[SAS Renewal Automation](https://confluence.betagy.com/confluence/display/DBA/SAS+Renewal+Automation)

SQL Server Agent Job was created on *DB-VM3-PROD-AZ2\INST3* with name *DBAToolbox - Renew SAS Access Policy.*This job will in turn execute a Powershell script [Renew SAS.ps1](https://confluence.betagy.com/confluence/download/attachments/50070333/Renew%20SAS.ps1?version=2&modificationDate=1586936161883&api=v2) which will:

* Create a new Access Policy for the storage container
* Check if a database credential exists for the storage account, and if so drop the credential.
* Create new credential for the storage with new secret

Important parameters

| **Parameter** | **Purpose** |
| --- | --- |
| instancesToCheck | Array of SQL Server instances where the credential will be updated. Script is designed to run from single machine but update credential on multiple instances. |
| ExpiryDaysPeriod | Number of days before the access policy expires. **Changes to this parameter should be reflected in changes to the Agent job schedule**. |
| storageAccountName | Name of storage account. Needs to be updated if script is to be used on different account. |
| containerName | Name of container. Needs to be updated if script is to be used on different container as Access Policies are container specific. |

Notes

* Storage Account Key needed to connect to the Azure storage account is stored in encrypted format in a separate file. The script used to create this file is attached [Encrypt Storage Access Key.ps1](https://confluence.betagy.com/confluence/download/attachments/50070333/Encrypt%20Storage%20Access%20Key.ps1?version=1&modificationDate=1586937946348&api=v2)
* Job failures are still shown as successful in the SQL Agent job history. To get around this an email notification will be sent whenever this job is executed. If job fails, the notification will be highlighted as High Priority, otherwise it is just for informational purposes.

# Custom Parameters

$instancesToCheck = 'DB-VM3-PROD-AZ2\INST3'

$ExpiryDaysPeriod = 100

# Script Parameters

$subscriptionID = '432fc7b1-3da7-4408-9f23-367a4f18e474' # the ID of subscription name you will use

$storageAccountName = 'dbbtgsqlbackups' # the storage account name you will create or use

$containerName = 'sql-backups-availabilitygroup'

$dt = Get-Date -Format "yyyyMMdd-HHmm"

$policyName = 'Prod-DB-Backups-policy-' + $dt

$key = New-Object byte[](128)

$Key = (143,57,184,185, 49, 120, 225, 229, 174, 128, 90, 1, 133, 152, 110, 54,197, 231, 113, 78, 202, 231, 19, 77, 117, 237, 116, 255, 97, 12, 195, 85)

$keyString = [System.Text.Encoding]::Unicode.GetString($Key)

$secureKey = ConvertTo-SecureString -String $keyString -AsPlainText -Force

$encryptedSecureString = Get-Content S:\MSSQL14.INST3\MSSQL\JOBS\SecureStrings-AES.SecureString.txt

$secureString = ConvertTo-SecureString -String $encryptedSecureString -SecureKey $secureKey

$cred = New-Object System.Management.Automation.PSCredential('UserName', $secureString)

$StorageAccountKey = $cred.GetNetworkCredential().Password

#Email Settings

$From = "dbalerts@betagy.com"

$MyCreds = New-Object System.Management.Automation.PSCredential("Betking\_SendGrid", (ConvertTo-SecureString "imeklTL86i9un1zR5Ffg" -AsPlainText -Force))

$SmtpSettings = @{

From = $From

To = "db-replication@betagy.com" #“b.adlington@betagy.com”

Subject = “SAS Renewal Status"

SmtpServer = “smtp.sendgrid.net”

Credential = $MyCreds

Port = 587

}

$Style = "<style>BODY{font-family: Arial; font-size: 8pt;}"

$Style = $Style + "TABLE{border: 1px solid black; border-collapse: collapse;}"

$Style = $Style + "TH{border: 1px solid black; background: #dddddd; padding: 5px; }"

$Style = $Style + "TD{border: 1px solid black; padding: 5px; }"

$Style = $Style + "</style>"

$mailBody = $Style + "<table><tr><th>Instance</th><th>Policy Name</th><th>Expiry Date</th></tr>"

try

{

# Create a new storage account context using an Azure Resource Manager storage account

$Context = New-AzStorageContext -StorageAccountName $StorageAccountName -StorageAccountKey $StorageAccountKey

# Sets up a Stored Access Policy and a Shared Access Signature for the new container

$policy = New-AzStorageContainerStoredAccessPolicy -Container $containerName -Policy $policyName -Context $Context -StartTime $(Get-Date).ToUniversalTime().AddMinutes(-5) -ExpiryTime $(Get-Date).ToUniversalTime().AddDays($ExpiryDaysPeriod) -Permission rwld

# Gets the Shared Access Signature for the policy

$sas = New-AzStorageContainerSASToken -name $containerName -Policy $policyName -Context $Context

Write-Host 'Shared Access Signature= '$($sas.Substring(1))''

# Sets the variables for the new container you just created

$container = Get-AzStorageContainer -Context $Context -Name $containerName

$cbc = $container.CloudBlobContainer

# Outputs the Transact SQL to the clipboard and to the screen to create the credential using the Shared Access Signature

Write-Host 'Credential T-SQL'

$tSql = "IF EXISTS (SELECT 1 FROM sys.credentials where name='{0}')

BEGIN DROP CREDENTIAL [{0}] END

CREATE CREDENTIAL [{0}] WITH IDENTITY='Shared Access Signature', SECRET='{1}'" -f $cbc.Uri,$sas.Substring(1)

$tSql | clip

Write-Host $tSql

foreach ($SqlSrv in $instancesToCheck) {

Invoke-Sqlcmd -Query $tSql -ServerInstance $SqlSrv

$mailBody = $mailBody + "<tr><td>$SqlSrv</td><td>$policyName</td><td>"+(Get-Date).ToUniversalTime().AddDays($ExpiryDaysPeriod)+"</td></tr>"

}

$mailBody = $mailBody + "</table>"

Send-MailMessage @SmtpSettings -Body $mailBody -UseSsl -BodyAsHtml

}

catch

{

foreach ($SqlSrv in $instancesToCheck) {

$mailBody = $mailBody + "<tr><td>$SqlSrv</td><td>$policyName</td><td style=""color:red;font-weight:bold"">Renewal Failed</td></tr>"

}

Send-MailMessage @SmtpSettings -Body $mailBody -UseSsl -BodyAsHtml -Priority High

}