FileSystem.GetFileDetails

Method	**GetFileDetails**
Module	FileSystem
Library	Core
Action	Retrieves operating system attributes related to the specified file.
Parameters	`FilePath` (string): The full path of the file. `ComputeHash` (boolean; optional, default `false`): Whether to calculate the Hash for the file. ⚠ FilePath must not contain wildcards, they are assumed to be part of the FilePath.
Return values	If `FilePath` is an empty string then an `InvalidParameter` error is returned along with the following string: • `FilePath parameter should not be empty` If `FilePath` does not exist or is invalid (both the same situation depending on the platform) then a Success(no content) status. If `FilePath` itself cannot be accessed (exceptions will appear in the agent log) or no files are found then this is considered a successful execution but no results are returned. Otherwise, a single row containing the following columns will be returned: • `FilePath` (string): The full path. • `FileSystemItem` (string): The type of item the path leads to: `file/directory/symbolic link/block device/character device/fifo/socket/unknown` • `Size` (int): The size in bytes if a file, blank if a directory. • `Hash` (string): If requested and this path is a file, then this column is the SHA-256 hash of the contents of the file. A symbolic link is a link to a file, not a file. • `Owner`(string): The account that is the owner of the file system item represented in the agent's operating system specific way. See **notes** below. • `Group` (string): The group account that is considered to own the file system item represented in the agent's operating system specific way. See **notes** below. • `DateCreated` (datetime): The creation date of the file system item in Windows in ISO 8601 UTC format, or blank if non-Windows. • `DateModified` (datetime): The last modified date of the file system item in ISO 8601 UTC format. `Attributes` (string): A concatenated list of file attributes from the following, attributes mostly apply to files and directories, so 'file' can be substituted by 'directory' below. The attributes available does depend on what format the file system is, ext2 and above support attributes most systems will default to ext4 if they have it, but obviously it depends on exactly what the fs type is of the volume that is mounted under the path specified. Android and Solaris do not support retrieving attributes. For MacOSX file flags are displayed as Attributes, however extended attributes are not currently displayed.

Attribute	Windows	Linux	Mac	Meaning
read-only	yes			the file can only be read, this is independent of the security permissions
hidden	yes		yes	the file will not appear unless 'show hidden files' is enabled in explorer
system	yes			the file is considered to be a vital part of the operating system
archive	yes			the file has changed since the last system backup (which would clear this flag next time it is backed up)
device	yes			the file is actually a conduit for a device
temporary	yes			the file is marked as temporary, typically deleted once the file has been closed
sparse	yes			the file is masquerading as a massive file with large areas of nothing
compressed	yes	yes		the file is compressed on disk
offline	yes			the file content is unavailable right now, it is stored somewhere else
indexed	yes			the file content will be used in Windows Search results
encrypted	yes			the file content is encrypted
virtual	yes			the file exists in a virtual space rather than physically occupying disk space
immutable		yes	yes	the file cannot be altered
touch access time		yes		update the access time when the file is accessed
append only		yes		the file can only be opened for reading or appending to the end
copy on write		yes		the original file, when written to, is copied elsewhere then the write is on that copy, preserving the original
sync directory writes		yes		force writes of directories to disk surface
nodump		yes	yes	skip the file when 'dump' is used to back up the file
compression error		yes		used by the experimental compression patches to indicate that a compressed file has a compression error
extents		yes		the file storage mechanism uses extents rather than listing every block individually, this also indicates less fragmentation of the file
huge		yes		file storage units are in blocks rather than sectors
htree indexed		yes		the htree program should index this file
journaled		yes		file writes are written to the journal first, then to the file itself
delete securely		yes		the file is deleted first by overwriting it on disk with zeros then deleting it
sync updates		yes		any writes to the file are written immediately to the disk surface, not cached
top directory		yes		indicates that subdirectories with this directory are scattered around different areas of the disk
no tail merging		yes		see https://en.wikipedia.org/wiki/Block_suballocation
undeletable		yes		the item cannot be deleted regardless of the other permissions available
raw access to compression		yes		indicates that a raw contents of a compressed file can be accessed directly by the experimental compression patches
dirty compression		yes		indicates that a raw contents of a compressed file are 'dirty'
arch			yes	the file has been archived (opposite of windows archive)
opaque			yes	make a file opaque, for instance a directory 'file'
`sappnd`			yes	system append only file
schg			yes	make the system file unchangable (immutable)
uappnd			yes	user append only file
uchg			yes	make the user file unchangable (immutable)

Mac Specific

On macOS from version 10.4 it is possible, through the **xattr** and **ls -l@** commands to add, edit, delete and display arbitrary extended attributes on a file system item. These will be listed, if present on the file, after any standard attributes above if bestowed on the file.

Extended attributes used by Apple include these commonly found :

- com.apple.FinderInfo
- com.apple.LaunchServices.OpenWith
- com.apple.ResourceFork
- com.apple.TextEncoding
- com.apple.genstore.info
- com.apple.genstore.orig_perms_v1
- com.apple.genstore.origdisplayname
- com.apple.genstore.origposixname
- com.apple.metadata:com_apple_backup_excludeItem
- com.apple.quarantine

Example

```
FileSystem.GetFileDetails(FilePath:"c:\path\file.txt", ComputeHash:true);
```

Platforms

- Windows
- Linux
- MacOS
- ~~Solaris Intel~~
- ~~Solaris Spare~~
- Android

Notes	Calculating the hash can be expensive so it is optional. Some result columns may not apply to the platform that the agent is running on, for instance on Unix it is not possible to determine when a file system item was created. In such circumstances the row will contain empty cells for those columns that are not applicable to the platform. This method has been restricted to only search the local computer for the translation of SIDs into "domain\user". If the Owner/Group SID identifies a user in a domain (i.e. not local to the device), then the SID will be stringized into the "S-1-5-21-xxx-yyy" format. The reason for this local lookup only is that the Agents involved will all be running the instruction at roughly the same time and thus this could be interpreted as a denial of service attack on the domain controllers.