A cookbook

Examples for
PowerShell for WebSphere MQ

A collection of one-liners demonstrating the sorts of operations possible with PowerShell administration for WebSphere MQ.

For more ideas, see http://www.phpbber.com/phpbb/viewforum.php?f=3&mforum=powershellforwe

Table of Contents

get a list of local queue managers

reverse sort the list by name, showing the name, ID and description

```
PS C:\> Get-WMQQueueManager | Select Name, QueueManagerIdentifier, QueueManagerDescription | Sort-Object Name -descending

Name                                QueueManagerIdentifier                       QueueManagerDescription
----                                ----------------------                       -----------------------
Test                                Test_2007-09-03_09.33.44                     to check name collision
TeSt                                TeSt_2007-09-03_09.33.47                     to check name collision
TEST                                TEST_2007-09-03_09.34.02                     Test queue manager - to be deleted
test                                test_2007-09-02_22.14.03
post                                post_2007-09-03_09.34.19
dale                                dale_2007-09-03_09.01.37                     personal qmgr - for client development work
```

---

show local queue managers which have names which end in 'st'

```
PS C:\> Get-WMQQueueManager *st | Select Name, QueueManagerIdentifier

Name                                 QueueManagerIdentifier
----                                 ----------------------
post                                 post_2007-09-03_09.34.19
test                                 test_2007-09-02_22.14.03
Test                                 Test_2007-09-03_09.33.44
```

---

search for an object across multiple queue managers

find out which queue manager has a queue called "FINAL.Q"

```
PS C:\> Get-WMQQueue FINAL.Q | Select Name, @{e={$_.QueueManager.Name};n='Queue Manager'}

Name                                    Queue Manager
----                                    -------------
FINAL.Q                                 post                                ...
```

---

getting a subset of objects across a subset of queue managers

get a list of queues which contain the word "CLUSTER" in their name, from queue managers with names ending in "st"

```
PS C:\> Get-WMQQueue *CLUSTER* *st | Select Name, @{e={$_.QueueManager.Name};n='Qmgr'}

Name                                              Qmgr
----                                              ----
SYSTEM.CLUSTER.COMMAND.QUEUE                      post
SYSTEM.CLUSTER.REPOSITORY.QUEUE                   post
SYSTEM.CLUSTER.TRANSMIT.QUEUE                     post
SYSTEM.CLUSTER.COMMAND.QUEUE                      test
SYSTEM.CLUSTER.REPOSITORY.QUEUE                   test
SYSTEM.CLUSTER.TRANSMIT.QUEUE                     test
SYSTEM.CLUSTER.COMMAND.QUEUE                      Test
SYSTEM.CLUSTER.REPOSITORY.QUEUE                   Test
SYSTEM.CLUSTER.TRANSMIT.QUEUE                     Test
```

---

find queues which are getting full and increase allocated space

Show all local queues on all local queue managers where the current queue depth is less than 10 messages away from it's max depth setting

```
PS C:\> Get-WMQQueue | where {$_.QueueType -eq “Local” -and $_.CurrentDepth -gt ($_.MaximumDepth - 10)}
| Select Name, CurrentDepth, MaximumDepth

Name                                     CurrentDepth              MaximumDepth
----                                     ------------              ------------
MYQ                    ...                          7                        15
```

And if you want to increase the maximum depth of these queues – such as to add space for 10 more messages…

```
PS C:\> Get-WMQQueue | where {$_.QueueType -eq “Local” -and $_.CurrentDepth -gt ($_.MaximumDepth - 10)}
| foreach {Set-WMQQueue $_ -MaximumDepth ($_.MaximumDepth + 10)}
```

---

count the number of queues on your system

Count all queues on your system. Or just count local queues. Or queues matching any criteria you are interested in.

```
PS C:\> Get-WMQQueue -QmgrName SANDBOX | Measure-Object

Count    : 25

PS C:\> Get-WMQQueue -QmgrName SANDBOX | where { $_.CurrentDepth -gt 1 } | Measure-Object

Count    : 12
```

find transmission queues and their usage

get all transmission queues (except the cluster transmit queue) from all queue managers and show their depths and open counts

```
PS C:\> Get-WMQQueue | where {$_.Usage -eq “Transmission” -and $_.Name -ne
"SYSTEM.CLUSTER.TRANSMIT.QUEUE"} | Select Name, CurrentDepth, OpenInputCount, OpenOutputCount,
@{e={$_.QueueManager.Name};n='Queue Manager'}

Name                        CurrentDepth      OpenInputCount     OpenOutputCount Queue Manager
----                        ------------      --------------     --------------- -------------
TRANS1                                 0                   0                   0 dale            ...
```

get a list of SSL-enabled channels

get all channels from all local queue managers which have an SSL Cipher Spec applied, and show their name, sslciph, conname and the name of the queue manager they are on - sorted by channel name

```
PS C:\> Get-WMQChannel | where { $_.SSLCipherSpec } | Select Name, @{e={$_.QueueManager.Name};n='Queue Manager'},
SSLCipherSpec, ConnectionName | Sort Name

Name             Queue Manager        SSLCipherSpec                    ConnectionName
----             -------------        -------------                    --------------
SECURE           post                 NULL_MD5                         dlane.hursley.ibm.com(9090)
SECURE.R         test                 TRIPLE_DES_SHA_US                dlane.hursley.ibm.com(9091)
SECURE.X         dale                 TLS_RSA_WITH_AES_256_CBC_SHA     dlane.hursley.ibm.com(9094)
```

get details about all sender channels from a subset of queue managers

get all non-system (i.e. channels with names that don't start with SYSTEM) sender channels from local queue managers with names ending in "st", and show their name, conname, transmit queue, sslciph, and the name of the queue manager they are on

```
PS C:\> Get-WMQChannel * *st | Where {$_.Name -notlike "SYSTEM.*" -and $_.ChannelType -eq “Sender”} | Select Name, ConnectionName,
TransmissionQueueName, SSLCipherSpec, @{e={$_.QueueManager.Name};n='Hosting Queue Manager'}

Name                 ConnectionName                    TransmissionQueueName           SSLCipherSpec              Hosting Queue Manager
----                 --------------                    ---------------------           -------------              ---------------------
SECURE               dlane.hursley.ibm.com(9090)       TRANS1                          NULL_MD5                   post
SECURE.R             dlane.hursley.ibm.com(9091)       TRANSR                          TRIPLE_DES_SHA_US          test
```

generating HTML reports

generate an HTML webpage with a table showing the name, description and depth information for all queues on the 'test' queue manager, and open this HTML file in a web-browser

```
PS C:\> Get-WMQQueue * test | ConvertTo-Html -property Name,Description,CurrentDepth,MaximumDepth -title "Queues
on my test queue manager" > myqueues.htm
PS C:\> Invoke-Item myqueues.htm
```

generating CSV spreadsheets

generate a CSV spreadsheet containing the name, description and depth information for all queues on the 'test' queue manager, and open this in Excel

```
PS C:\> Get-WMQQueue * test | Select Name, Description, CurrentDepth, MaximumDepth | Export-Csv -path
myqueues.csv
PS C:\> Invoke-Item myqueues.csv
```

---

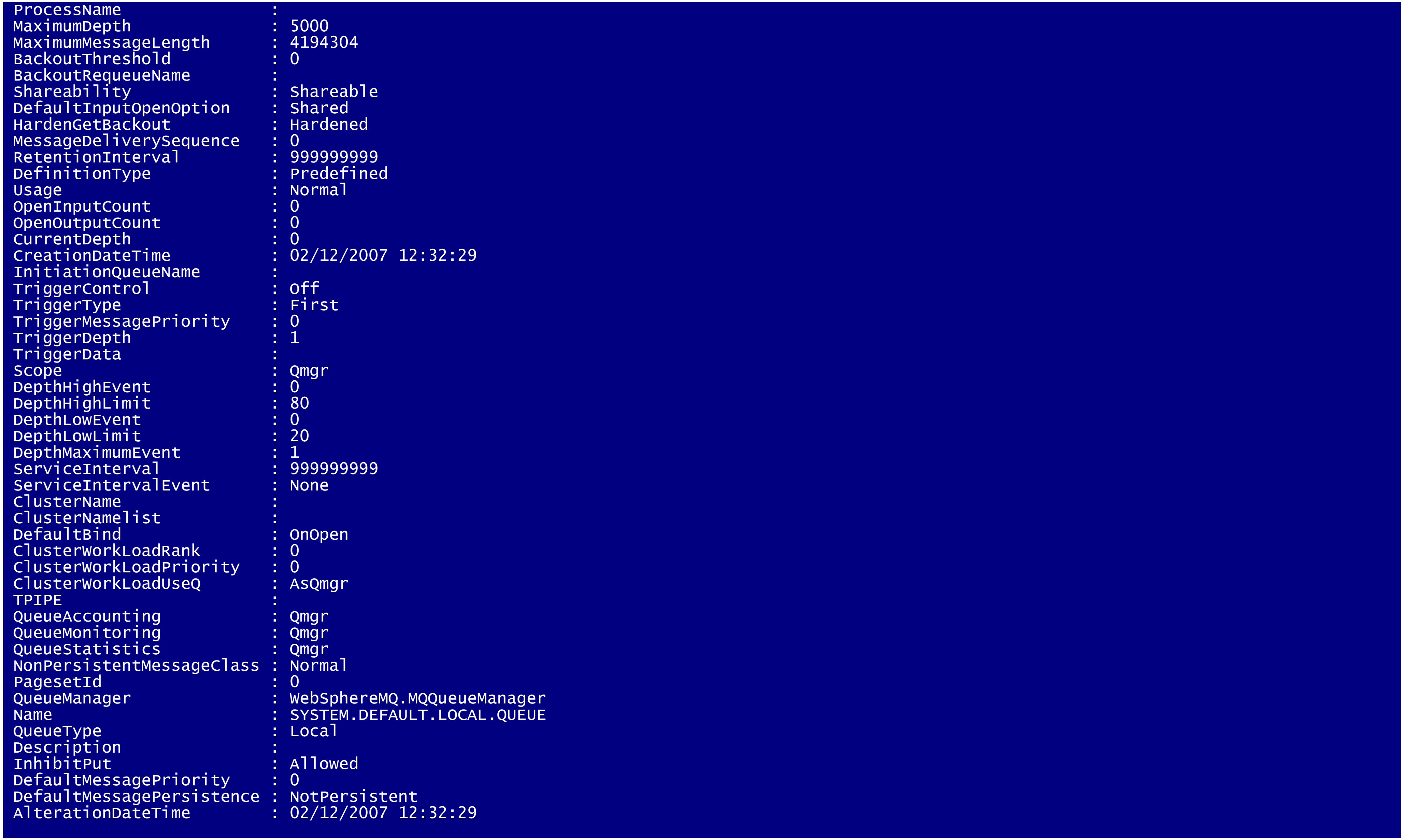
using constants

note: a useful utility function

```
function Get-EnumValues{
    [enum]::getvalues($args[0]) | select @{n="Name";e={$_}},@{n="Value";e={$_.value__}} | ft -auto
}

PS C:\> Get-EnumValues([WebSphereMQ.MQC+InhibitGetTypes])

     Name Value
     ---- -----
  Allowed     0
Inhibited     1
```

get a queue...

```
PS C:\> Get-WMQQueue SYSTEM.DEFAULT.LOCAL.QUEUE


InhibitGet                : Allowed
```



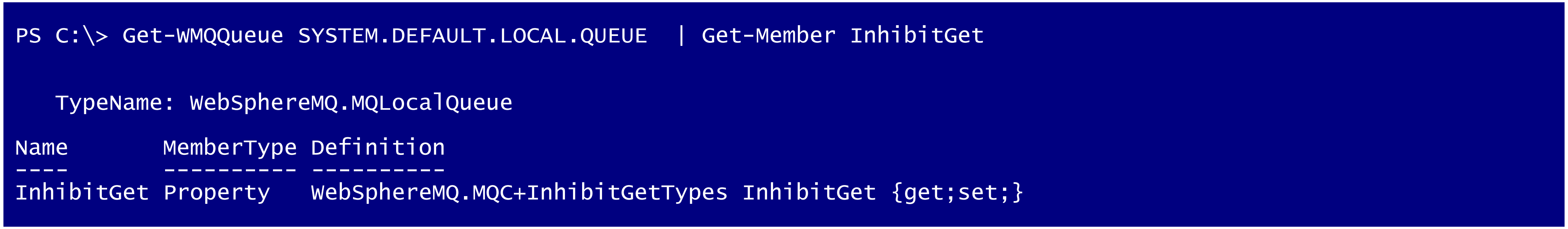
```
ProcessName                :
MaximumDepth               : 5000
MaximumMessageLength       : 4194304
BackoutThreshold           : 0
BackoutRequeueName         :
Shareability               : Shareable
DefaultInputOpenOption     : Shared
HardenGetBackout           : Hardened
MessageDeliverySequence    : 0
RetentionInterval          : 999999999
DefinitionType             : Predefined
Usage                      : Normal
OpenInputCount             : 0
OpenOutputCount            : 0
CurrentDepth               : 0
CreationDateTime           : 02/12/2007 12:32:29
InitiationQueueName        :
TriggerControl             : Off
TriggerType                : First
TriggerMessagePriority     : 0
TriggerDepth               : 1
TriggerData                :
Scope                      : Qmgr
DepthHighEvent             : 0
DepthHighLimit             : 80
DepthLowEvent              : 0
DepthLowLimit              : 20
DepthMaximumEvent          : 1
ServiceInterval            : 999999999
ServiceIntervalEvent       : None
ClusterName                :
ClusterNamelist            :
DefaultBind                : OnOpen
ClusterWorkLoadRank        : 0
ClusterWorkLoadPriority    : 0
ClusterWorkLoadUseQ        : AsQmgr
TPIPE                      :
QueueAccounting            : Qmgr
QueueMonitoring            : Qmgr
QueueStatistics            : Qmgr
NonPersistentMessageClass  : Normal
PagesetId                  : 0
QueueManager               : WebSphereMQ.MQQueueManager
Name                       : SYSTEM.DEFAULT.LOCAL.QUEUE
QueueType                  : Local
Description                :
InhibitPut                 : Allowed
DefaultMessagePriority     : 0
DefaultMessagePersistence  : NotPersistent
AlterationDateTime         : 02/12/2007 12:32:29
```

I can see that I’m allowed to GET from it. I want to stop that...

At the moment, InhibitGet is set to 'Allowed'.

How do I know the right term for 'not allowed'?

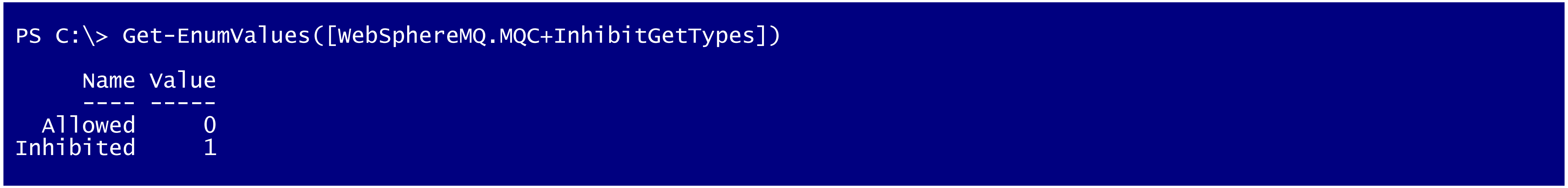
Using Get-Member



```
PS C:\> Get-WMQQueue SYSTEM.DEFAULT.LOCAL.QUEUE  | Get-Member InhibitGet


   TypeName: WebSphereMQ.MQLocalQueue

Name       MemberType Definition
----       ---------- ----------
InhibitGet Property   WebSphereMQ.MQC+InhibitGetTypes InhibitGet {get;set;}
```

Seeing the '+' symbol tells me the type is an enum called InhibitGetTypes in the WebSphereMQ.MQC class.

What are the valid elements in this enum?



```
PS C:\> Get-EnumValues([WebSphereMQ.MQC+InhibitGetTypes])

     Name Value
     ---- -----
  Allowed     0
Inhibited     1
```

This tells me I can either just set InhibitGet to 1 ...

```
PS C:\> Get-WMQQueue SYSTEM.DEFAULT.LOCAL.QUEUE  | Set-WMQQueue -InhibitGet 1
PS C:\> Get-WMQQueue SYSTEM.DEFAULT.LOCAL.QUEUE  | Select Name, InhibitGet

Name                                                                 InhibitGet
----                                                                 ----------
SYSTEM.DEFAULT.LOCAL.QUEUE                                            Inhibited
```

or be a little more verbose and readable by using the enum name ...

```
PS C:\> Get-WMQQueue SYSTEM.DEFAULT.LOCAL.QUEUE | Set-WMQQueue -InhibitGet ([WebSphereMQ.MQC+InhibitGetTypes]::Inhibited)
PS C:\> Get-WMQQueue SYSTEM.DEFAULT.LOCAL.QUEUE | Select Name, InhibitGet

Name                                                                 InhibitGet
----                                                                 ----------
SYSTEM.DEFAULT.LOCAL.QUEUE                                            Inhibited
```

Note that the absolute name of the enum is not required. If you leave it out, it is implicitly added...

```
PS C:\> Get-WMQQueue | Where { $_.InhibitGet –eq “Inhibited” }
PS C:\> Get-WMQQueue SYSTEM.DEFAULT.LOCAL.QUEUE | Set-WMQQueue -InhibitGet Inhibited
PS C:\>
PS C:\> Get-WMQQueue SYSTEM.DEFAULT.LOCAL.QUEUE | Select Name, InhibitGet

Name                                                                 InhibitGet
----                                                                 ----------
SYSTEM.DEFAULT.LOCAL.QUEUE                                            Inhibited
```

Relying on feedback

Alternatively, I could always have been lazy and made an intentional error - and used the error message feedback to guide me...

```
PS C:\> Get-WMQQueue SYSTEM.DEFAULT.LOCAL.QUEUE | Set-WMQQueue -InhibitGet THIS_DEFINITELY_SHOULDNT_WORK
Set-WMQQueue : Cannot bind parameter 'InhibitGet'.
Cannot convert value "THIS_DEFINITELY_SHOULDNT_WORK" to type "WebSphereMQ.MQC+InhibitGetTypes" due to invalid enumeration values.
Specify one of the following enumeration values and try again.
The possible enumeration values are "Allowed, Inhibited".
```

Using PCF constants

Finally, if you are used to the standard IBM constant names, the implementation of WebSphere MQ for PowerShell is such that it uses the same values

```
PS C:\> Get-WMQQueue SYSTEM.DEFAULT.LOCAL.QUEUE | Set-WMQQueue -InhibitGet ([IBM.WMQ.MQC]::MQQA_GET_INHIBITED)
PS C:\> Get-WMQQueue SYSTEM.DEFAULT.LOCAL.QUEUE | Select Name, InhibitGet

Name                                                                 InhibitGet
----                                                                 ----------
SYSTEM.DEFAULT.LOCAL.QUEUE                                            Inhibited
```

---

equivalent approaches to pipelining

These all do the same thing…

```
PS C:\> Get-WMQQueue * (Get-WMQQueueManager DALE) | Select Name
PS C:\> Get-WMQQueue -QmgrName DALE | Select Name
PS C:\> Get-WMQQueue -Qmgr (Get-WMQQueueManager DALE) | Select Name
PS C:\> Get-WMQQueueManager DALE | Get-WMQQueue * | Select Name
```

…similarly, if you want to use a couple of lines…

```
PS C:\> $qm =  Get-WMQQueueManager DALE
PS C:\> Get-WMQQueue * $qm | Select Name
```

---

requiring confirmation

Commands which result in a change to WMQ objects have a confirm level of MEDIUM.

If your ConfirmPreference is higher than this, you will not be prompted for confirmation:

```
PS C:\> dir Variable:\ConfirmPreference

Name                           Value
----                           -----
ConfirmPreference              High


PS C:\> Get-WMQQueue TESTQ DALE | Set-WMQQueue -InhibitPut Allowed
PS C:\>
PS C:\>
PS C:\> Get-WMQQueue TESTQ DALE | Select Name, InhibitPut

Name                        InhibitPut
----                        ----------
TESTQ                          Allowed
```

If your ConfirmPreference is equal to or lower than this, you will be prompted for confirmation:

```
PS C:\> Set-Variable ConfirmPreference Medium
PS C:\>
PS C:\> Get-WMQQueue TESTQ DALE | Set-WMQQueue -InhibitPut Inhibited

Confirm change to WebSphere MQ object
Modifying put inhibit of TESTQ on DALE to 1
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"): Y
PS C:\> Get-WMQQueue TESTQ DALE | Select Name, InhibitPut

Name                        InhibitPut
----                        ----------
TESTQ                        Inhibited
```

If you already have ConfirmPreference set to low/medium, then you can override the prompt for confirmation:

```
PS C:\> dir Variable:\ConfirmPreference

Name                           Value
----                           -----
ConfirmPreference              Medium


PS C:\> Get-WMQQueue TESTQ DALE | Set-WMQQueue -InhibitPut Allowed -Force
PS C:\> Get-WMQQueue TESTQ DALE | Select Name, InhibitPut

Name                        InhibitPut
----                        ----------
TESTQ                          Allowed
```

If you already have ConfirmPreference set to high, then you can override the need for confirmation:

```
PS C:\> dir Variable:\ConfirmPreference

Name                           Value
----                           -----
ConfirmPreference              High


PS C:\> Get-WMQQueue TESTQ DALE | Set-WMQQueue -InhibitPut Inhibited -Confirm

Confirm change to WebSphere MQ object
Modifying put inhibit of TESTQ on DALE to 1
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"): Y
PS C:\>
PS C:\> Get-WMQQueue TESTQ DALE | Select Name, InhibitPut

Name                        InhibitPut
----                        ----------
TESTQ                        Inhibited
```

comparing current state with a previous known good state

Store information about all queues which have a "CurrentDepth" value in a CSV file

```
PS C:\> Get-WMQQueue -QmgrName DALE | where { $_.CurrentDepth } | Export-Csv dalesqueues.csv
```

Change the state

```
PS C:\> $testmessage = New-WMQMessage –StringData “Hello World”
PS C:\>
PS C:\> Send-WMQMessage $testmessage (Get-WMQQueue TESTQ DALE)
```

Compare based on queue depth

```
PS C:\> Compare-Object (Import-Csv dalesqueues.csv) (Get-WMQQueue -QmgrName DALE | where
{$_.CurrentDepth }) -property name, currentdepth

name                                           currentdepth SideIndicator
----                                           ------------ -------------
TESTQ                                                     3 =>
TESTQ                                                     2 <=
```

delete all non-system objects

for example, non-system process objects

```
PS C:\> Get-WMQProcess | where {$_.Name -notlike "SYSTEM.*"} | Remove-WMQProcess
```

---

modify running objects

Modify a running service

```
PS C:\> Get-WMQService SERV TESTQM | Stop-WMQService -PassThru | Set-WMQService -StartCommand
'newstartcommandhere' -PassThru | Start-WMQService
```

Modify a running listener

```
PS C:\> Get-WMQListener LISTR TESTQM | Stop-WMQListener -PassThru | Set-WMQListener -Port 9999 -PassThru
| Start-WMQListener
```

These commands should work in theory, but in practice they will likely fail – as Stop and Start cmdlets do not wait for the operations to complete before returning. This will be addressed with a –Wait parameter in a future version.

---

sending messages

```
PS C:\>
PS C:\> Get-WMQQueue FIS* | Select Name, CurrentDepth

Name                                          CurrentDepth
----                                          ------------
FISH                                                     0
FISH2                                                    0
FISH3                                                    0


PS C:\> $testmessage = New-WMQMessage
PS C:\> $testmessage.Format = [IBM.WMQ.MQC]::MQFMT_STRING
PS C:\> $testmessage.WriteString("Hello World")
PS C:\>
PS C:\> $testmessage2 = New-WMQMessage –StringData “Hello World again”
PS C:\>
PS C:\> Send-WMQMessage $testmessage (Get-WMQQueue FIS*)
PS C:\> Send-WMQMessage $testmessage (Get-WMQQueue FIS*)
PS C:\>
PS C:\> Get-WMQQueue FIS* | Select Name,CurrentDepth

Name                                          CurrentDepth
----                                          ------------
FISH                                                     2
FISH2                                                    2
FISH3                                                    2
```

receiving messages

```
PS C:\> $mymessage = Receive-WMQMessage (Get-WMQQueue FISH)
PS C:\> $mymessage.ReadString($mymessage.MessageLength)
Hello World
```

---

sending the string contents of a text file

```
PS C:\> $message = New-WMQMessage -StringData (Get-Content testdata.txt)
PS C:\> Send-WMQMessage $message (Get-WMQQueue TESTQUEUE)
```

or in one line:

```
PS C:\> Send-WMQMessage (New-WMQMessage -StringData (Get-Content testdata.txt)) (Get-WMQQueue TESTQUEUE)
```

---

receiving string message data into a text file

```
PS C:\> $received = Receive-WMQMessage (Get-WMQQueue TESTQUEUE)
PS C:\> $received.ReadString($received.MessageLength) | Out-File testdata2.txt
```

getting less information by default

Get commands display all attributes of all objects by default – which can be a lot of information. This can be reduced to just the fields you are interested in.

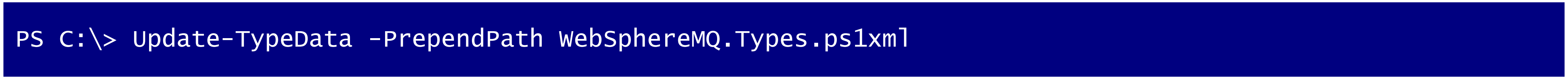
For example, for queues…

Instead of displaying all attributes for each queue, what if you are normally only interested in names, queue type and description? This can be done with a type data file.

Create a file called **`WebSphereMQ.Types.ps1xml`**

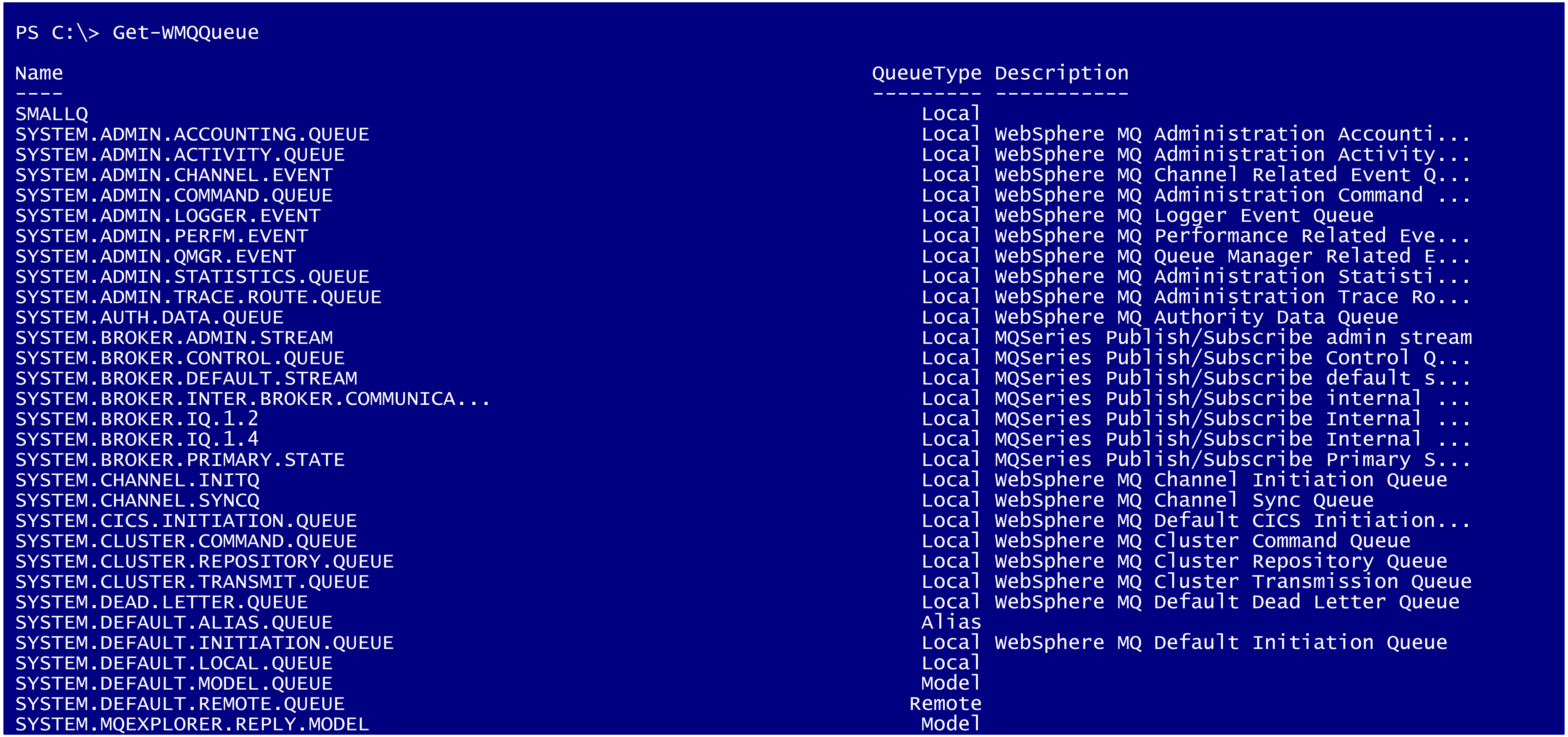
```
<Types>
 <Type>
    <Name>WebSphereMQ.MQQueue</Name>
    <Members>
        <MemberSet>
            <Name>PSStandardMembers</Name>
            <Members>
                <NoteProperty>
                    <Name>DefaultDisplayProperty</Name>
                    <Value>ReferencedPropertyName</Value>
                </NoteProperty>
                <PropertySet>
                    <Name>DefaultDisplayPropertySet</Name>
                    <ReferencedProperties>
                        <Name>Name</Name>
                        <Name>QueueType</Name>
                        <Name>Description</Name>
                    </ReferencedProperties>
                </PropertySet>
                <PropertySet>
                    <Name>DefaultKeyPropertySet</Name>
                    <ReferencedProperties>
                        <Name>Name</Name>
                    </ReferencedProperties>
                </PropertySet>
            </Members>
        </MemberSet>
    </Members>
 </Type>
</Types>
```

Then in a PowerShell window, update the type data with the information in the file:



```
PS C:\> Update-TypeData -PrependPath WebSphereMQ.Types.ps1xml
```

After this, WebSphereMQ.MQQueue objects will be displayed in a table with name, queue type and description:



```
PS C:\> Get-WMQQueue

Name                                                                  QueueType Description
----                                                                  --------- -----------
SMALLQ                                                                    Local
SYSTEM.ADMIN.ACCOUNTING.QUEUE                                             Local WebSphere MQ Administration Accounti...
SYSTEM.ADMIN.ACTIVITY.QUEUE                                               Local WebSphere MQ Administration Activity...
SYSTEM.ADMIN.CHANNEL.EVENT                                                Local WebSphere MQ Channel Related Event Q...
SYSTEM.ADMIN.COMMAND.QUEUE                                                Local WebSphere MQ Administration Command ...
SYSTEM.ADMIN.LOGGER.EVENT                                                 Local WebSphere MQ Logger Event Queue
SYSTEM.ADMIN.PERFM.EVENT                                                  Local WebSphere MQ Performance Related Eve...
SYSTEM.ADMIN.QMGR.EVENT                                                   Local WebSphere MQ Queue Manager Related E...
SYSTEM.ADMIN.STATISTICS.QUEUE                                             Local WebSphere MQ Administration Statisti...
SYSTEM.ADMIN.TRACE.ROUTE.QUEUE                                            Local WebSphere MQ Administration Trace Ro...
SYSTEM.AUTH.DATA.QUEUE                                                    Local WebSphere MQ Authority Data Queue
SYSTEM.BROKER.ADMIN.STREAM                                                Local MQSeries Publish/Subscribe admin stream
SYSTEM.BROKER.CONTROL.QUEUE                                               Local MQSeries Publish/Subscribe Control Q...
SYSTEM.BROKER.DEFAULT.STREAM                                              Local MQSeries Publish/Subscribe default s...
SYSTEM.BROKER.INTER.BROKER.COMMUNICA...                                   Local MQSeries Publish/Subscribe internal ...
SYSTEM.BROKER.IQ.1.2                                                      Local MQSeries Publish/Subscribe Internal ...
SYSTEM.BROKER.IQ.1.4                                                      Local MQSeries Publish/Subscribe Internal ...
SYSTEM.BROKER.PRIMARY.STATE                                               Local MQSeries Publish/Subscribe Primary S...
SYSTEM.CHANNEL.INITQ                                                      Local WebSphere MQ Channel Initiation Queue
SYSTEM.CHANNEL.SYNCQ                                                      Local WebSphere MQ Channel Sync Queue
SYSTEM.CICS.INITIATION.QUEUE                                              Local WebSphere MQ Default CICS Initiation...
SYSTEM.CLUSTER.COMMAND.QUEUE                                              Local WebSphere MQ Cluster Command Queue
SYSTEM.CLUSTER.REPOSITORY.QUEUE                                           Local WebSphere MQ Cluster Repository Queue
SYSTEM.CLUSTER.TRANSMIT.QUEUE                                             Local WebSphere MQ Cluster Transmission Queue
SYSTEM.DEAD.LETTER.QUEUE                                                  Local WebSphere MQ Default Dead Letter Queue
SYSTEM.DEFAULT.ALIAS.QUEUE                                                Alias
SYSTEM.DEFAULT.INITIATION.QUEUE                                           Local WebSphere MQ Default Initiation Queue
SYSTEM.DEFAULT.LOCAL.QUEUE                                                Local
SYSTEM.DEFAULT.MODEL.QUEUE                                                Model
SYSTEM.DEFAULT.REMOTE.QUEUE                                              Remote
SYSTEM.MQEXPLORER.REPLY.MODEL                                             Model
SYSTEM.MQSC.REPLY.QUEUE                                                   Model WebSphere MQ MQSC Reply Queue
SYSTEM.PENDING.DATA.QUEUE                                                 Local WebSphere MQ Deferred Message Queue
TESTQ                                                                     Local
XMITQ                                                                     Local
```

The other properties are still accessible – just not displayed by default. Pipe the output from a Get-WMQQueue command to a Select command identifying the properties to display for individual commands.

```
PS C:\> Get-WMQQueue | Select Name, InhibitPut, InhibitGet, OpenInputCount

Name                                             InhibitPut                    InhibitGet                OpenInputCount
----                                             ----------                    ----------                --------------
SMALLQ                                              Allowed                       Allowed                             0
SYSTEM.ADMIN.ACCOUNTING.QUEUE                       Allowed                       Allowed                             0
SYSTEM.ADMIN.ACTIVITY.QUEUE                         Allowed                       Allowed                             0
SYSTEM.ADMIN.CHANNEL.EVENT                          Allowed                       Allowed                             0
SYSTEM.ADMIN.COMMAND.QUEUE                          Allowed                       Allowed                             1
SYSTEM.ADMIN.LOGGER.EVENT                           Allowed                       Allowed                             0
SYSTEM.ADMIN.PERFM.EVENT                            Allowed                       Allowed                             0
SYSTEM.ADMIN.QMGR.EVENT                             Allowed                       Allowed                             0
SYSTEM.ADMIN.STATISTICS.QUEUE                       Allowed                       Allowed                             0
SYSTEM.ADMIN.TRACE.ROUTE.Q...                       Allowed                       Allowed                             0
...
```

---

adding additional properties to WebSphere MQ objects

Consider the following:

```
PS C:\> Get-WMQQueue SYSTEM.DEFAULT.LOCAL.QUEUE | Select Name, @{e={$_.QueueManager.Name};n='Qmgr'}

Name                                                          Qmgr
----                                                          ----
SYSTEM.DEFAULT.LOCAL.QUEUE                                    APPSQM
SYSTEM.DEFAULT.LOCAL.QUEUE                                    SANDBOX
SYSTEM.DEFAULT.LOCAL.QUEUE                                    TESTQMGR
```

Getting the name of the queue manager hosting an object can be made easier by adding a custom hostqmgrname property to objects.

This can be done with a type data file. Create a file called **`WebSphereMQ.Types.ps1xml`**

```
<Types>
 <Type>
    <Name>WebSphereMQ.MQQueue</Name>
    <Members>
        <ScriptProperty>
            <Name>HostQmgrName</Name>
           <GetScriptBlock>$this.QueueManager.Name</GetScriptBlock>
        </ScriptProperty>
    </Members>
 </Type>
</Types>
```

Then in a PowerShell window, update the type data with the information in the file:

```
PS C:\> Update-TypeData -PrependPath WebSphereMQ.Types.ps1xml
```

After this, queue objects will include a HostQmgrName property:

```
PS C:\> Get-WMQQueue SYSTEM.DEFAULT.LOCAL.QUEUE | Select Name, @{e={$_.QueueManager.Name};n='Qmgr'}, HostQmgrName

Name                                    Qmgr                                    HostQmgrName
----                                    ----                                    ------------
SYSTEM.DEFAULT.LOCAL.QUEUE              APPSQM                              ... APPSQM
SYSTEM.DEFAULT.LOCAL.QUEUE              SANDBOX                             ... SANDBOX
SYSTEM.DEFAULT.LOCAL.QUEUE              TESTQMGR                            ... TESTQMGR
```

---

getting the SSL properties of all queue managers

Get all SSL properties from queue managers

```
PS C:\> Get-WMQQueueManager | Select Name, SSL*


Name                 : APPSQM
SSLCRLNamelist       :
SSLCryptoHardware    :
SSLEvent             : Disabled
SSLFips              : No
SSLKeyRepository     : C:\Program Files\IBM\WebSphere MQ\qmgrs\APPSQM\ssl\key
SSLKeyResetCount     : 0
SSLTasks             : 0

Name                 : qmgr1fish
SSLCRLNamelist       :
SSLCryptoHardware    :
SSLEvent             : Disabled
SSLFips              : No
SSLKeyRepository     : C:\Program Files\IBM\WebSphere MQ\qmgrs\qmgr1fish\ssl\key
SSLKeyResetCount     : 0
SSLTasks             : 0
```

check that queue managers have the right SSL key repository file

get the SSL key repository property from the queue manager, and check that a file exists at that location (appending the correct file extension)

```
PS C:\> Get-WMQQueueManager | Select Name, @{e={ ( $_.SSLKeyRepository + ".kdb" ) | Test-Path };n='Exists'}

Name                                                                                                   Exists
----                                                                                                   ------
APPSQM                                                                                                  False
qmgr1SSL                                                                                                 True
qmgr2SSL                                                                                                 True
SANDBOX                                                                                                 False
TESTQMGR                                                                                                False
```

check that all channels have valid transmission queues

for every channel, check that it’s transmission queue name refers to an actual queue with usage set to Transmission and InhibitPut set to Allowed

```
PS C:\> Get-WMQChannel -QmgrName qmgr1test | Where {$_.Name -notlike "SYSTEM.*" } | Select Name,
TransmissionQueueName, @{e={ (Get-WMQQueue -qmgrname qmgr1test -Name ($_.TransmissionQueueName) | Where
{$_.Usage -eq “Transmission” -and $_.InhibitPut -eq “Allowed”}) -ne $null }; n='Valid transmission queue'}

Name                                 TransmissionQueueName                           Valid transmission queue
----                                 ---------------------                           ------------------------
broken                               nonexistent                                                        False
qmgr1test.qmgr2test                  xmit_queue                                                          True
```

In reality, this is stretching how much you would want to do in a single line – as it has lost the readability benefits that PowerShell should bring. This could be expanded into a function, making it easier to read and maintain.

However, once you get used to using PowerShell, it is possible to write lines like this interactively. And it shows how powerful the sort of queries you can write can be.

---

getting remote queue managers

specify connection information to remote queue managers, and use that to include remote queue managers in the objects returned by the Get-WMQQueueManager

```
PS C:\> $qmconns = @()
PS C:\> $qmconns += New-WMQQmgrConnDef -Name DALEQM -Hostname dlane.hursley.ibm.com -Channel SVRCN -Port 1414
PS C:\> $qmconns += New-WMQQmgrConnDef -Name CENTQM -Hostname sysserv.boulder.ibm.com -Channel SVRCN -Port 1418
PS C:\>
PS C:\> $qmgrs = Get-WMQQueueManager -Connections $qmconns
```

These queue manager objects can then be used in the same way as local queue managers.

---

specifying a subset of known remote queue managers

filter the returned queue managers based on their properties

```
PS C:\> $remoteqmgrs = Get-WMQQueueManager –Connections $qmconns | Where { $_.Hostname –like “*.hursley.ibm.com” }
```

Other such filters could be Where { $_.Platform –eq ‘UNIX’ }

getting properties of remote queue managers

get objects from remote queue managers, and show them together with properties of the queue manager that they are on

```
PS C:\> $conns = @()
PS C:\> $conns += New-WMQQmgrConnDef -Name DALEQM -Hostname dlane.hursley.ibm.com -Channel SVRCN -Port 1414
PS C:\> $conns += New-WMQQmgrConnDef -Name CENTQM -Hostname sysserv.boulder.ibm.com -Channel SVRCN -Port 1418
PS C:\>
PS C:\> $qmgrs = Get-WMQQueueManager -Connections $conns
PS C:\>
PS C:\> Get-WMQQueue -Qmgr $qmgrs MYTESTQ* | Select Name, @{e={$_.QueueManager.Name};n='Qmgr'},
@{e={$_.QueueManager.Platform};n='Platform'}, @{e={$_.QueueManager.Hostname};n='Hostname'}

Name                                    Qmgr                     Platform Hostname
----                                    ----                     -------- --------
MYTESTQ1                                ALOCALQMGR                Windows
MYTESTQX                                DALEQM                       UNIX dlane.hursley.ibm.com
MYTESTQA                                CENTQM                       UNIX sysserv.boulder.ibm.com
```

---

getting remote queue manager info from WebSphere MQ Explorer

if you’ve already taken the time to specify the connection details for your remote queue managers in WebSphere MQ Explorer (v6), why not reuse that?

```
PS C:\> $qmconns = Import-WMQQmgrConnDef "C:\Documents and Settings\Administrator\Application Data\IBM\MQ
Explorer\.metadata\.plugins\com.ibm.mq.explorer.ui\WMQ_Handles.xml"
```

Note that the exact path will vary depending on installation and user.

---

script multiple profiles for WebSphere MQ Explorer

the reverse is also possible – allowing PowerShell to be used to script the configuring of WebSphere MQ Explorer connections

```
PS C:\> Export-WMQQmgrConnDef -Connections $qmconns -Path "C:\Documents and Settings\Administrator\Application
Data\IBM\MQ Explorer\.metadata\.plugins\com.ibm.mq.explorer.ui\WMQ_Handles.xml"
```

Note that the exact path will vary depending on installation and user.
Further note that WebSphere MQ Explorer reads this file on start-up, so Explorer should be stopped before exporting.

---

getting queues from UNIX queue managers only

```
PS C:\> Get-WMQQueue Q* -Qmgr ( Get-WMQQueueManager -Connections $qmconndefs | where { $_.Platform -like
'UNIX' } ) | Select Name, @{e={$_.QueueManager.Name};n='Queue Manager'},
@{e={$_.QueueManager.Hostname};n='Server'}

Name                                         Queue Manager                  Server
----                                         -------------                  ------
QUEUE991                                     DALEQM                         dlane.hursley.ibm.com
QUEUE999                                     DALEQM                         dlane.hursley.ibm.com
QUEUE828                                     CENTQM                         sysserv.boulder.ibm.com
```
