A Script to View and Close SMB Files on Isilon Clusters

Adam Fox, Isilon Corporate Advisory Engineer [adam.fox@emc.com]

Problem Statement:

This script attempts to address an administrative issue on the Isilon Cluster. At the time of this writing, there is not a way outside of the cluster to see all of the SMB open files on the cluster. The MMC plug-in supported starting with OneFS 7.1.1 only shows the open files for the node to which the user connected. It is possible to the 'isi smb openfiles list' command on the cluster on each node using the 'isi_for_array' command, but that is not simple and either requires full administrative privileges or if the account is restricted, it requires custom changes to the sudo system.

This script will provide an alternative solution until OneFS adds a feature to address this in the OS.

Overview of smb_openfiles

Disclaimer and Support

The smb_openfiles script is written in Python but comes with a Windows executable and libraries such that it can be run on a Windows host that does not have Python installed. If Python is installed on the host, then the extra libraries included in the package are not needed. If Python is installed on the system, then only the script source (smb_openfiles.py) and the module papi.py are needed to run the script. If Python is not installed, then the entire package is needed. The code is open source and is neither officially supported by EMC nor included in any support contract. The code is offered as-is. The author will take requests for bug fixes and enhancements but there is no timeline for these requests and no guarantee that they will be completed.

Configuration

The script uses the OneFS Platform API (known as PAPI) to make a secure connection to any node on the cluster. Prior to OneFS 8.0, there is no way to know the names and IP addresses of each node on the cluster via the API. Therefore, in order to tell the script how to connect to each node, the script uses a simple configuration file to determine which IP addresses it needs to use to connect to each node of a cluster. An example of that config file is give here:

```
cluster:1:10.111.158.111
cluster:2:10.111.158.112
cluster:3:10.111.158.113
isilon:1:isilon-1.company.com
isilon:2:isilon-2.company.com
isilon:3:isilon-3.company.com
Figure 1:  Example of the configuration Script
```

In the above example, 2 clusters are defined in the same file. This is optional as the script can support different files for different environments if that is more convenient. The goal here is when the script is run to view the SMB open files, the user can decide which clusters the script should show. By default, it will connect to all of the nodes in the file, but a command line argument can be used to specify an individual cluster designated by the first colon-separated field. The second field is the node number (the script will display each node as <name>-<number> (e.g. cluster-1, cluster-2, and cluster-3). The third field is the name or IP address that the script should use to connect to the cluster.

As 8.x becomes more common, a future enhancement could be to allow the user to call out a cluster name, subnet, and pool and have the addresses discovered from that. But, currently, most clusters are on 7.x so this solution will suffice for now.

Authentication

Once the nodes have been determined, the script will prompt the user for a username and password. The password will, of course, not be echoed to the screen. The API requires credentials in order to determine if the request is allowed per system policies set up by the storage admin. Of course, the accounts 'root' or 'admin' will work. However, another user can be used as long as the user has API access and SMB privileges (ISI_PRIV_LOGIN_PAPI, and ISI_PRIV_SMB). If the goal is to only use the script to view the sessions, then only read-only access is required for ISI_PRIV_SMB. If the user is to be allowed to disconnect sessions, then the user must have read/write SMB privileges.

Modes of Operation

The script operates in one of two modes. Viewing Mode and Close Mode. Viewing mode is the default and will connect to each node and display the SMB open files on each node of the cluster.

Close Mode is where a user specifies files to be closed. This requires the user to specify the node number as well as the ID of the open file which is shown in viewing mode. Close mode is not the default and is invoked with the –c or --close flag.

Syntax

While there are several options, the goal was to make the script easy to invoke for most use cases. The syntax of the script is defined below:

```
Usage: smb_openfiles[.py]
    List Files (default mode):
        [{-C | --cluster}] : Specify a cluster from the config
file
        [{-f | --file}] : Specify an alternative config file
    Close File Mode:
        [{-c | --close}] : Close file(s) Selects Close File Mode
        [{-n | --node]} : Specifies a node # where the file is
open
        [{-i | id}] : Specify a comma-separated list of IDs
    [-h | --help] : display usage syntax
```

This is the usage output that comes if the script is invoked with the
--help flag. The options are described in more detail here.

Whether the script name needs the .py extention depends on whether it is being run
as a script or as the included Windows binary. The former requires the extension,
as that is the name of the source file. The later does not as it will have an .exe
extension, which is not required on the Windows CLI.

Viewing Mode (List Open Files Options)
With no options, the script will look a file in the same directory as the script called
'nodes.conf'. The file format was described above. If that file exists, the script will
parse that file and will connect to each node defined in that file.

If multiple clusters are defined in the same file but the intention is to only display
one cluster's nodes, the –C flag can be used to specify the name of the cluster that is
to be displayed. The first field of each line in the file defines the name of the cluster.

If multiple files are used for different use cases or if the file exists someplace other
than the same directory as the script, the –f flag is used to specify the location of the
file. There is no requirement for the file to end in .conf, any filename is allowed
when this flag is specified.

Close Mode
If the user wishes to disconnect a specific session, then disconnect mode is used.
This is invoked with either the –c or --close flag. No arguments are used with this
flag. Using this flag will then make the rest of the options in disconnect mode
required and an error will be raised if they are all not used. The order of the flags is
not significant.

The script needs to know the node that has the session that is to be disconnected.
This is specified with the –n or --node flag. The argument is a single node's name
and number as defined the conf file or after the name in viewing mode. Do not use
the SmartConnect name as that could give you any of the nodes and it's important to
find the node that actually has the connection.

The script also needs to know the ID of the open file to be closed. This is specified with the –i or --id flag. The argument to this flag is a comma-separated list of IDs as they are displayed from view mode.

If the file was successfully closed, the script will print "ID x Closed" to the screen. If there is an error, it will be displayed.

Examples

The following are examples of ways to invoke the script along with an explanation of the options selected.

- View all nodes in the nodes.conf file which resides in current directory
    - `smb_openfiles.py`
- View only the nodes from the cluster called 'isilon'
    - `smb_openfiles.py –c isilon`
- View all nodes from the config file h:\home\user1\nodes.conf
    - `smb_openfiles.py –f h:home\user1\nodes.conf`
- Close file ID 123 on node 3 of cluster5
    - `smb_openfiles.py –c –n cluster5-3 –i 123`