 0)
hop1 (0, 0, 0)
hop2 (0, 0, 0)
hop3 (0, 0, 0)
hop4 (0, 0, 0)
Topics
Event Hubs
Notification Hubs
Relays
View Queue: hop0
Description
Authorization Rules
Path
Relative URI:
hop0
Duplicate Detection History Time Window
Days: Hours: Minutes: Seconds: Millisecs:
0 0 10 0 0
Queue Properties
Max Queue Size In GB:
1 GB
Max Delivery Count:
10
User Description:
Forward To:
hop
Forward Dead Lettered Messages To:
Auto Delete On Idle
Days: Hours: Minutes: Seconds: Millisecs:
106751 2 48 5 477
Default Message Time To Live
Days: Hours: Minutes: Seconds: Millisecs:
106751 2 48 5 477
Lock Duration
Days: Hours: Minutes: Seconds: Millisecs:
0 0 1 0 0
Queue Settings
Enable Batched Operations
Enable Dead Lettering On Message Expiration
Enable Partitioning
Enable Express
Requires Duplicate Detection
Requires Session
Enforce Message Ordering
Queue Information
Name
Value
Status
Active
Is ReadOnly
False
Size In Bytes
0
Created At
14.06.2019 21:33:14
Accessed At
14.06.2019 21:33:20
Updated At
14.06.2019 21:33:14
Active Message Count
0
DeadLetter Message Count
0
Scheduled Message Count
0
Transfer Message Count
0
Transfer DL Message Count
0
Message Count
0
Purge
Purge DLQ
Messages
Deadletter
Transf DLQ
Refresh
Status
Delete
Log
<23:33:35> The queue hop has been successfully retrieved.
<23:33:35> The queue hop0 has been successfully retrieved.
<23:33:35> The queue hop1 has been successfully retrieved.
<23:33:35> The queue hop2 has been successfully retrieved.
<23:33:35> The queue hop3 has been successfully retrieved.
<23:33:35> The queue hop4 has been successfully retrieved.
<23:33:45> The queue hop0 has been successfully updated.
<23:34:10> The queue hop has been successfully retrieved.
<23:34:10> The queue hop0 has been successfully retrieved.
<23:34:10> The queue hop1 has been successfully retrieved.
<23:34:10> The queue hop2 has been successfully retrieved.
<23:34:10> The queue hop3 has been successfully retrieved.
<23:34:10> The queue hop4 has been successfully retrieved.

Pub/Sub


Sender
topic
Rule
Subscription1
Subscription2

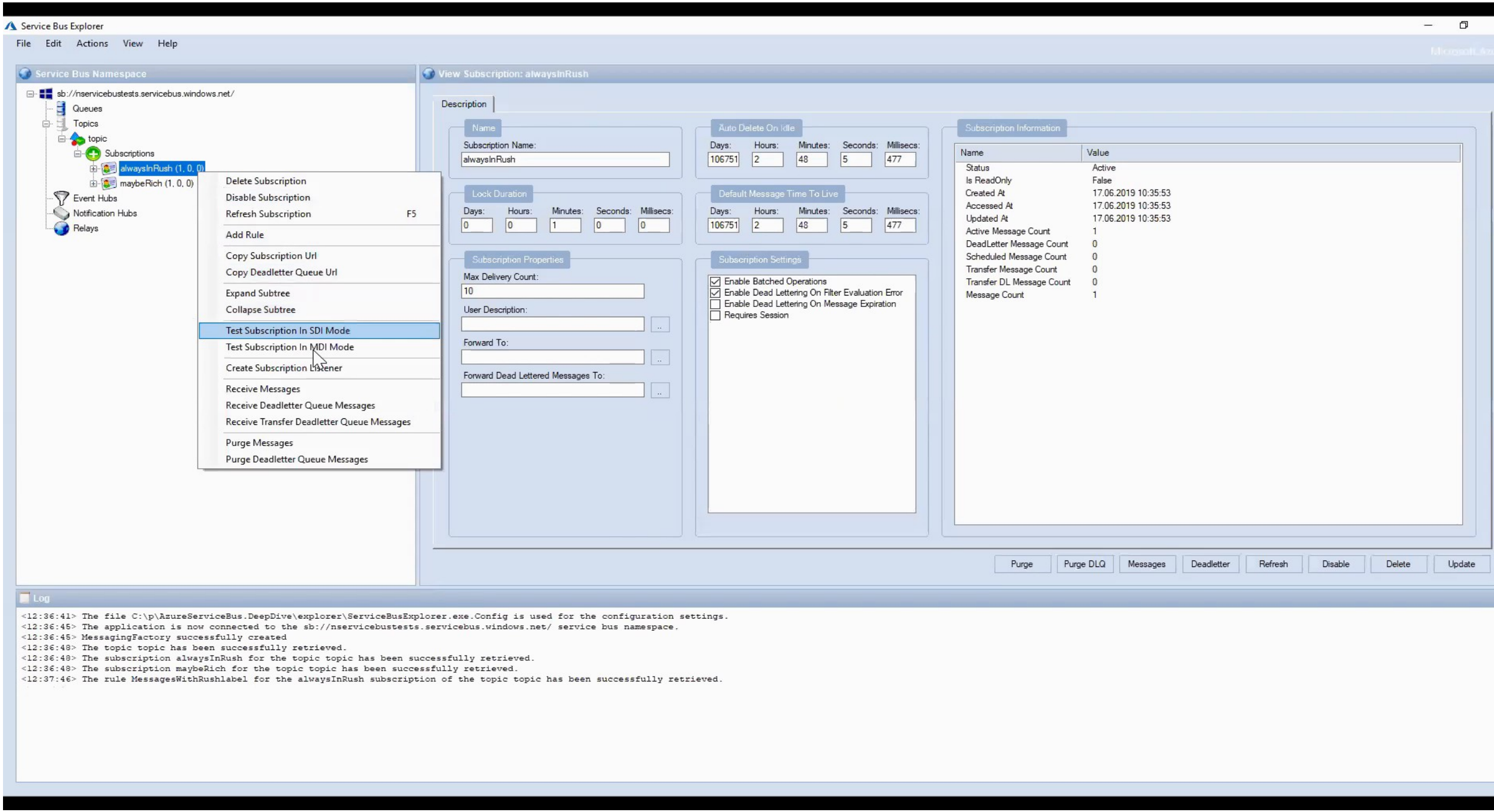

Filter
Rule
Action
Boolean Filter
SQL Filter
Correlation Filter

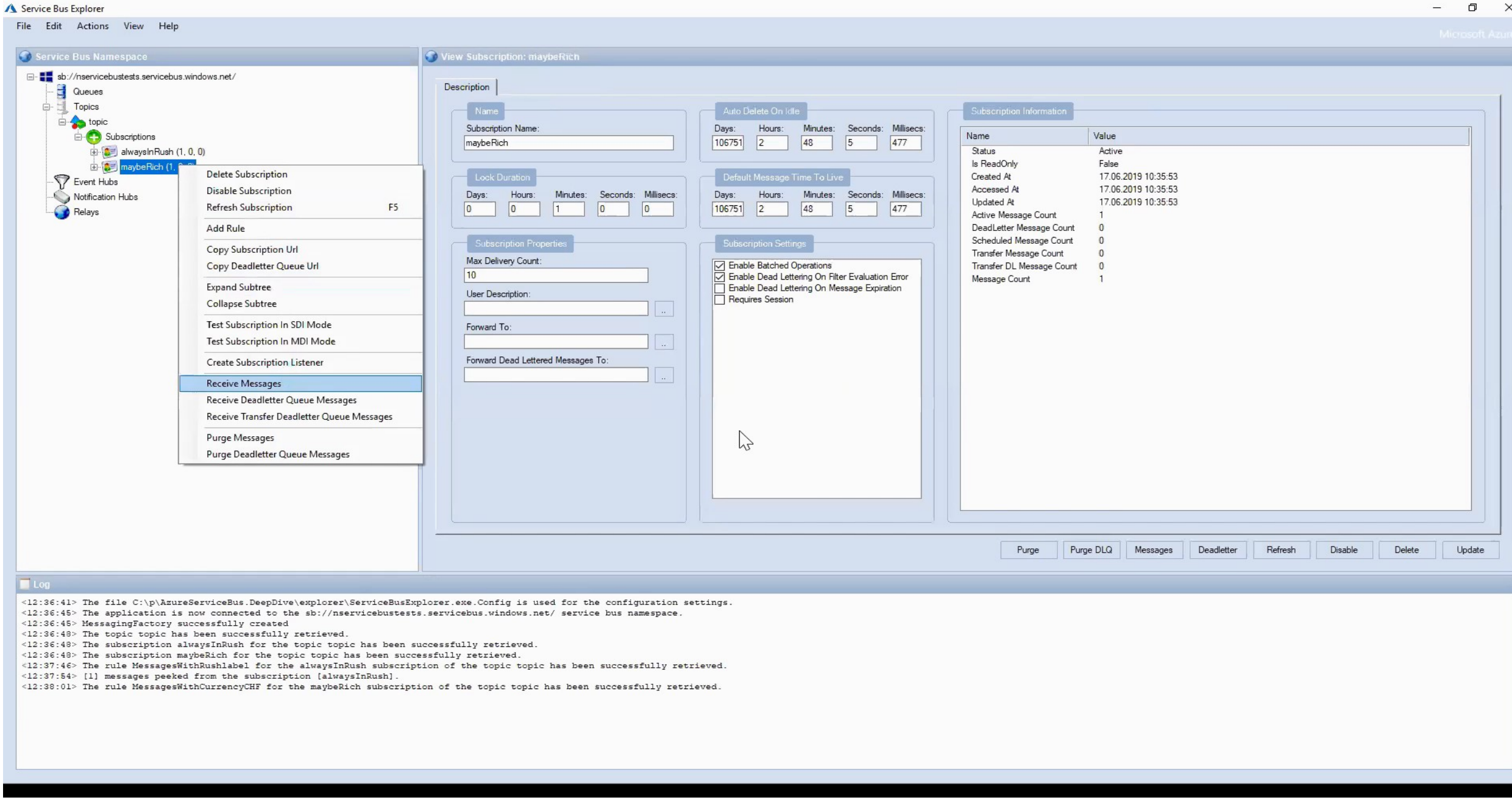
```
var topicDescription = new CreateTopicOptions (topicName);
await client.CreateTopicAsync(topicDescription);

var subscriptionDescription =
      new CreateSubscriptionOptions(topicName, rushSubscription);
await client.CreateSubscriptionAsync(subscriptionDescription);

subscriptionDescription =
new CreateSubscriptionOptions(topicName, currencySubscription);
await client.CreateSubscriptionAsync(subscriptionDescription);
```

```
var ruleDescription = new CreateRuleOptions
{
    Name = "MessagesWithRushlabel",
    Filter = new CorrelationRuleFilter
    {
        Subject = "rush"
    },
    Action = null
};
await client.
     CreateRuleAsync(topicName, rushSubscription, ruleDescription);
```

```
ruleDescription = new CreateRuleOptions
{
    Name = "MessagesWithCurrencyCHF",
    Filter = new SqlRuleFilter("currency = 'CHF'"),
    Action = new SqlRuleAction("SET currency = 'Złoty'")
};
await client
 .CreateRuleAsync(topicName, currencySubscription, ruleDescription);
```

```
var message = new ServiceBusMessage("Damn I have no time!")
{
    Subject = "rush"
};
await sender.SendMessageAsync(message);

message = new ServiceBusMessage("I'm rich! I have 1000");
message.ApplicationProperties.Add("currency", "CHF");
await sender.SendMessageAsync(message);
```

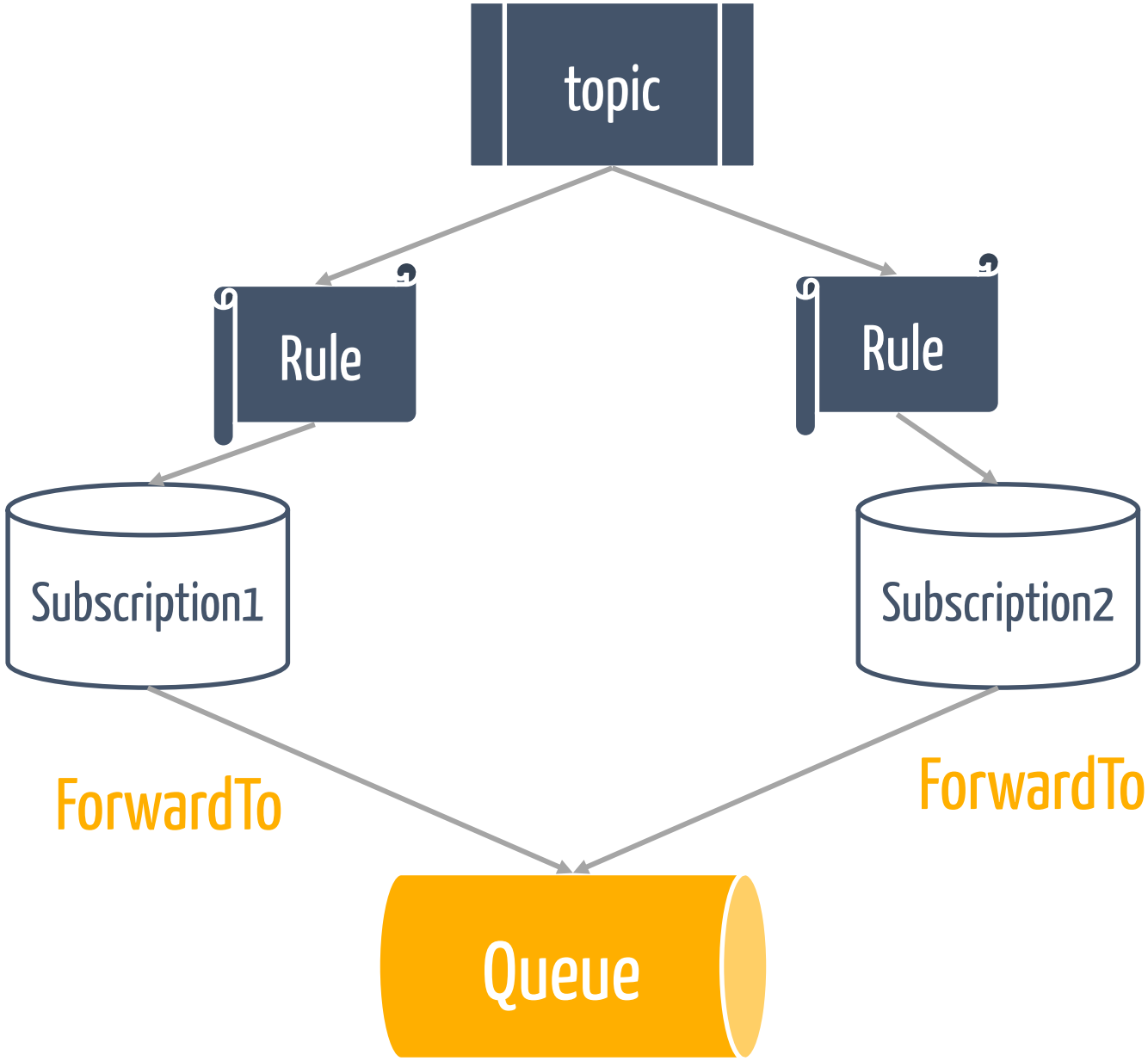

Service Bus Explorer
File
Edit
Actions
View
Help
Service Bus Namespace
sb://nservicebustests.servicebus.windows.net/
Queues
Topics
topic
Subscriptions
alwaysInRush (1, 0, 0)
maybeRich (1, 0, 0)
Event Hubs
Notification Hubs
Relays
Delete Subscription
Disable Subscription
Refresh Subscription F5
Add Rule
Copy Subscription Url
Copy Deadletter Queue Url
Expand Subtree
Collapse Subtree
Test Subscription In SDI Mode
Test Subscription In MDI Mode
Create Subscription Listener
Receive Messages
Receive Deadletter Queue Messages
Receive Transfer Deadletter Queue Messages
Purge Messages
Purge Deadletter Queue Messages
View Subscription: alwaysInRush
Description
Name
Subscription Name:
alwaysInRush
Auto Delete On Idle
Days: 106751
Hours: 2
Minutes: 48
Seconds: 5
Milliseconds: 477
Lock Duration
Days: 0
Hours: 0
Minutes: 1
Seconds: 0
Millisecs: 0
Default Message Time To Live
Days: 106751
Hours: 2
Minutes: 48
Seconds: 5
Milliseconds: 477
Subscription Properties
Max Delivery Count:
10
User Description:
Forward To:
Forward Dead Lettered Messages To:
Subscription Settings
Enable Batched Operations
Enable Dead Lettering On Filter Evaluation Error
Enable Dead Lettering On Message Expiration
Requires Session
Subscription Information
Name
Value
Status Active
Is ReadOnly False
Created At 17.06.2019 10:35:53
Accessed At 17.06.2019 10:35:53
Updated At 17.06.2019 10:35:53
Active Message Count 1
DeadLetter Message Count 0
Scheduled Message Count 0
Transfer Message Count 0
Transfer DL Message Count 0
Message Count 1
Purge
Purge DLQ
Messages
Deadletter
Refresh
Disable
Delete
Update
Log
<12:36:41> The file C:\p\AzureServiceBus.DeepDive\explorer\ServiceBusExplorer.exe.Config is used for the configuration settings.
<12:36:45> The application is now connected to the sb://nservicebustests.servicebus.windows.net/ service bus namespace.
<12:36:45> MessagingFactory successfully created
<12:36:48> The topic topic has been successfully retrieved.
<12:36:48> The subscription alwaysInRush for the topic topic has been successfully retrieved.
<12:36:48> The subscription maybeRich for the topic topic has been successfully retrieved.
<12:37:46> The rule MessagesWithRushlabel for the alwaysInRush subscription of the topic topic has been successfully retrieved.


Service Bus Explorer
File Edit Actions View Help
Service Bus Namespace
sb://nservicebustests.servicebus.windows.net/
Queues
Topics
topic
Subscriptions
alwaysInRush (1, 0, 0)
maybeRich (1,
Event Hubs
Notification Hubs
Relays
Delete Subscription
Disable Subscription
Refresh Subscription F5
Add Rule
Copy Subscription Url
Copy Deadletter Queue Url
Expand Subtree
Collapse Subtree
Test Subscription In SDI Mode
Test Subscription In MDI Mode
Create Subscription Listener
Receive Messages
Receive Deadletter Queue Messages
Receive Transfer Deadletter Queue Messages
Purge Messages
Purge Deadletter Queue Messages
View Subscription: maybeRich
Description
Name
Subscription Name:
maybeRich
Auto Delete On Idle
Days: 106751 Hours: 2 Minutes: 48 Seconds: 5 Millisecs: 477
Lock Duration
Days: 0 Hours: 0 Minutes: 1 Seconds: 0 Millisecs: 0
Default Message Time To Live
Days: 106751 Hours: 2 Minutes: 48 Seconds: 5 Millisecs: 477
Subscription Properties
Max Delivery Count:
10
User Description:
Forward To:
Forward Dead Lettered Messages To:
Subscription Settings
Enable Batched Operations
Enable Dead Lettering On Filter Evaluation Error
Enable Dead Lettering On Message Expiration
Requires Session
Subscription Information
Name Value
Status Active
Is ReadOnly False
Created At 17.06.2019 10:35:53
Accessed At 17.06.2019 10:35:53
Updated At 17.06.2019 10:35:53
Active Message Count 1
DeadLetter Message Count 0
Scheduled Message Count 0
Transfer Message Count 0
Transfer DL Message Count 0
Message Count 1
Purge
Purge DLQ
Messages
Deadletter
Refresh
Disable
Delete
Update
Log
<12:36:41> The file C:\p\AzureServiceBus.DeepDive\explorer\ServiceBusExplorer.exe.Config is used for the configuration settings.
<12:36:45> The application is now connected to the sb://nservicebustests.servicebus.windows.net/ service bus namespace.
<12:36:45> MessagingFactory successfully created
<12:36:48> The topic topic has been successfully retrieved.
<12:36:48> The subscription alwaysInRush for the topic topic has been successfully retrieved.
<12:36:48> The subscription maybeRich for the topic topic has been successfully retrieved.
<12:37:46> The rule MessagesWithRushlabel for the alwaysInRush subscription of the topic topic has been successfully retrieved.
<12:37:54> [1] messages peeked from the subscription [alwaysInRush].
<12:38:01> The rule MessagesWithCurrencyCHF for the maybeRich subscription of the topic topic has been successfully retrieved.

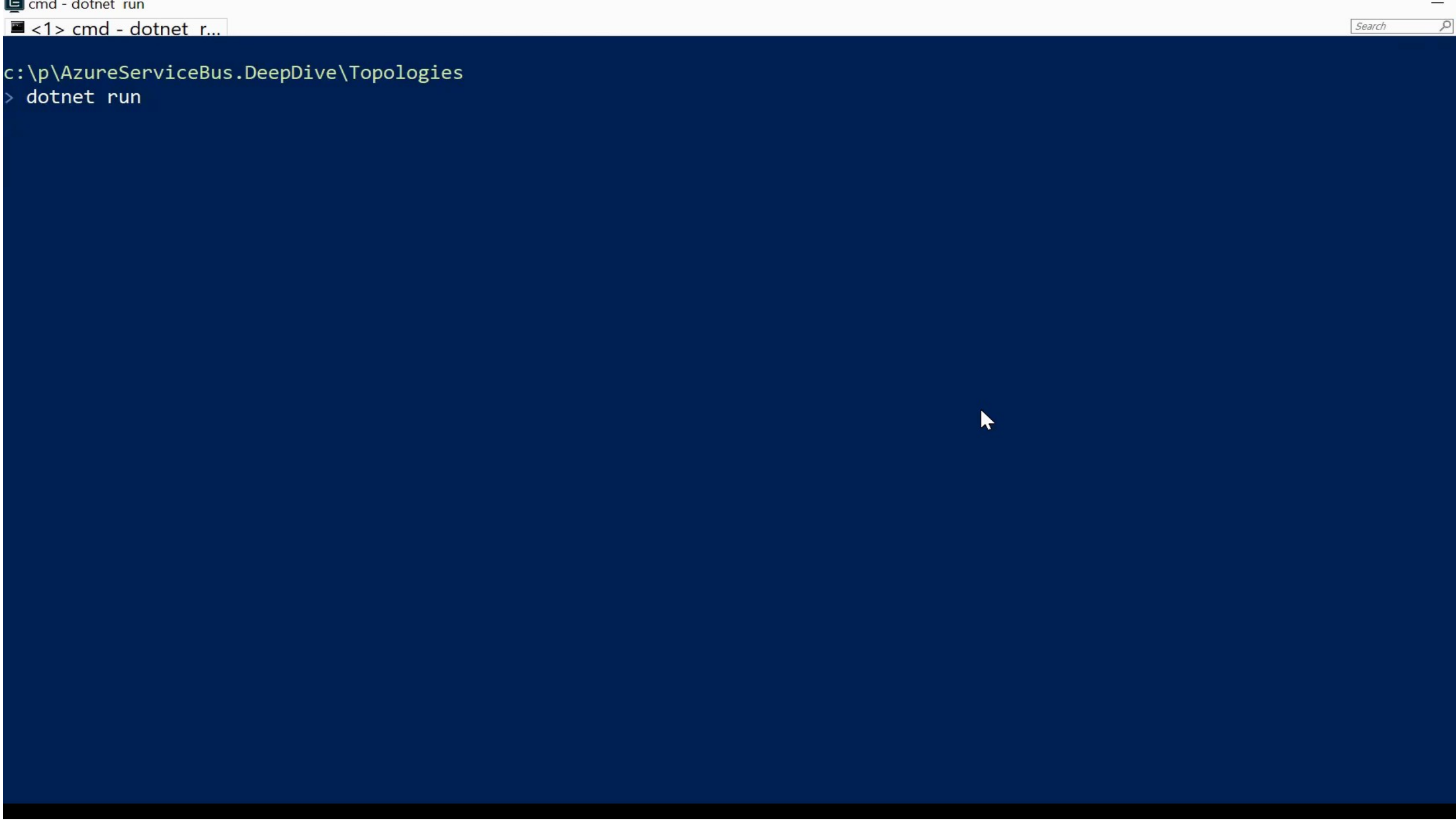
Favor Correlation filter over SQL filter

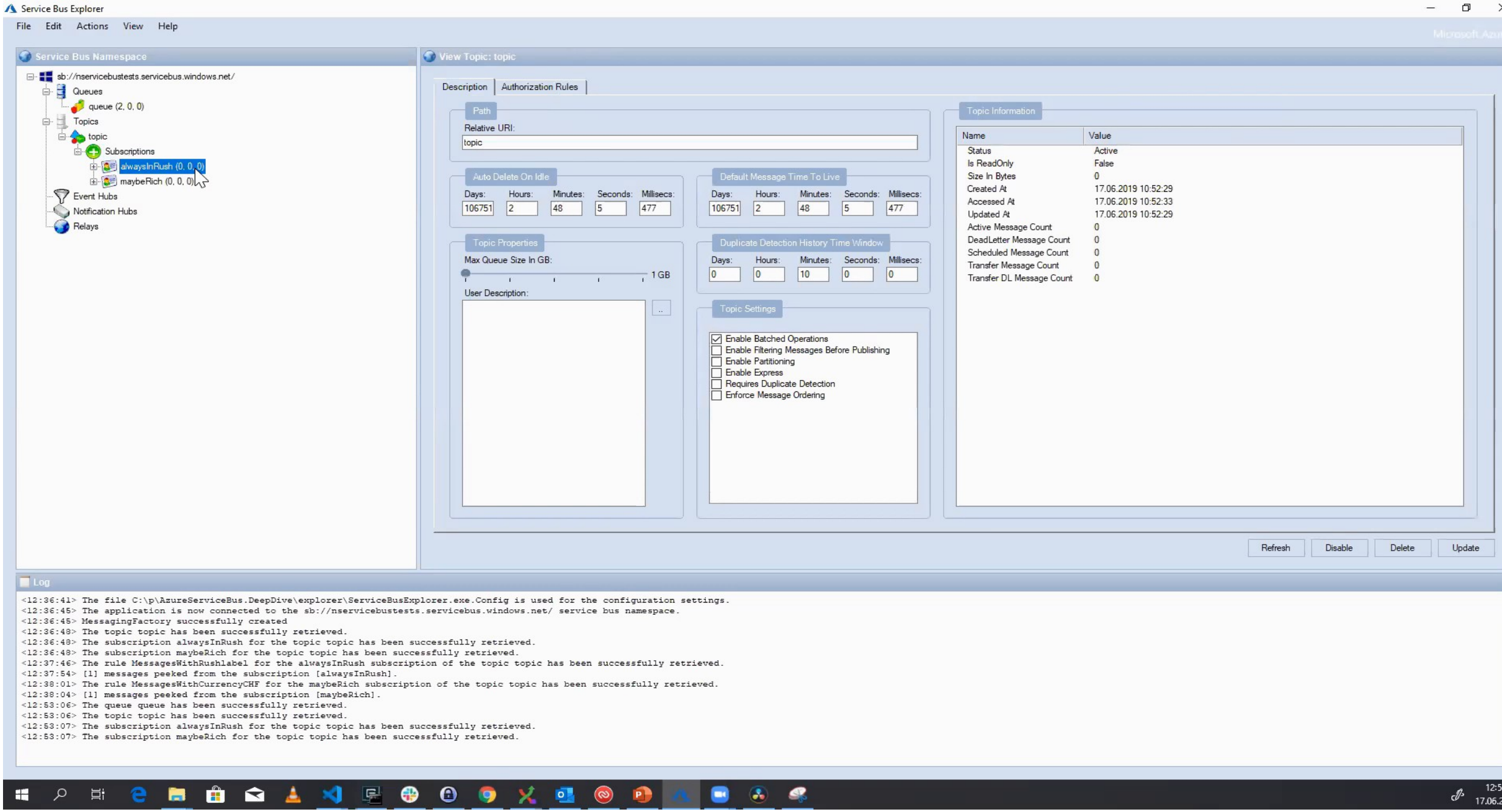
Subscriptions are virtual queues and subscribers need to receive from them

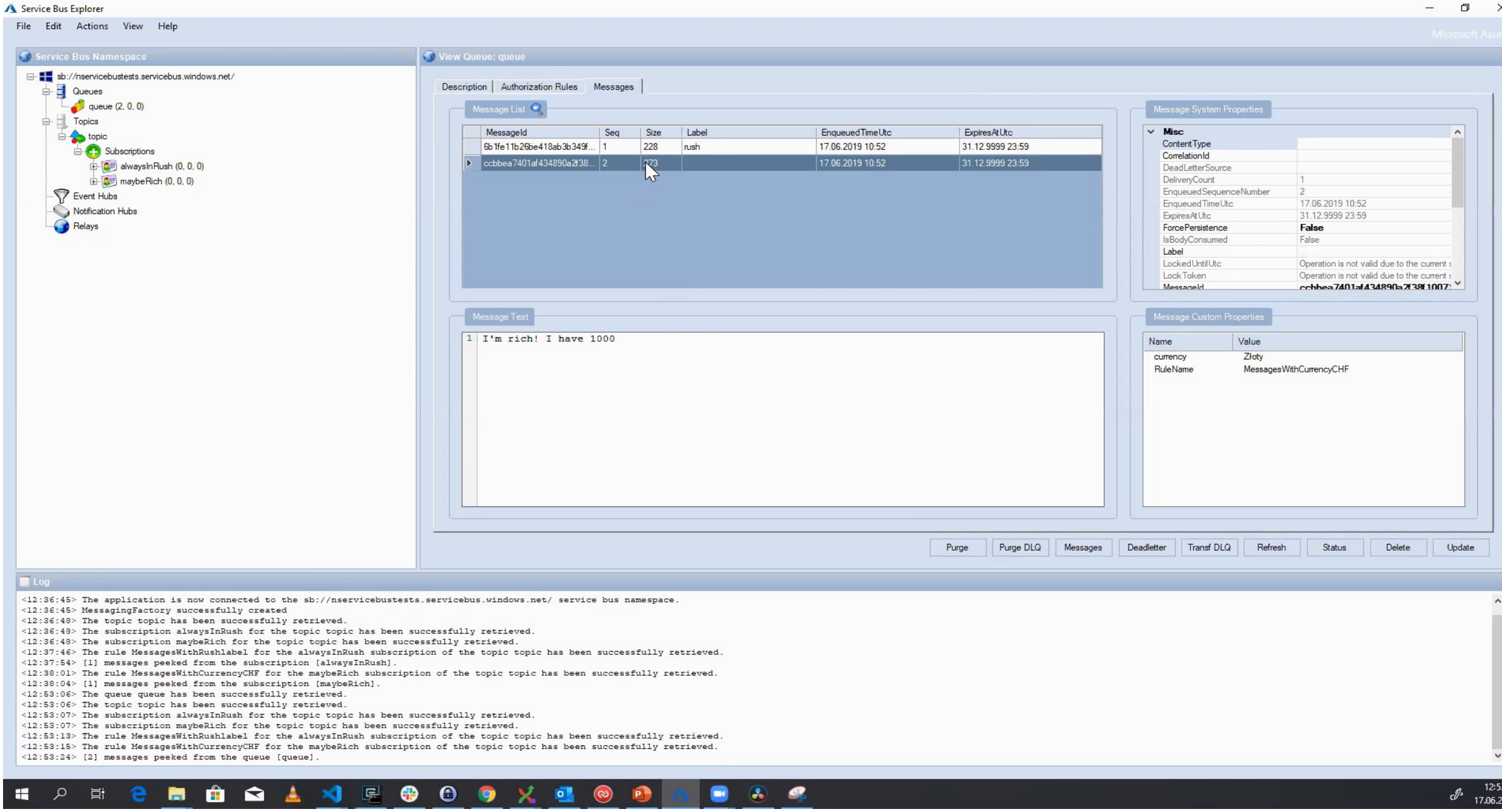
Topologies


topic
Rule
Rule
Subscription1
Subscription2
ForwardTo
ForwardTo
Queue

```
var subscriptionDescription =
            new CreateSubscriptionOptions(topicName, rushSubscription)
    {
        ForwardTo = inputQueue
    };
    await client.CreateSubscriptionAsync(subscriptionDescription);

    subscriptionDescription =

    new CreateSubscriptionOptions(topicName, currencySubscription)
    {
        ForwardTo = inputQueue
    };
    await client.CreateSubscriptionAsync(subscriptionDescription);
```


cmd - dotnet run
<1> cmd - dotnet r...
Search
c:\p\AzureServiceBus.DeepDive\Topologies
> dotnet run


Service Bus Explorer
File
Edit
Actions
View
Help
Service Bus Namespace
sb://nservicebustests.servicebus.windows.net/
Queues
queue (2, 0, 0)
Topics
topic
Subscriptions
alwaysInRush (0, 0, 0)
maybeRich (0, 0, 0)
Event Hubs
Notification Hubs
Relays
View Topic: topic
Description
Authorization Rules
Path
Relative URI:
topic
Auto Delete On Idle
Days: 106751
Hours: 2
Minutes: 48
Seconds: 5
Millisecs: 477
Default Message Time To Live
Days: 106751
Hours: 2
Minutes: 48
Seconds: 5
Millisecs: 477
Topic Properties
Max Queue Size In GB:
1 GB
User Description:
Duplicate Detection History Time Window
Days: 0
Hours: 0
Minutes: 10
Seconds: 0
Millisecs: 0
Topic Settings
Enable Batched Operations
Enable Filtering Messages Before Publishing
Enable Partitioning
Enable Express
Requires Duplicate Detection
Enforce Message Ordering
Topic Information
Name
Value
Status Active
Is ReadOnly False
Size In Bytes 0
Created At 17.06.2019 10:52:29
Accessed At 17.06.2019 10:52:33
Updated At 17.06.2019 10:52:29
Active Message Count 0
DeadLetter Message Count 0
Scheduled Message Count 0
Transfer Message Count 0
Transfer DL Message Count 0
Refresh
Disable
Delete
Update
Log
<12:36:41> The file C:\p\AzureServiceBus.DeepDive\explorer\ServiceBusExplorer.exe.Config is used for the configuration settings.
<12:36:45> The application is now connected to the sb://nservicebustests.servicebus.windows.net/ service bus namespace.
<12:36:45> MessagingFactory successfully created
<12:36:48> The topic topic has been successfully retrieved.
<12:36:48> The subscription alwaysInRush for the topic topic has been successfully retrieved.
<12:36:48> The subscription maybeRich for the topic topic has been successfully retrieved.
<12:37:46> The rule MessagesWithRushlabel for the alwaysInRush subscription of the topic topic has been successfully retrieved.
<12:37:54> [1] messages peeked from the subscription [alwaysInRush].
<12:38:01> The rule MessagesWithCurrencyCHF for the maybeRich subscription of the topic topic has been successfully retrieved.
<12:38:04> [1] messages peeked from the subscription [maybeRich].
<12:53:06> The queue queue has been successfully retrieved.
<12:53:06> The topic topic has been successfully retrieved.
<12:53:07> The subscription alwaysInRush for the topic topic has been successfully retrieved.
<12:53:07> The subscription maybeRich for the topic topic has been successfully retrieved.


Service Bus Explorer
File Edit Actions View Help
Service Bus Namespace
sb://nservicebustests.servicebus.windows.net/
Queues
queue (2, 0, 0)
Topics
topic
Subscriptions
alwaysInRush (0, 0, 0)
maybeRich (0, 0, 0)
Event Hubs
Notification Hubs
Relays
View Queue: queue
Description
Authorization Rules
Messages
Message List
MessageId Seq Size Label EnqueuedTimeUtc ExpiresAtUtc
6b1fe11b26be418ab3b349f... 1 228 rush 17.06.2019 10:52 31.12.9999 23:59
ccbbea7401af434890a2f38... 2 273 17.06.2019 10:52 31.12.9999 23:59
Message System Properties
Misc
ContentType
CorrelationId
DeadLetterSource
DeliveryCount 1
EnqueuedSequenceNumber 2
EnqueuedTimeUtc 17.06.2019 10:52
ExpiresAtUtc 31.12.9999 23:59
ForcePersistence False
IsBodyConsumed False
Label
LockedUntilUtc Operation is not valid due to the current
LockToken Operation is not valid due to the current
MessageId ccbbea7401af434890a2f38f10071
Message Text
1 I'm rich! I have 1000
Message Custom Properties
Name Value
currency Zloty
RuleName MessagesWithCurrencyCHF
Purge
Purge DLQ
Messages
Deadletter
Transf DLQ
Refresh
Status
Delete
Update
Log
<12:36:45> The application is now connected to the sb://nservicebustests.servicebus.windows.net/ service bus namespace.
<12:36:45> MessagingFactory successfully created
<12:36:48> The topic topic has been successfully retrieved.
<12:36:48> The subscription alwaysInRush for the topic topic has been successfully retrieved.
<12:36:48> The subscription maybeRich for the topic topic has been successfully retrieved.
<12:37:46> The rule MessagesWithRushlabel for the alwaysInRush subscription of the topic topic has been successfully retrieved.
<12:37:54> [1] messages peeked from the subscription [alwaysInRush].
<12:38:01> The rule MessagesWithCurrencyCHF for the maybeRich subscription of the topic topic has been successfully retrieved.
<12:38:04> [1] messages peeked from the subscription [maybeRich].
<12:53:06> The queue queue has been successfully retrieved.
<12:53:06> The topic topic has been successfully retrieved.
<12:53:07> The subscription alwaysInRush for the topic topic has been successfully retrieved.
<12:53:07> The subscription maybeRich for the topic topic has been successfully retrieved.
<12:53:13> The rule MessagesWithRushlabel for the alwaysInRush subscription of the topic topic has been successfully retrieved.
<12:53:15> The rule MessagesWithCurrencyCHF for the maybeRich subscription of the topic topic has been successfully retrieved.
<12:53:24> [2] messages peeked from the queue [queue].

Atomic Sends

```
using (var scope =
      new TransactionScope(TransactionScopeAsyncFlowOption.Enabled))
{
    var message = new ServiceBusMessage("Deep Dive 1");
    await sender.SendMessageAsync(message);

    message = new ServiceBusMessage("Deep Dive 2");
    await sender.SendMessageAsync(message);

    scope.Complete();
}
```

cmd - dotnet run

<1> cmd - dotnet r...

Search

```
c:\p\AzureServiceBus.DeepDive\AtomicSend
> dotnet run
Sent message 1 in transaction '12c87d8b-87ad-4816-8218-7225fe9f850e:1'
#'0' messages in 'queue'
Sent message 2 in transaction '12c87d8b-87ad-4816-8218-7225fe9f850e:1'
About to complete transaction scope.
```

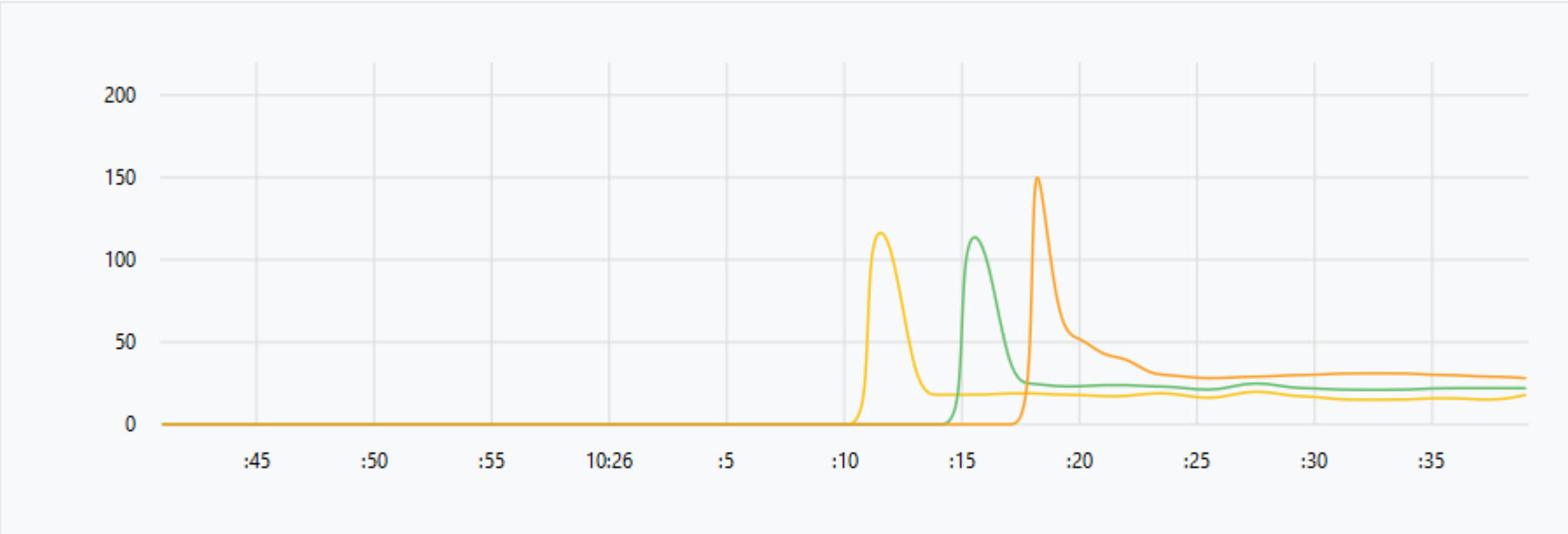
```
c:\p\AzureServiceBus.DeepDive\AtomicSend
> dotnet run
Sent message 1 in transaction '12c87d8b-87ad-4816-8218-7225fe9f850e:1'
#'0' messages in 'queue'
Sent message 2 in transaction '12c87d8b-87ad-4816-8218-7225fe9f850e:1'
About to complete transaction scope.
#'0' messages in 'queue'
Completed transaction scope.
#'2' messages in 'queue'
```

Transaction not completed

Transaction completed

Batching

Region	Average Latency (ms)
Switzerland North (Zurich)	18 ms
Germany West Central (Frankfurt)	22 ms
France Central (Paris)	28 ms



Latency Test

Geography	Region	Physical Location	Average Latency (ms)
Europe	France Central	Paris	28 ms
Europe	Germany West Central	Frankfurt	22 ms
Europe	Switzerland North	Zurich	18 ms

```
var messages = new List<ServiceBusMessage>();
for (var i = 0; i < 10; i++)
{
    var message = new ServiceBusMessage("Deep Dive{i}");
    messages.Add(message);
}

await sender.SendMessagesAsync(messages);
```

cmd - dotnet run

<1> cmd - dotnet r...

Search

```
c:\p\AzureServiceBus.DeepDive\Batching
> dotnet run
Sending 10 messages in a batch.

Sending 6500 messages in a batch.
The received message (delivery-id:0, size:271890 bytes) exceeds the limit (262144 bytes) currently allowed on the link.

Sending 101 messages in a batch with in transaction 'e943ed70-447a-421c-a777-876b2249bf9a:1'.
```

```
var messagesToSend = new Queue<ServiceBusMessage>();
for (var i = 0; i < 4500; i++)
{
    var message = new ServiceBusMessage($"Deep Dive{i}. Deep Dive{i}. Deep Dive
{i}.");
    messagesToSend.Enqueue(message);
}
```

```
var messageCount = messagesToSend.Count;
int batchCount = 1;
while (messagesToSend.Count > 0)
{
    using ServiceBusMessageBatch messageBatch = await sender.CreateMessageBatchAsync();

    if (messageBatch.TryAddMessage(messagesToSend.Peek()))
    {
        messagesToSend.Dequeue();
    }
    else
    {
        throw new Exception($"Message {messageCount - messagesToSend.Count} is too large a
nd cannot be sent.");
    }

    while (messagesToSend.Count > 0 && messageBatch.TryAddMessage(messagesToSend.Peek()))
    {
        messagesToSend.Dequeue();
    }


    await sender.SendMessagesAsync(messageBatch);
}
```

Upgrade to premium tier

Use Claim Check Pattern

Send Via


Incoming
Transfer
Outgoing1
Outgoing2

```
await using var transactionalClient =
            new ServiceBusClient(connectionString,
            new ServiceBusClientOptions
        {
            EnableCrossEntityTransactions = true,
        });

        receiver = transactionalClient.CreateProcessor(inputQueu
        e);
        sender = transactionalClient.CreateSender(destinationQue
        ue);
```

```
receiver.ProcessMessageAsync += async processMessageEventArgs =>
{
    var message = processMessageEventArgs.Message;
    using (var scope = new TransactionScope(TransactionScopeAsyncFlowOption.Enabled))
    {
        await sender.SendMessageAsync(new ServiceBusMessage("Will not leak"));

        if (!message.ApplicationProperties.ContainsKey("Win"))
                    throw new InvalidOperationException();

        await sender.SendMessageAsync(new ServiceBusMessage("Will not leak"));

        scope.Complete();
    }

};
```

cmd - dotnet run

<1> cmd - dotnet r...

Search

```
c:\p\AzureServiceBus.DeepDive\SendVia
> dotnet run
#'0' messages in 'destination'
Received message with 'bdc126e8424549acb9d1c23a5af31799' and content 'Kick off'
```

cmd - dotnet run

<1> cmd - dotnet r...

Search

```
c:\p\AzureServiceBus.DeepDive\SendVia
> dotnet run
#'0' messages in 'destination'
Received message with 'bdc126e8424549acb9d1c23a5af31799' and content 'Kick off'
#'1' messages in 'destination'
Received message with 'bdc126e8424549acb9d1c23a5af31799' and content 'Kick off'
#'2' messages in 'destination'

Received message with '3ef5f1087b43424ba0fcf1987df53ee5' and content 'Fail'
#'0' messages in 'destination'
Received message with '3ef5f1087b43424ba0fcf1987df53ee5' and content 'Fail'
#'0' messages in 'destination'
Received message with 'b2f109763d3442788e6cf41f67fce078' and content 'Win'
#'2' messages in 'destination'
```

TransferDLQ

```
receiver.ProcessMessageAsync += async processMessageEventArgs =>
{
    var message = processMessageEventArgs.Message;
    using (var scope = new TransactionScope(TransactionScopeAsyncFlowOption.Enabled))
    {
        await sender.SendMessageAsync(new ServiceBusMessage("Will not leak"));

        var client = new ServiceBusAdministrationClient(connectionString);

        QueueProperties queueProperties = await client.GetQueueAsync(destinationQueue);
        queueProperties.Status = EntityStatus.SendDisabled;

        await client.UpdateQueueAsync(queueProperties);

                scope.Complete();
            }

        };
```

```
var client = new  ServiceBusAdministrationClient(connectionString);

QueueRuntimeProperties info = await client
       .GetQueueRuntimeInfoAsync(destination);


long activeMessageCount =  info.ActiveMessageCount;
long deadLetterMessageCount = info.DeadLetterMessageCount;
long transferDeadLetterMessageCount = info.TransferDeadLetterMessageC
ount;

string destinationDeadLetterPath = EntityNameHelper
              .FormatDeadLetterPath(destination);
string destinationTransferDeadLetterPath = EntityNameHelper
              .FormatTransferDeadLetterPath(destination);
```

cmd - dotnet run

<1> cmd - dotnet r...

Search

```
c:\p\AzureServiceBus.DeepDive\TransferDLQ
> dotnet run
Received message with '4509ba5ef4f94bcdbfa1972a56eed23d' and content 'Kick off'
#'1' messages in 'queue'
#'0' messages in 'queue/$DeadLetterQueue'
#'1' messages in 'queue/$Transfer/$DeadLetterQueue'
```

Dedup

```
var queueDescription = new CreateQueueOptions(destination)
{
    RequiresDuplicateDetection = true,
    DuplicateDetectionHistoryTimeWindow = TimeSpan.FromSeconds(20)
};
await client.CreateQueueAsync(queueDescription);
```

```
var content = Encoding.UTF8.GetBytes("Message1Message1");
var messageId = new Guid(content).ToString();

var messages = new List<ServiceBusMessage>
{
    new ServiceBusMessage(content) { MessageId = messageId },
    new ServiceBusMessage(content) { MessageId = messageId },
    new ServiceBusMessage(content) { MessageId = messageId }
};

await sender.SendMessagesAsync(messages);
```

cmd - dotnet run

<1> cmd - dotnet r...

Search

```
c:\p\AzureServiceBus.DeepDive\Dedup
> dotnet run
Messages sent
Received message with '7373654d-6761-3165-4d65-737361676531' and content 'Message1Message1'
```

SUMMARY

ENTERPRISE MESSAGING FEATURES

MESSAGE TRANSACTIONAL PROCESSING

RELIABILITY

GUARANTEED THROUGHPUT AND LATENCY

TOTAL: PRICELESS

Azure Service Bus Transport

Tags

Source | NServiceBus.Transport.AzureServiceBus (1.x)
Target NServiceBus Version: 7.x

The Azure Servi[illegible]ce Bus transport leverages the .NET Standard Microsoft.Azure.ServiceBus client SDK.

Azure Service B[illegible]us is a messaging service hosted on the Azure platform that allows for exchanging messages between various applicati[illegible]ons in a loosely coupled fashion. The service offers guaranteed message delivery and supports a range of standard protocc[illegible]ols (e.g. REST, AMQP, WS*) and APIs such as [illegible]tive pub/sub, delayed delivery [illegible]

Configurin[illegible]g endpoint

To use Azure Se[illegible]rvice Bus as the underlying transport:

```
var transport = endpointConfiguration.UseTransport<AzureServiceBusTransport>();
transport.ConnectionString("Endpoint=sb://[NAMESPACE].servicebus.windows.net/;SharedAccessKeyName=[KEYNAME];SharedAccessKey=[KEY]");
```

Copy/Edit

The Azure Service Bus transport requires a connection string to connect to a namespace.

Samples

- Azure Service Bus Native Integration
  Consuming messages published by non-NServiceBus endpoints.
- Azure Service Bus Send/Reply Sample
  Demonstrates the send/reply pattern with Azure Service Bus.

Particular
Software

Thanks

Slides, Links...

github.com/danielmarbach/AzureServiceBus.DeepDive

Particular Software

Q&A

Particular Software

Software Engineer
Microsoft MVP

@danielmarbach
particular.net/blog
planetgeek.ch