FileSystem.GetFilePermissions

Method	**GetFilePermissions`**
Module	FileSystem
Library	Core
Action	Returns the set of permissions for a specified file or directory.
Parameters	`FilePath` (string) : The full path of the file or directory.
Return values	Each permission is returned as a row of these values: `UserOrRole` (string): A user-friendly representation of a username or role. So for example username `BUILTIN\ADMINISTRATORS` is the user-friendly representation of Security Id S-1-5-32-544. A Linux role is of the form `Owner`, `Group` or `Others`. `Inheritance` (string): The inheritance properties for this permission, for example `Inherited`, `None` or "`ForInheritanceOnly,FoldersInherit,FilesInherit`". Currently populated for Windows and macOS. `AccessClass` (string): An access class for this permission, typically `Allow`, `Deny`, `Audit` or `Alarm`. `AccessProperties` (string): The access properties for this permission, for example "`Read,Write,Execute`", `FullControl` or `Special`. The individual access property values are supported as follows:

Access Property	Windows	Linux	Mac	Solaris	Android	Meaning if access allowed
`AddFile`			yes			(ACL) For a directory, add a file
`AddSubDirectory`			yes			(ACL) For a directory, add a subdirectory
`Append`			yes			(ACL) Append to a file
`AppendData`	yes					Append to a file
`ChangePerms`	yes					Modify the DACL in the object security descriptor for the file or directory
`Chown`			yes			(ACL) Change owner
`CreateFiles`	yes					For a directory, create files within
`CreateFolders`	yes					For a directory, create subdirectories
`Delete`	yes		yes			(ACL) For a directory, delete an existing file or subdirectory
`DeleteChild`	yes		yes			Windows: For a directory, delete it and all the files it contains (its children), even if the files are read-only. Non-Windows: (ACL) For a directory, delete a file or subdirectory.
`Execute`	yes	yes	yes	yes	yes	The file can be executed or the directory can be traversed and searched
`FullControl`	yes					Every right that Modify has, plus Change Permission and Take Ownership rights
`List`			yes			(ACL) The contents of the directory can be listed
`ListFolder`	yes					For a directory, list the contents
`Modify`	yes					Create, delete, change, and move files within their directory
`Read`		yes	yes	yes	yes	The file/directory can be read
`ReadAttr`	yes		yes			(ACL) Read basic (non-ACL) attributes
`ReadData`	yes					The file can be read
`ReadExtAttr`	yes		yes			(ACL) Read extended (i.e.named) attributes
`ReadPerms`	yes					Read the security descriptor for the file or directory, excluding the information in the SACL
`ReadSecurity`			yes			(ACL) Read ACL attributes
`SetGIDOnExe`		yes	yes	yes	yes	If an executable file, run it with the group ID of the group
`SetUIDOnExe`		yes	yes	yes	yes	If an executable file, run it with the user ID of the owner
`Special`	yes					Some other access property
`Sticky`		yes	yes	yes	yes	For a directory, only the owner and root can rename or delete objects within it
`Synchronize`			yes			(ACL) Access the file locally at the server with synchronized reads and writes
`TakeOwnership`	yes					Change the owner
`TraverseFolder`	yes					For a directory, traverse it
`Write`		yes	yes	yes	yes	The file can be written to or the directory can have objects created within it
`WriteAttr`	yes		yes			(ACL) Write basic (non-ACL) attributes
`WriteData`	yes					The file can be written to
`WriteExtAttr`	yes		yes			(ACL) Write extended (i.e. named) attributes
`WriteSecurity`			yes			(ACL) Write ACL attributes

Example

```
FileSystem.GetFilePermissions(FilePath:"C:\\Test");
```

Platforms

- Windows
- Linux
- MacOS
- Solaris Intel
- Solaris Sparc
- Android

Notes	Tachyon uses the same terms as Windows Explorer such as `TakeOwnership` except that it presents these attributes as single words in a comma-separated list. In addition Windows Explorer uses the word "`Special`" to report complex permissions such as readExtendedAttributes, readAttributes, readPermissions, etc. Tachyon accurately reports every attribute, because "`Special`" says only that standard permissions are not applied to the file. Tachyon follows the Windows Explorer example and distinguishes between `Modify` and `FullControl`. `FullControl` containing every right that `Modify` does plus `ChangePerms` and `TakeOwnership` rights. With modify rights a user can create, delete, change, and move files within their directories. But they cannot change the permissions or change the owner of these files. This ensures that permissions set on these files, by an administrator, will remain uniform. GetFilePermissions 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 on a domain that is not the local machine, 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. For non-Windows endpoints, file mode permissions such as `drwsrwsrwt` are presented as minimal ACLs with the three roles `Owner`, `Group` and `Other` being presented. In additional if any special modes are set for the file or directory then a Special role is also included which can include the set GID on execution, set UID on execution and sticky bit. For more information on these bits refer to http://permissions-calculator.org/info/ as a good explanation. ACL support was added in the Linux 2.6 kernel and is available on a filesystem basis (ext2, ext3, ext, IBM JFS, XFS, reiserfs). Linux distributions typically offer ACL as an optional install feature, for example 'yum install acl' on Centos/Red Hat. Also the file system will need to be mounted with the acl option. Linux ACL support is POSIX 1003.1e DS 17 compliant. This solution only offers read, write, execute permissions on a per user or per group basis, plus directory ACL attribute inheritance. The Linux solution is inferior to either MacOSX ACLs or Windows ACLs. Typically ACLS are set using **setf acl** and displayed using **getfacl**. ACL support is enabled by default for MacOSX Leopard onwards and the support is much richer than the Linux ACL solution, it is quite similar to Windows ACL solution. Typically ACLs are set using **ch mod** and displayed using **ls -le**. A good description of MacOSX ACL support is here. ACL support for Windows is rich and comparable to MacOSX ACL support. A good description is here. A good table of operating system/file system support for ACLs is here.