

Ultimate Guide to Logging

Your open-source resource for understanding, analyzing, and troubleshooting system logs



Centralizing Windows Logs

You can use the tools in this article to centralize your Windows event logs from multiple servers and desktops. By properly administering your logs, you can track the health of your systems, keep your log files secure, and filter contents to find specific information.

Why Centralize Logs?

Centralizing your logs saves time and increases the reliability of your log data. When Windows log files are stored locally on each server, you have to individually log in to each one to go through them and look for any errors or warnings. If the server is unresponsive, you might be out of luck. If you aren't sure which servers are affected, you have to hunt through each one, which can take a long time on large networks. The log files are also safer in a centralized location because even when your instances are terminated or your files are deleted (intentionally or unintentionally), the centralized backup copies of your logs are unaffected.

Windows Event Subscription

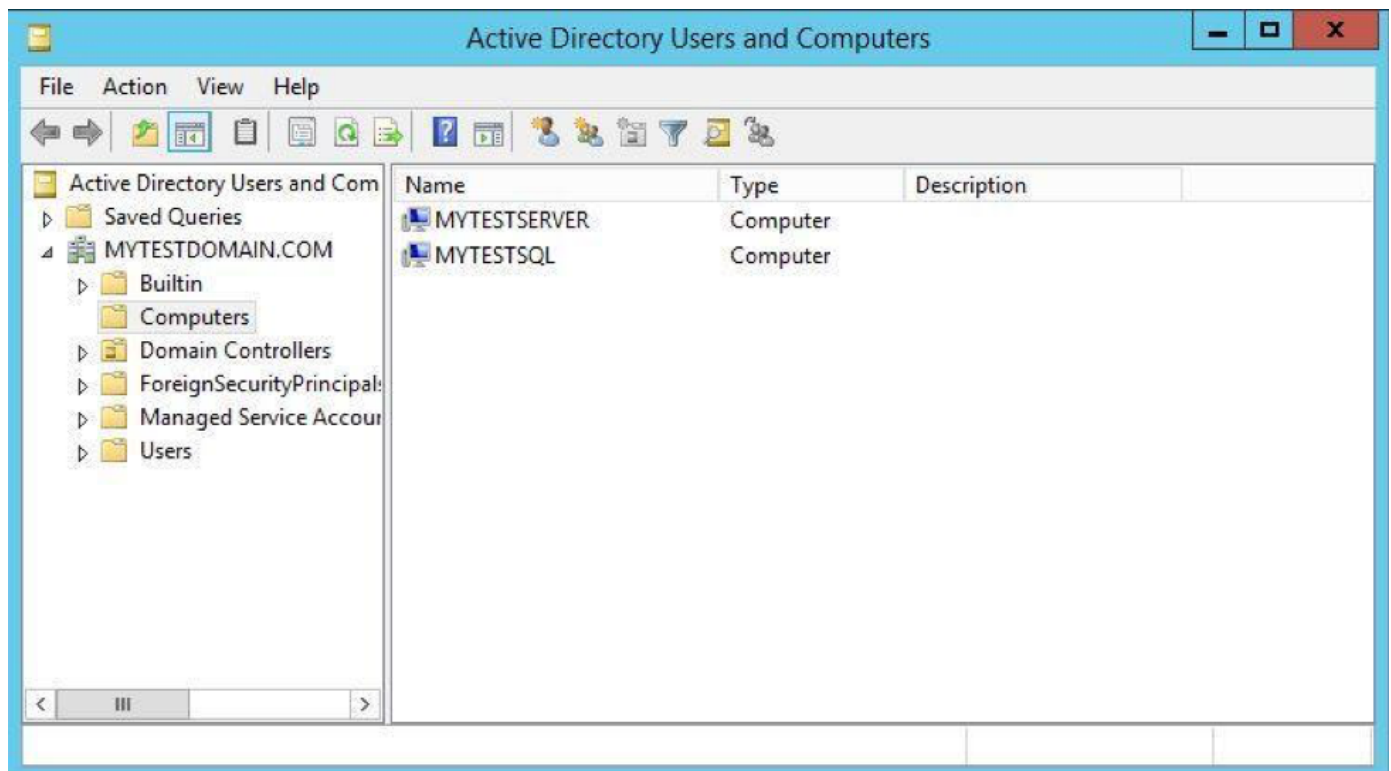
It is possible for a Windows server to forward its events to a **collector** server. In this scenario, the **collector** server becomes a central repository for Windows logs from other servers (called event **sources**) in the network. The stream of events from a source to a collector is called a **subscription**.

This procedure demonstrates how to set it up. These steps work on Windows Server 2008 R2, Windows Server 2012, and Windows Server 2019.

Example System

We are using two Active Directory Domain-joined Windows Server 2012 systems. The domain name is mytestdomain.com (<https://mytestdomain.com/>), and both machines are registered with the domain.

Source server **MYTESTSQL** hosts a **SQL Server 2014** instance. **Collector** server **MYTESTSERVER** works as an **event log subscriber** to centralize all SQL Server-related logs from **MYTESTSQL**.



Setup

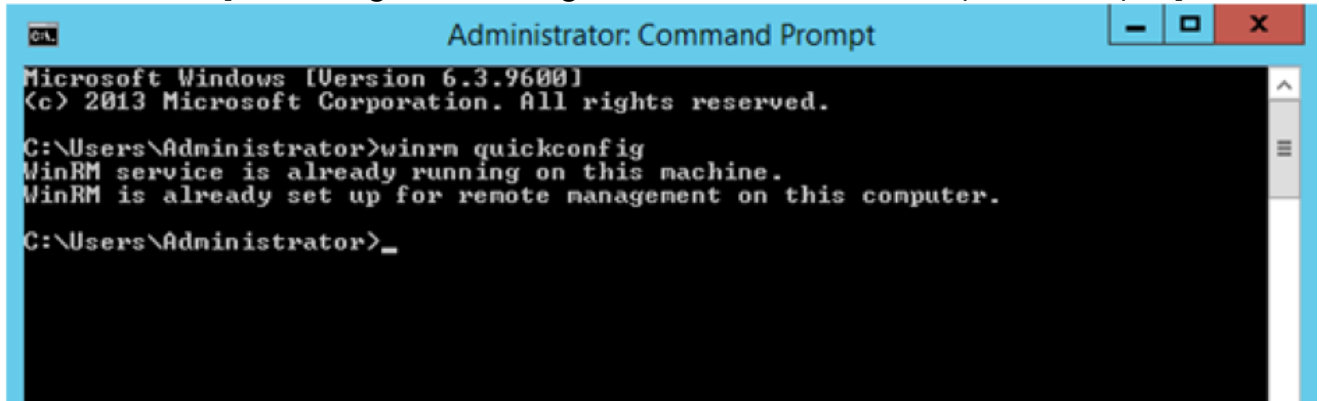
Enable the Windows Remote Management Service

Windows Remote Management (WinRM) is a protocol for exchanging information across systems in your infrastructure. You must enable it on each of your **source** computers to exchange log files.

1. Remotely log into the **source** computer (MYTESTSQL) as a local or domain administrator.
2. Enable **Windows Remote Management Service** from a Command Prompt:

```
winrm quickconfig
```

If it is already running, a message similar to this example is displayed.

A screenshot of a Windows Command Prompt window titled "Administrator: Command Prompt". The window shows the output of the 'winrm quickconfig' command. The text displayed is: "Microsoft Windows [Version 6.3.9600] (c) 2013 Microsoft Corporation. All rights reserved. C:\Users\Administrator>winrm quickconfig WinRM service is already running on this machine. WinRM is already set up for remote management on this computer. C:\Users\Administrator>_".

```
Administrator: Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>winrm quickconfig
WinRM service is already running on this machine.
WinRM is already set up for remote management on this computer.

C:\Users\Administrator>_
```

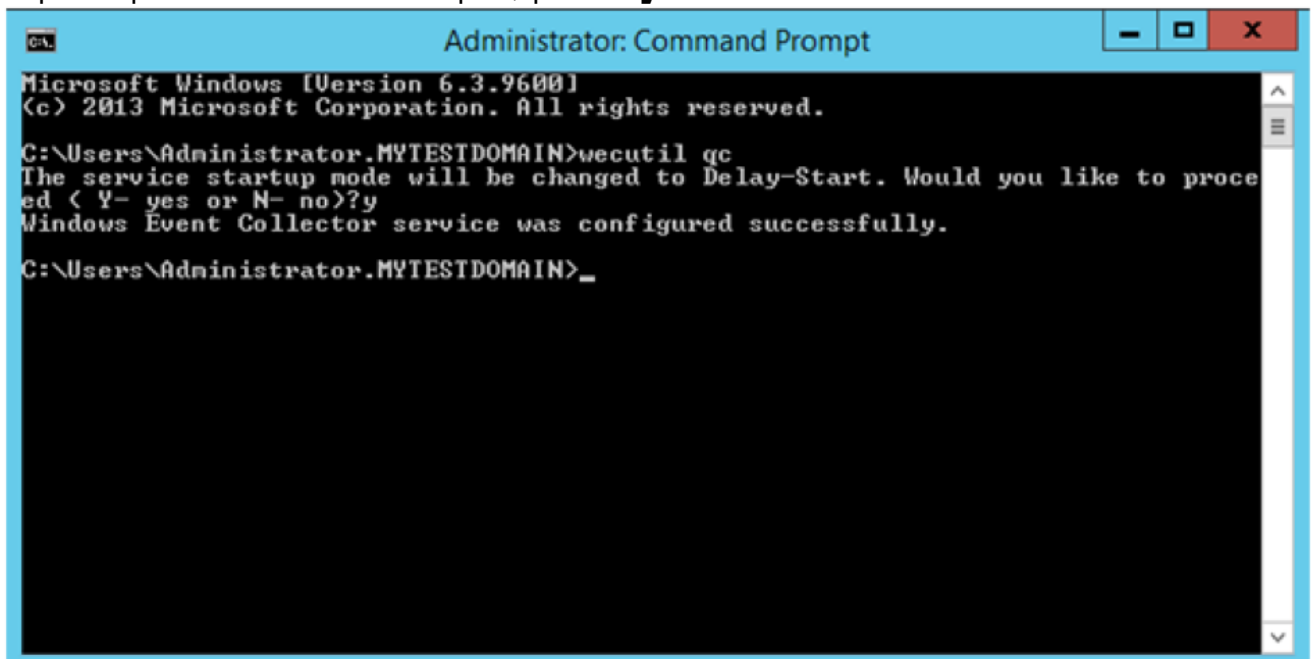
Configure the Windows Event Collector Service

You must enable the **Windows Event Collector Service** on your **collector** server to allow it to receive logs from your **sources**.

1. Remotely log into the **collector** computer (MYTESTSERVER) as a local or domain administrator.
2. Configure the **Windows Event Collector Service** from a Command Prompt:

```
wecutil qc
```

If prompted like the example, press **y**

A screenshot of a Windows Command Prompt window titled "Administrator: Command Prompt". The window shows the output of the 'wecutil qc' command. The text displayed is: "Microsoft Windows [Version 6.3.9600] (c) 2013 Microsoft Corporation. All rights reserved. C:\Users\Administrator.MYTESTDOMAIN>wecutil qc The service startup mode will be changed to Delay-Start. Would you like to proceed (Y- yes or N- no)?y Windows Event Collector service was configured successfully. C:\Users\Administrator.MYTESTDOMAIN>_".

```
Administrator: Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

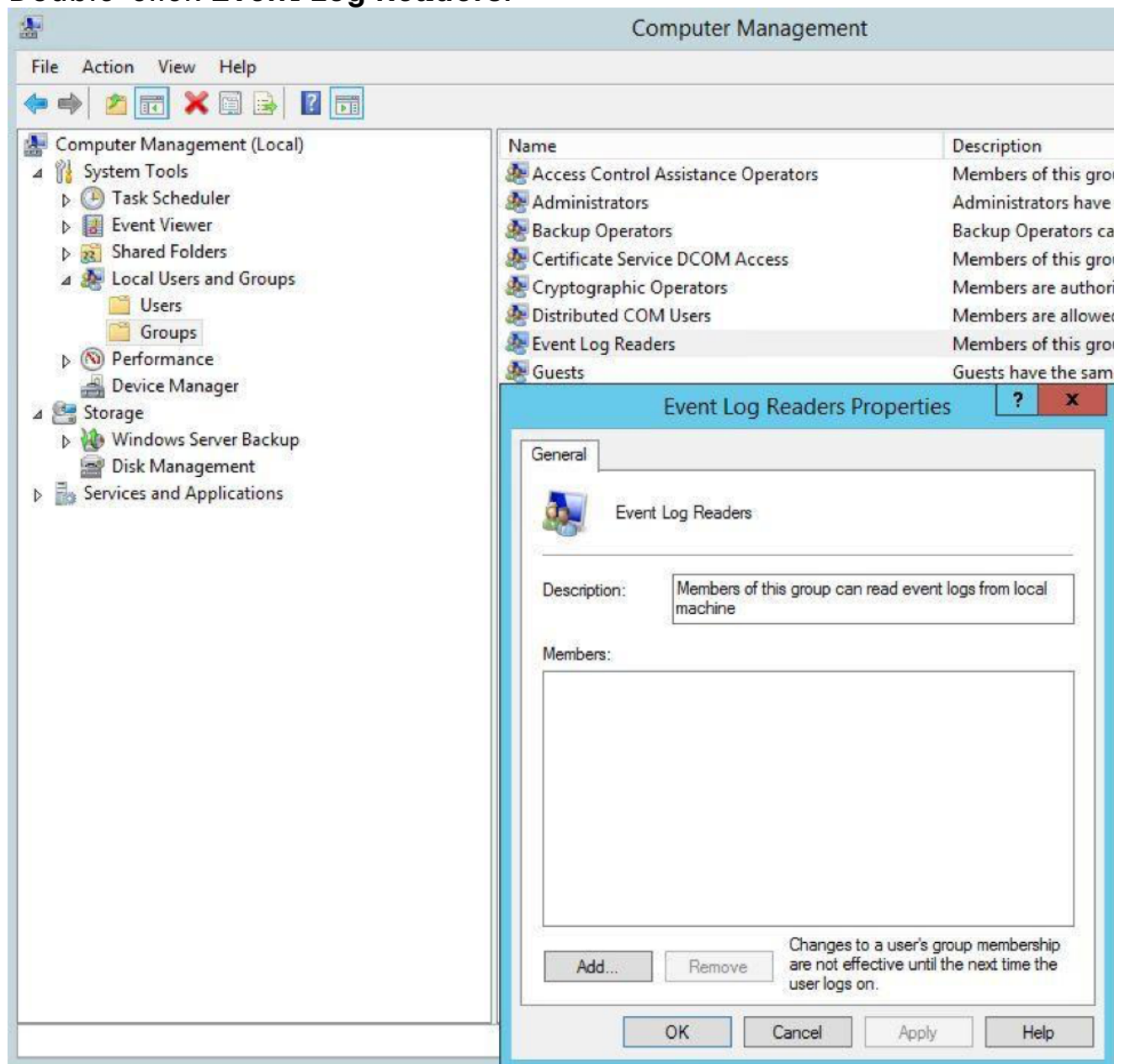
C:\Users\Administrator.MYTESTDOMAIN>wecutil qc
The service startup mode will be changed to Delay-Start. Would you like to proceed (Y- yes or N- no)?y
Windows Event Collector service was configured successfully.

C:\Users\Administrator.MYTESTDOMAIN>_
```

Configure the Event Log Readers Group

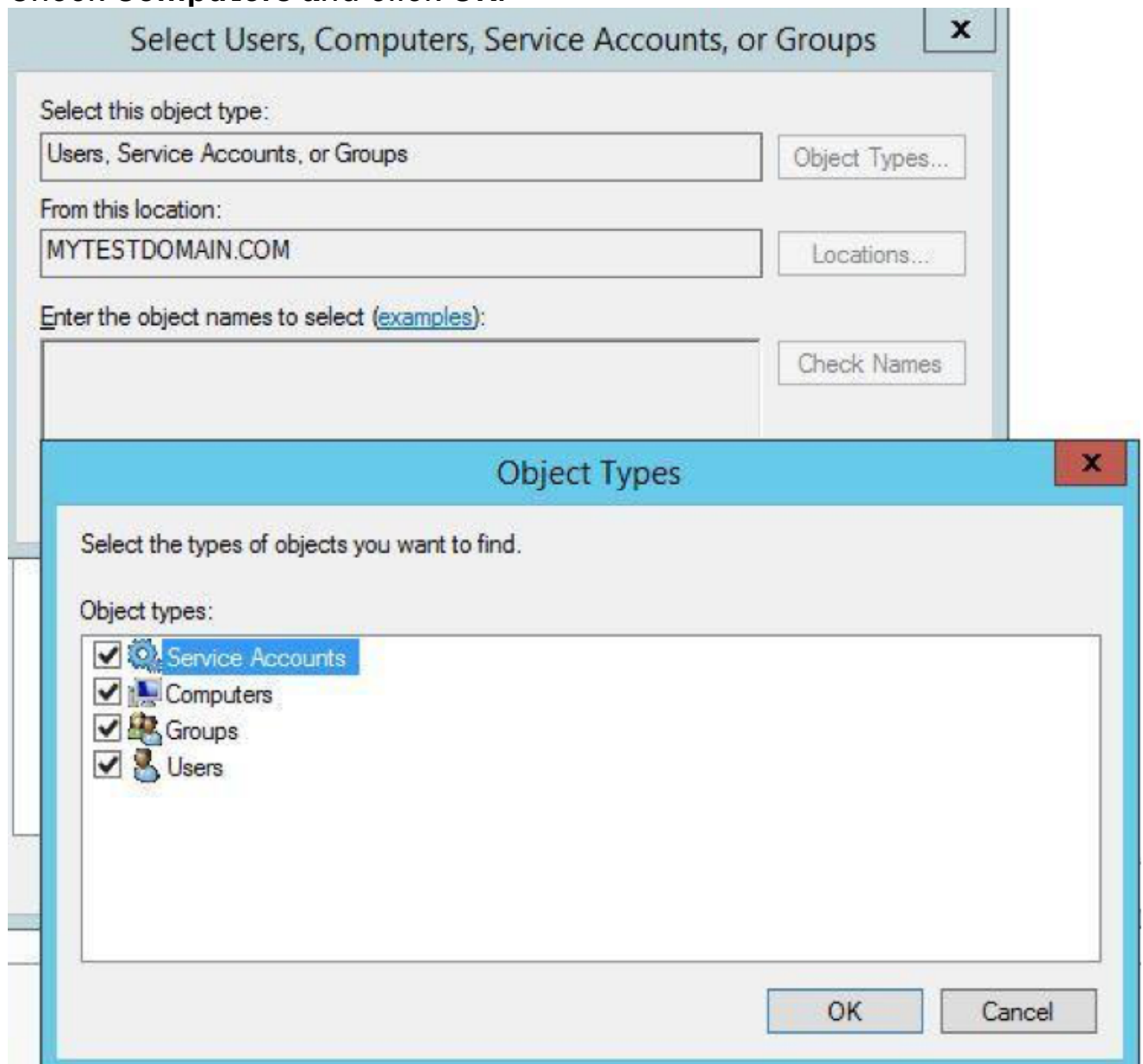
By default, certain logs are restricted to administrators. This may cause problems when receiving logs from other systems. To avoid this, you can grant access to the **collector** computer by adding it to the **Event Log Readers** group.

1. Go back to the **source** computer (MYTESTSQL).
2. Open **Server Manager**.
3. Open **Computer Management**.
4. Expand **Local Users and Groups** node from the Navigation pane and select **Groups**.
5. Double-click **Event Log Readers**.



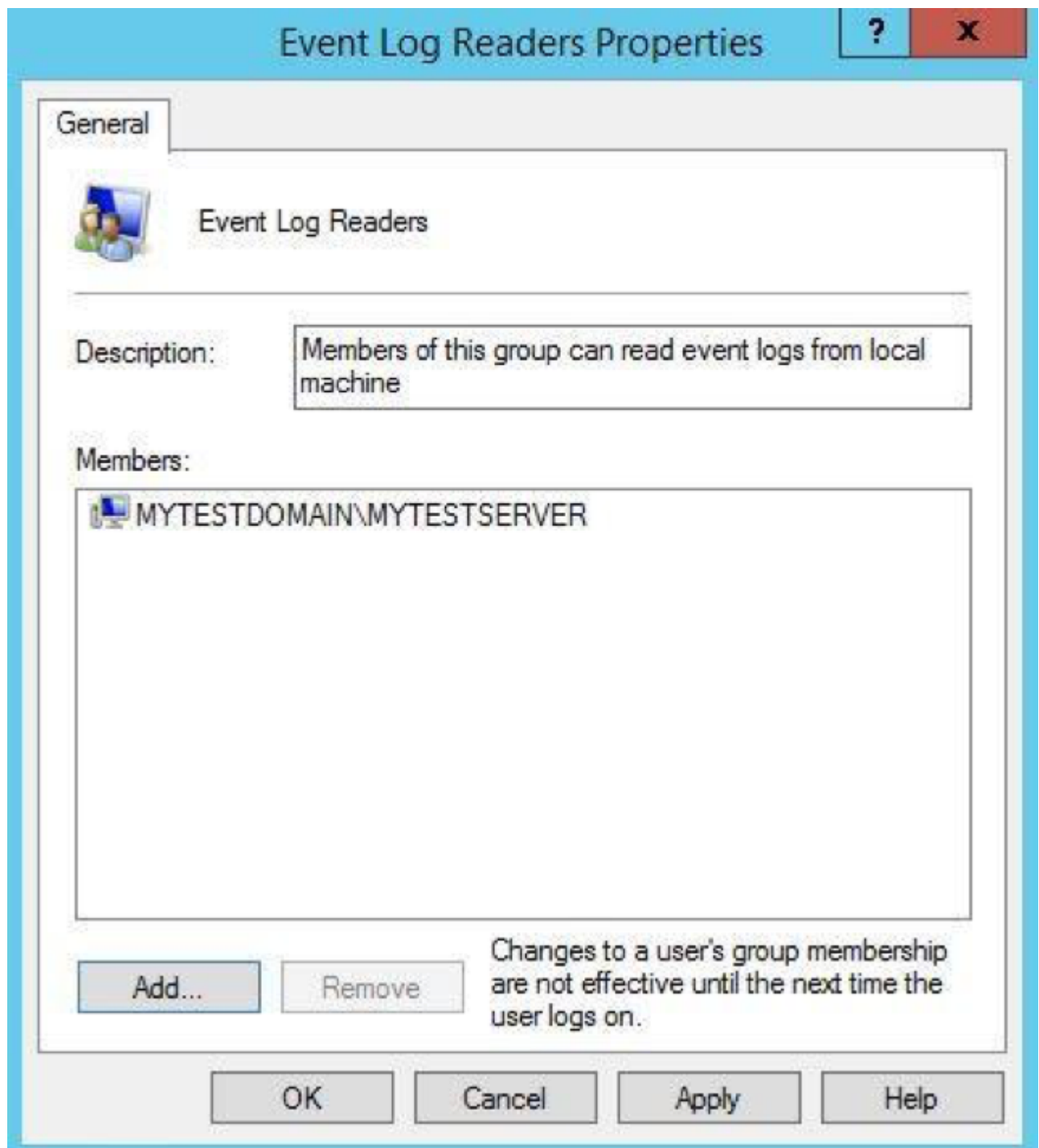
6. Click **Add** to open the **Select Users, Computers, Service Accounts, or Groups** dialog.
7. Click **Object Types**.

8. Check **Computers** and click **OK**.



9. Enter **MYTESTSERVER** as the object name and click **Check Names**. If the computer account is found, it is confirmed with an underline.

10. Click **OK** twice to close the dialog boxes.



Configure Windows Firewall

If the source computer is running Windows Firewall, ensure it allows **Remote Event Log Management** and **Remote Event Monitor** traffic.

Allow apps to communicate through Windows Firewall

To add, change, or remove allowed apps and ports, click Change settings.

What are the risks of allowing an app to communicate?

[Change settings](#)

Allowed apps and features:

Name	Domain	Private	Public	
<input type="checkbox"/> File and Printer Sharing over SMBDirect	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input checked="" type="checkbox"/> File Server Remote Management	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/> iSCSI Service	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> Key Management Service	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> Netlogon Service	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> Network Discovery	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> Performance Logs and Alerts	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input checked="" type="checkbox"/> Remote Desktop	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input checked="" type="checkbox"/> Remote Event Log Management	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input checked="" type="checkbox"/> Remote Event Monitor	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> Remote Scheduled Tasks Management	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/> Remote Service Management	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

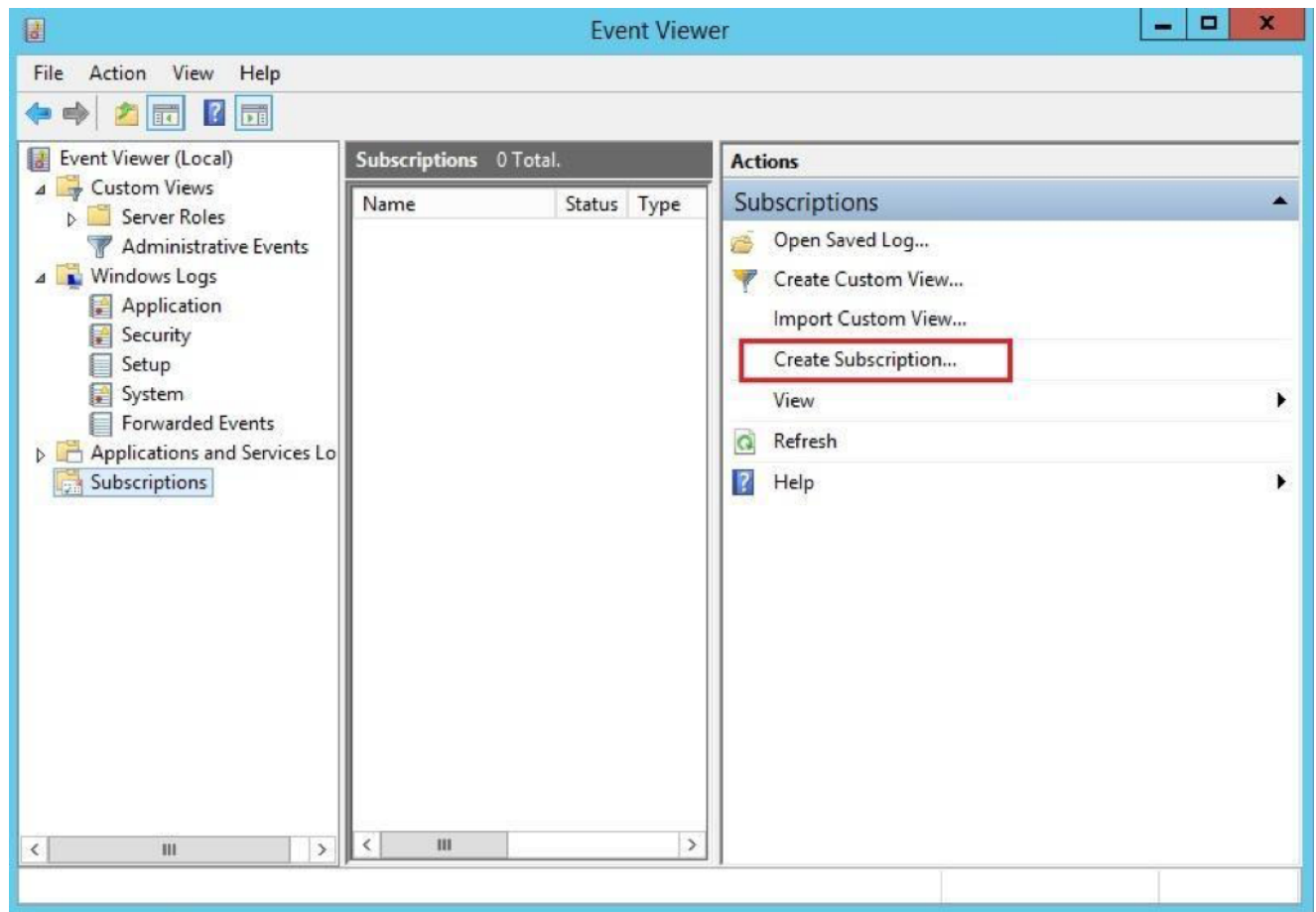
[Details...](#) [Remove](#)

[Allow another app...](#)

Create a Subscription

Subscriptions define the relationship between a **collector** and a **source**. You can configure a **collector** to receive events from any number of **sources** (a source-initiated subscription), or specify a limited set of **sources** (a collector-initiated subscription). In this example, we create a **collector**-initiated subscription since we know which computer logs we want to receive.

1. Start the Event Viewer application on the collector server **MYTESTSERVER**.
2. Select **Subscriptions** from the Navigation pane
3. Click **Create Subscription** in the Actions pane.



4. On the **Subscription Properties**, enter the following as shown in the example:

Subscription name: **MYTESTSQL_EVENTS**

Description: **Events from remote source server MYTESTSQL**

Destination log: **Forwarded Events**

Select **Collector initiated** and click **Select Computers** to open the **Computers** dialog.

Subscription Properties - MYTESTSQL_EVENTS

Subscription name: MYTESTSQL_EVENTS

Description: Events from remote source server MYTESTSQL

Destination log: Forwarded Events

Subscription type and source computers

☒ Collector initiated Select Computers...

This computer contacts the selected source computers and provides the subscription.

☐ Source computer initiated Select Computer Groups...

Source computers in the selected groups must be configured through policy or local configuration to contact this computer and receive the subscription.

Events to collect: Select Events...

User account (the selected account must have read access to the source logs):

Machine Account

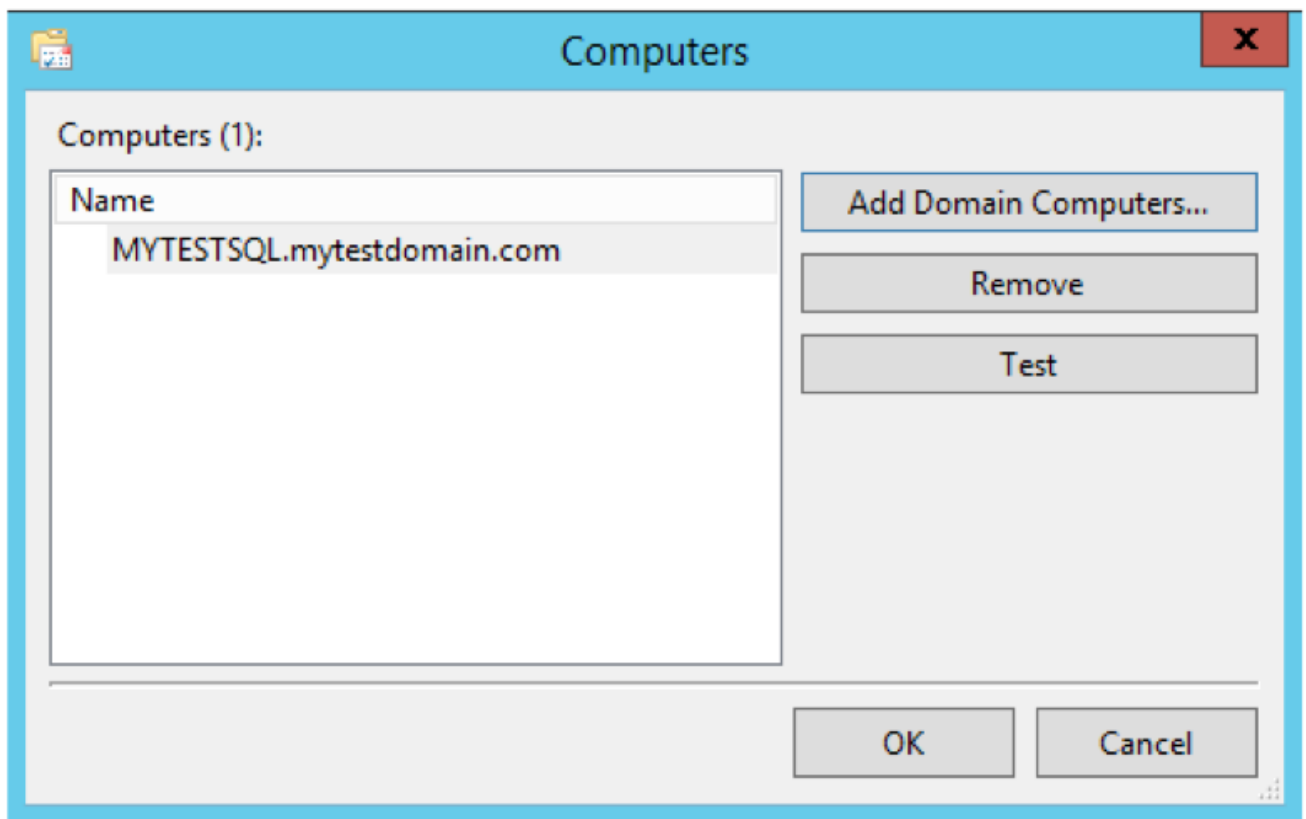
Change user account or configure advanced settings: Advanced...

OK Cancel

5. Click **Add Domain Computers**.

6. Enter **MYTESTSQL** as the object name and click **Check Names**. If the computer is found, it is confirmed with an underline.

7. Click **OK**.



8. Click **OK** to return to the **Subscription Properties**.
9. Click **Select Events** to open the **Query Filter** and enter the following to set the remote server to forward all application events from the last 24 hours:
 - Logged: **Last 24 hours**
 - Check all **Event levels**
 - Select **By log**
 - Event logs: Select **Application** from the drop-down list

Query Filter

Filter XML

Logged: Last 24 hours

Event level: ☒ Critical ☒ Warning ☒ Verbose
☒ Error ☒ Information

☒ By log Event logs: Application

☐ By source Event sources:

Includes/Excludes Event IDs: Enter ID numbers and/or ID ranges separated by commas. To exclude criteria, type a minus sign first. For example 1,3,5-99,-76

<All Event IDs>

Task category:

Keywords:

User: <All Users>

Computer(s): <All Computers>

Clear

OK Cancel

10. Click **OK** to return to the **Subscription Properties**.
11. Click **Advanced** to open the **Advanced Subscription Settings** and enter the following:
 - Select **Machine Account**
 - Select **Minimize Latency**
 - Protocol: **HTTP**
 - Port: **5985**

Advanced Subscription Settings X

User Account:
The selected account must have read access to the source logs

☒ Machine Account
☐ Specific User

MYTESTDOMAIN\Administrator User and Password...

Event Delivery Optimization:

☐ Normal
☐ Minimize Bandwidth
☒ Minimize Latency
☐ Custom

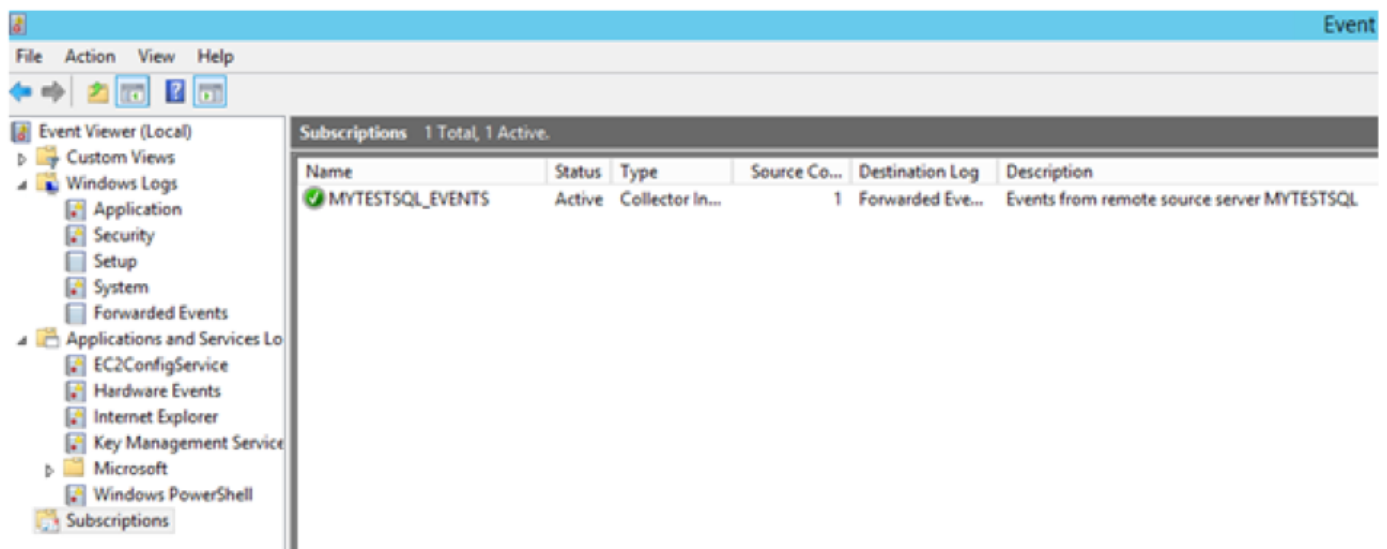
Protocol: HTTP Port: 5985

OK Cancel

12. Click **OK** to return to the **Subscription Properties**.

13. Click OK to close.

The Subscription node in the collector computer event viewer now shows the new subscription.



Event Viewer (Local)

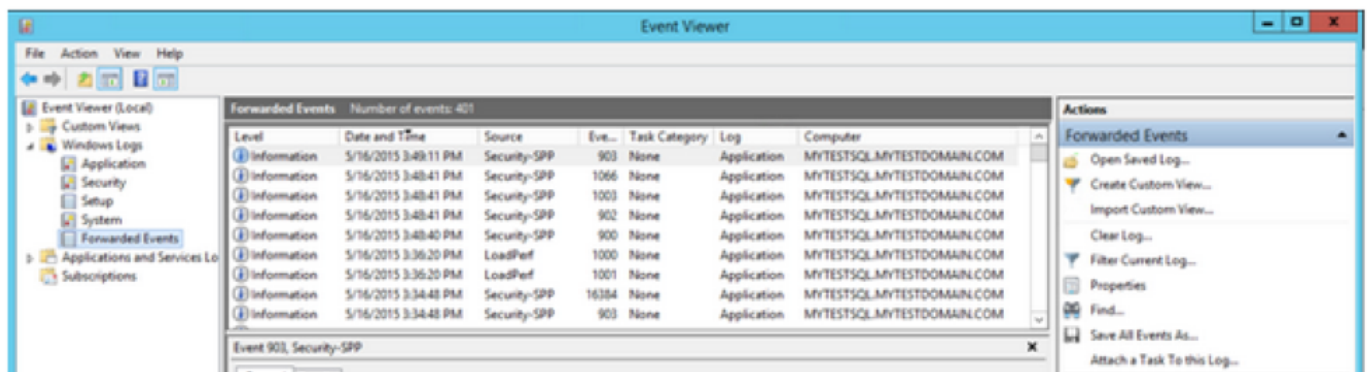
File Action View Help

Subscriptions 1 Total, 1 Active.

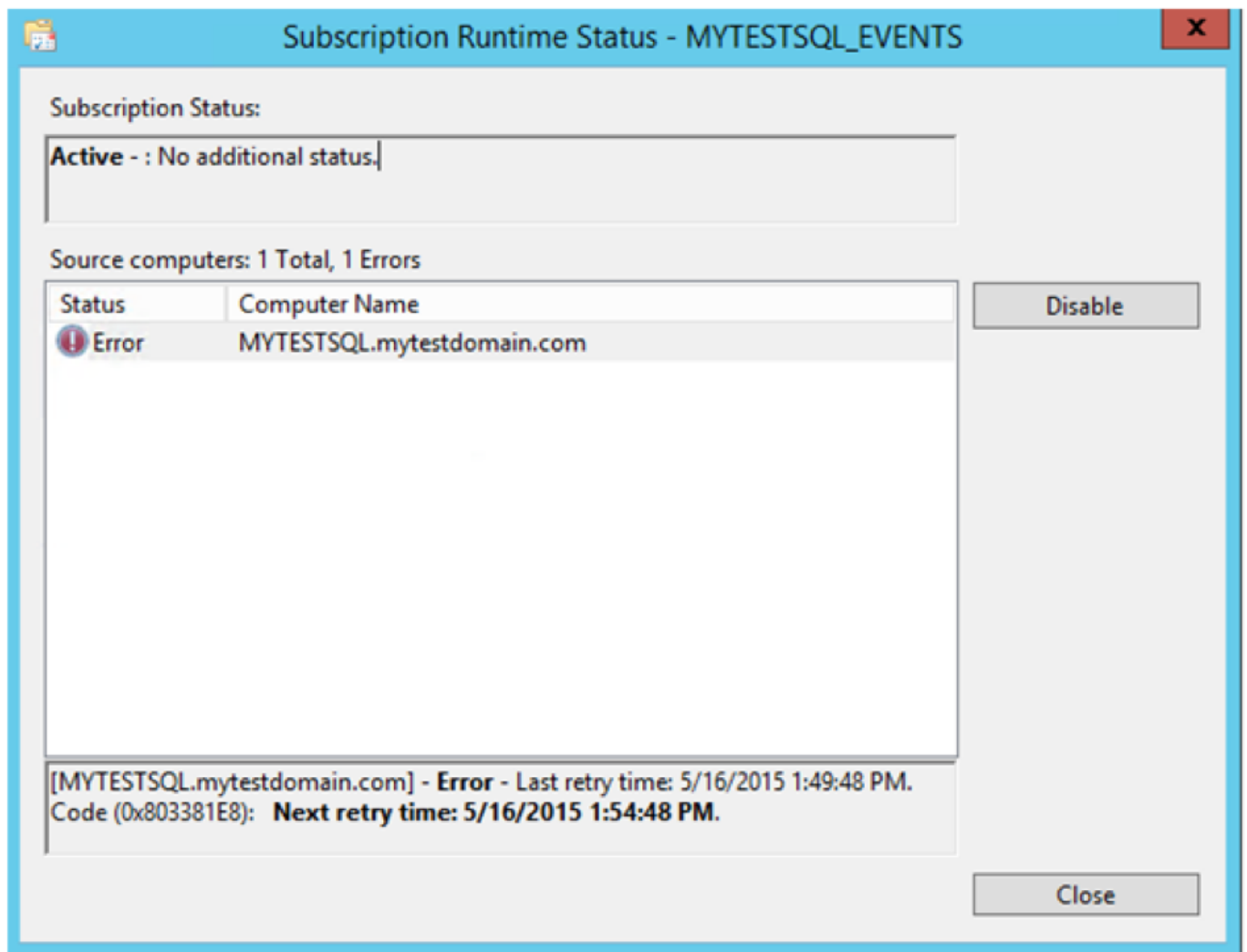
Name	Status	Type	Source Co...	Destination Log	Description
MYTESTSQL_EVENTS	Active	Collector In...	1	Forwarded Eve...	Events from remote source server MYTESTSQL

Verify Events on Collector Computer

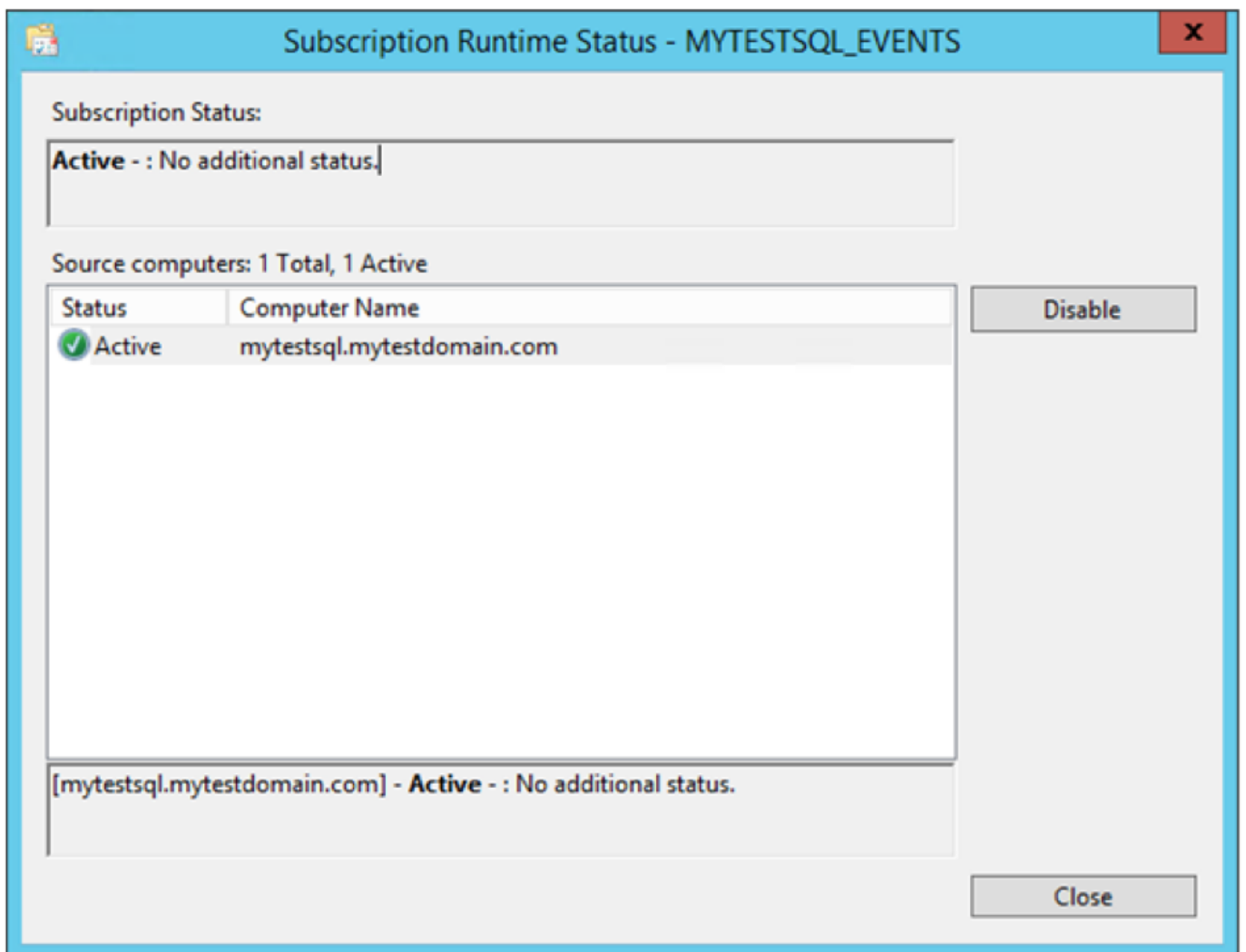
Select **Forwarded Events** from the Navigation pane on the collector computer.



The **Computer** column in the Details pane indicates the events are from the remote computer **MYTESTSQL.MYTESTDOMAIN.COM**. You can enable or disable the collector subscription by right-clicking on the subscription and choosing **Disable**. The status of the subscription is then shown as disabled in the main window. An active collector subscription does not mean it is succeeding. To see if the collector can connect to the source, right-click on the subscription and select **Runtime Status**. In this example, the collector can't connect to the source. By default, it retries every five minutes.



If all is OK, **Subscription Runtime Status** shows a green tick with an active status.



Create a Custom View (Optional)

Once the events are forwarded, you can create custom views to see the consolidated events. For example, you might create a custom view for error events. This example creates a custom view for SQL Server–related messages. A collector computer may host thousands of records from dozens of servers. Using a custom view enables you to create order from an overload of information. For detailed steps, see the section **Creating a Custom View** in

Windows Logging Basics (<https://www.loggly.com/ultimate-guide/windows-logging-basics/>).

•

Windows Logging Services

There are several Windows services you can use to centralize all your logging data to an external logging service. These services send logs over **syslog** to a cross-platform log server or cloud-based logging service like SolarWinds® Loggly®.

We recommend **NXLog** (<https://nxlog.org/>), a popular, freely downloadable service that runs in the background. Alternately, there is **syslog-ng** (<https://www.balabit.com/network-security/syslog-ng>) and **Snare** (<https://www.snaresolutions.com/>), which are services that collect your log files. All these services provide additional professional support for a fee.

Install NXLog

This example installs and configures **NXlog** to package your log files.

Download and install the current version of NXlog
(<https://nxlog.org/system/files/products/files/1/nxlog-ce-2.8.1248.msi>). The download includes an intuitive installer. Once the installation is complete, open the configuration file. By default, the **NXLog** configuration file is located at **C:/Program Files (x86)/nxlog/conf/nxlog.conf**

You can create different types of configuration modules.

- Inputs for the source of your logs

- Outputs for where to send the logs

- Routes to map your inputs to your outputs

Whenever you make changes to the NXlog configuration file, you must restart the **NXlog service**.

Configure NXLog

This example modifies the **NXLog** configuration file to centralize your Windows event logs. Adding the code snippet below to the end of your nxlog.conf file enables the module and gives it the name “eventlog”. The **im_msvistalog** input module sends new entries to the Windows event log, including system, hardware, application, and security-related events.

```
# Windows Event Log

<Input eventlog>

# Uncomment im_msvistalog for Windows Vista/2008 and later

Module im_msvistalog

# Uncomment im_mseventlog for Windows XP/2000/2003

# Module im_mseventlog

# If you prefer to send events as JSON data

Exec $Message = to_json();

</Input>
```

File Logs

NXLog can be used to read logs files stored on a drive. In this example, the file name is **FILE1**. SavePos TRUE means that NXLog will track its current location in the log file on exit. Exec \$Message = \$raw_event means NXLog will ingest the raw log message without applying any additional formatting. The file name can also include directories or wild cards.

```
<Input FILE1>

Module im_file

File "FILE1"

SavePos TRUE

Exec $Message = $raw_event;

</Input>
```

IIS Logs

As we covered in the [Windows Logging Basics \(https://www.loggly.com/ultimate-guide/windows-logging-basics/\)](https://www.loggly.com/ultimate-guide/windows-logging-basics/)

section, IIS logs contain access logs stored in W3C format. We recommend you convert them to JSON format for easy processing by a log management tool. **NXLog** can do this conversion using the W3C extension. Make sure you use the proper format in the configuration file, so the parsing happens correctly, and you are including log files from all your sites.

```

<Extension w3c>

Module xm_csv

Fields $date, $time, $s-ip, $cs-method, $cs-uri-stem, $cs-uri-query, $s-port, $cs-username, $c-
FieldTypes string, string, string, string, string, string, integer, string, string, string, str
Delimiter ' '
QuoteChar '"'
EscapeControl FALSE
UndefValue -

</Extension>

# Convert the IIS logs to JSON and use the original event time

<Input IIS_Site1>

Module      im_file

File        "C:/inetpub/logs/LogFiles/W3SVC1/u_ex*"

SavePos     TRUE

Exec if $raw_event =~ /^#/ drop();

else

{

w3c->parse_csv();

$SourceName = "IIS";

$Message = to_json();

}

</Input>

```

SQL Server Error Logs

SQL Server is Microsoft's enterprise-class flagship database platform. It comes in a suite of database and data warehouse tools. SQL Server typically has its own logs saved in the application's installation directory in the Windows file system. The default location for SQL Server 2012 is

C:/Program Files/Microsoft SQL

Server/MSSQL11.MSSQLSERVER/MSSQL/Log. The log entries are also sent to the Windows application event log.

SQL Server operations like backup and restore, query timeouts, or slow I/Os are therefore easy to find from Windows application event log, while security-related messages like failed login attempts are captured in Windows security event log.

Forwarding Logs to a Server

NXLog can forward logs from any of the inputs described above to an external destination such as a log server or cloud-based log management service. To do this, NXLog uses concepts called **Outputs** and **Routes**. Outputs are modules that provide functionality for sending logs to a destination, such as a file or remote server. Routes are the paths that a log message takes from an input (such as the `im_msvistalog` module) to an output (such as a log management service).

To forward logs, add an output module in your **`nxlog.conf`** configuration file. Then add a **Route** module to send logs from your chosen inputs to your chosen outputs. In this example, we are sending logs as **syslog** over **TCP** to the host **HOSTNAME** over the default **syslog port 514**. We create a route that takes logs from the **eventlog** input and sends it to the new output (named **out**):

```
<Output out>

Module om_tcp

Host HOSTNAME

Port 514

</Output>

<Route 1>

Path eventlog => out

</Route>
```

Several log management solutions offer specific setup instructions for Windows logging. **Loggly** is an example of one provider and has more detailed information about setting up **NXLog** to gather your log files in their guide,

[Logging from Windows \(https://www.loggly.com/docs/logging-from-windows/\)](https://www.loggly.com/docs/logging-from-windows/).

Encrypting Logs with TLS

By default, logs sent over the Internet are transmitted in clear text. This means snoopers can intercept and view your log data. It is best practice to encrypt your log data when it's in transit, especially if it contains sensitive information like personal identification details, government-regulated data, or financial information. The most common protocol for encrypting syslog communication is **TLS**, or

Transport Layer Security.

(https://en.wikipedia.org/wiki/Transport_Layer_Security).

TLS encrypts your logs, preventing anyone from snooping on sensitive data in your logs. Best practice is not to log information like passwords, but some applications do it anyway. TLS encryption helps keep this data safer. Encryption prevents malicious parties located between your log sources and destinations from reading or modifying your log data.

Here is an example setting up **NXLog** configuration with TLS encryption for Loggly.

1. Download Loggly's digital certificate from the **NXLog TLS configuration page** (<https://www.loggly.com/docs/nxlog-tls-configuration/>).
2. Copy the digital certificate file to your **NXLog** cert directory:
copy loggly_full.crt C:/Program Files*/nxlog/cert
3. Configure your output module with **om_ssl** and the certificate location. The default **syslog port** for encrypted logs is **6514**. AllowUntrusted FALSE prevents a connection to the server if the certificate is untrusted or self-signed:

```
<Output out>
Module om_ssl
Host server.example.com
Port 6514
CAFile %CERTDIR%/example.crt
AllowUntrusted FALSE
</Output>
```

How do you centralize your logs? Add a comment to let us know!