Windows PowerShell Get-Help on Cmdlet 'Set-AzDataLakeGen2AclRecursive'

PS:\>Get-HELP Set-AzDataLakeGen2AclRecursive -Full

NAME

Set-AzDataLakeGen2AclRecursive

SYNOPSIS

Set ACL recursively on the specified path.

SYNTAX

Set-AzDataLakeGen2AclRecursive [-FileSystem] <System.String> [[-Path] <System.String>] -Acl <Microsoft.WindowsAzure.Commands.Storage.Model.ResourceModel.PSPathAccessControlEntry[]> [-AsJob] [-BatchSize <System.Int32>] [-Context <Microsoft.Azure.Commands.Common.Authentication.Abstractions.IStorageContext>] [-ContinuationToken <System.String>] [-ContinueOnFailure] [-DefaultProfile <Microsoft.Azure.Commands.Common.Authentication.Abstractions.Core.IAzureContextContainer>] [-MaxBatchCount <System.Int32>] [-Confirm] [-WhatIf] [<CommonParameters>]

DESCRIPTION

The Set-AzDataLakeGen2AclRecursive cmdlet sets ACL recursively on the specified path. The input ACL will replace original ACL completely.

PARAMETERS

-Acl <Microsoft.WindowsAzure.Commands.Storage.Model.ResourceModel.PSPathAccessControlEntry[]>

The POSIX access control list to set recursively for the file or directory.

Required? true

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-AsJob <System.Management.Automation.SwitchParameter>

Run cmdlet in the background

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-BatchSize <System.Int32>

If data set size exceeds batch size then operation will be split into multiple requests so that progress can be tracked. Batch size should be between 1 and 2000.

Default is 2000.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-Context <Microsoft.Azure.Commands.Common.Authentication.Abstractions.IStorageContext>

Azure Storage Context Object

Required? false

Position? named

Default value None

Accept pipeline input? True (ByPropertyName, ByValue)

Accept wildcard characters? false

-ContinuationToken <System.String>

Continuation Token.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-ContinueOnFailure <System.Management.Automation.SwitchParameter>

Set this parameter to ignore failures and continue proceeing with the operation on other sub-entities of the directory. Default the operation will terminate

quickly on encountering failures.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-DefaultProfile <Microsoft.Azure.Commands.Common.Authentication.Abstractions.Core.IAzureContextContainer>

The credentials, account, tenant, and subscription used for communication with Azure.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-FileSystem <System.String>

FileSystem name

Required? true

Position? 0

Default value None

Accept pipeline input? True (ByValue)

Accept wildcard characters? false

-MaxBatchCount <System.Int32>

Maximum number of batches that single change Access Control operation can execute. If data set size exceeds MaxBatchCount multiply BatchSize, continuation token

will be return.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-Path <System.String>

The path in the specified FileSystem that to change Acl recursively. Can be a file or directory. In the format 'directory/file.txt' or 'directory1/directory2/'.

Skip set this parameter to change Acl recursively from root directory of the Filesystem.

Required? false

Position? 1

Default value None

Accept pipeline input? True (ByValue)

Accept wildcard characters? false

-Confirm <System.Management.Automation.SwitchParameter>

Prompts you for confirmation before running the cmdlet.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-WhatIf <System.Management.Automation.SwitchParameter>

Shows what would happen if the cmdlet runs. The cmdlet is not run.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug,

ErrorAction, ErrorVariable, WarningAction, WarningVariable,

OutBuffer, PipelineVariable, and OutVariable. For more information, see

about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

INPUTS

System.String

Microsoft.Azure.Commands.Common.Authentication.Abstractions.IStorageContext

OUTPUTS

System.String

NOTES

-------- Example 1: Set ACL recursively on a directory --------

$acl = Set-AzDataLakeGen2ItemAclObject -AccessControlType user -Permission rwx

$acl = Set-AzDataLakeGen2ItemAclObject -AccessControlType group -Permission rw- -InputObject $acl

$acl = Set-AzDataLakeGen2ItemAclObject -AccessControlType other -Permission "rw-" -InputObject $acl

Set-AzDataLakeGen2AclRecursive -FileSystem "filesystem1" -Path "dir1" -Acl $acl -Context $ctx

FailedEntries :

TotalDirectoriesSuccessfulCount : 7

TotalFilesSuccessfulCount : 5

TotalFailureCount : 0

ContinuationToken :

This command first creates an ACL object with 3 acl entries, then sets ACL recursively on a directory.

Example 2: Set ACL recursively on a root directory of filesystem

$result = Set-AzDataLakeGen2AclRecursive -FileSystem "filesystem1" -Acl $acl -Context $ctx

$result

FailedEntries : {dir1/dir2/file4}

TotalDirectoriesSuccessfulCount : 500

TotalFilesSuccessfulCount : 2500

TotalFailureCount : 1

ContinuationToken :

VBaHi5TfyO2ai1wYTRhIL2FjbGNibjA2c3RmATAxRDVEN0UzRENFQzZCRTAvYWRsc3Rlc3QyATAxRDY2M0ZCQTZBN0JGQTkvZGlyMC9kaXIxL2ZpbGUzFgAAAA==

$result.FailedEntries

Name IsDirectory ErrorMessage

---- ----------- ------------

dir0/dir2/file4 False This request is not authorized to perform this operation using this permission.

user need fix the failed item , then can resume with ContinuationToken

$result = Set-AzDataLakeGen2AclRecursive -FileSystem "filesystem1" -Acl $acl -ContinuationToken $result.ContinuationToken -Context $ctx

$result

FailedEntries :

TotalDirectoriesSuccessfulCount : 100

TotalFilesSuccessfulCount : 1000

TotalFailureCount : 0

ContinuationToken :

This command first sets ACL recursively to a root directory and failed, then resume with ContinuationToken after user fix the failed file.

-------- Example 3: Set ACL recursively chunk by chunk --------

```
$token = $null
$TotalDirectoriesSuccess = 0
$TotalFilesSuccess = 0
$totalFailure = 0
$FailedEntries = New-Object System.Collections.Generic.List[System.Object]
do
{
        $result = Set-AzDataLakeGen2AclRecursive -FileSystem "filesystem1" -Path "dir1" -Acl $acl -BatchSize 100 -MaxBatchCount 2 -ContinuationToken $token -Context $ctx

    # echo $result
    $TotalFilesSuccess += $result.TotalFilesSuccessfulCount
    $TotalDirectoriesSuccess += $result.TotalDirectoriesSuccessfulCount
    $totalFailure += $result.TotalFailureCount
    $FailedEntries += $result.FailedEntries
    $token = $result.ContinuationToken
}while (($token -ne $null) -and ($result.TotalFailureCount -eq 0))
echo ""
echo "[Result Summary]"
echo "TotalDirectoriesSuccessfulCount: `t$($TotalDirectoriesSuccess)"
echo "TotalFilesSuccessfulCount: `t`t`t$($TotalFilesSuccess)"
echo "TotalFailureCount: `t`t`t`t`t$($totalFailure)"
echo "ContinuationToken: `t`t`t`t`t$($token)"
echo "FailedEntries:"$($FailedEntries | ft)
```

This script sets ACL rescursively on directory chunk by chunk, with chunk size as BatchSize * MaxBatchCount. Chunk size is 200 in this script.

Example 4: Set ACL recursively on a directory and ContinueOnFailure, then resume from failures one by one

```
$result = Set-AzDataLakeGen2AclRecursive -FileSystem "filesystem1" -Path "dir1" -Acl $acl -ContinueOnFailure -Context $ctx
```

```
$result

FailedEntries                  : {dir0/dir1/file1, dir0/dir2/file4}
TotalDirectoriesSuccessfulCount : 100
TotalFilesSuccessfulCount       : 500
TotalFailureCount              : 2
ContinuationToken              :
VBaHi5TfyO2ai1wYTRhIL2FjbGNibjA2c3RmATAxRDVEN0UzRENFQzZCRTAvYWRsc3Rlc3QyATAxRDY2M0ZCQTZBN0J
GQTkvZGlyMC9kaXIxL2ZpbGUzFgAAAA==

$result.FailedEntries

Name            IsDirectory ErrorMessage
----            ----------- ------------
dir0/dir1/file1       False This request is not authorized to perform this operation using this permission.
dir0/dir2/file4       False This request is not authorized to perform this operation using this permission.

# user need fix the failed item , then can resume with ContinuationToken

foreach ($path in $result.FailedEntries.Name)
      {
         # user code to fix failed entry in $path

         #set ACL again
         Set-AzDataLakeGen2AclRecursive -FileSystem "filesystem1" -Path $path -Acl $acl -Context $ctx
      }
```

This command first sets ACL recursively to a directory with ContinueOnFailure, and some items failed, then resume the failed items one by one.

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/az.storage/set-azdatalakegen2aclrecursive