Storage System API

> Requires: *phpFox version >= **4.8.0***

From phpFox version 4.8.0, we have supported new Storage System APIs. It's helpful for 3rd-party apps to manage and manipulate files in storage systems.

Here are supported storage systems in phpFox:

1. Local filesystem
2. Amazon Simple Storage System (S3)
3. Digital Ocean Space
4. FTP storage file system via FTP connect
5. SFTP server via SSH connect
6. S3 Compatible storage system (see List of S3-compatible storage providers)

Below are some sample codes to manipulate files with storage systems on phpFox site:

Retrieve a Storage Manager object

File path */PF.Src/Core/Storage/StorageManager.php*

```
<?php

// Get storage object associate with storageId = 1.
$storage = Phpfox::getLib('storage')->getStorage(1);

// Get default storage
$storage = Phpfox::getLib('storage')->getStorage();

// Get default storage Id
$defaultStorageId = Phpfox::getLib('storage')->getStorageId();
```

Get the public URL of a file

```
<?php
//get default Storage system
$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';

// @param string $path     The path to the file.
// @param string $contents The file contents.
// @param array  $config   An optional configuration array.
// @return string full url of file

$fileUrl = $storage->getUrl($path);

?>
```

Create a file or update if exists

```php
<?php

$path =  'PF.Base/file/new-file-api.txt';
$content = 'file api content';
$config = ['visibility'=>'public'];
$storage = Phpfox::getLib('storage')->getStorage();

// @param string $path     The path to the file.
// @param string $contents The file contents.
// @param array  $config   An optional configuration array.
// @return bool True on success, false on failure.

$isSuccessful = $storage->put($path, $contents, $config);

?>
```

Check whether a file exists

```php
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';

// @param string $path
// @return bool
$isExisted = $storage->has($path);

?>
```

Read a file contents

```php
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';

// @param string $path The path to the file.
// @throws FileNotFoundException
// @return string|false The file contents or false on failure.
$fileContent = $storage->read($path);

?>
```

Retrieves a read-stream for a path

```php
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';

// Retrieves a read-stream for a path.
// @param string $path The path to the file.
// @throws FileNotFoundException
// @return resource|false The path resource or false on failure.
$fileStream = $storage->readStream($path);
?>
```

List contents of a directory

```
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$directory =  'PF.Base/file';
$recursive =  false;

// @param string $directory The directory to list.
// @param bool   $recursive Whether to list recursively.
// @return array A list of file metadata.

$fileMetaList = $storage->listContents($directory, $recursive);
?>
```

Get a file's metadata

```
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';
// @param string $path The path to the file.
// @throws FileNotFoundException
// @return array|false The file metadata or false on failure.

$fileInfo = $storage->getMetadata($path);
?>
```

Get a file's size

```
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';

// @param string $path The path to the file.
// @throws FileNotFoundException
// @return int|false The file size or false on failure.

$fileSize = $storage->getSize($path);
?>
```

Get file's mime type

```
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';

// @param string $path The path to the file.
// @throws FileNotFoundException
// @return string|false The file mime-type or false on failure.

$fileMimeType = $storage->getMimetype($path);
?>
```

Get file's timestamp

```
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';

// @param string $path The path to the file.
// @throws FileNotFoundException
// @return int|false The timestamp or false on failure.

$lastUpdate = $storage->getTimestamp($path);
?>
```

Get a file's visibility

```
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';

// @param string $path The path to the file.
// @throws FileNotFoundException
// @return string|false The visibility (public|private) or false on failure.

$fileVisibility = $storage->getVisibility($path);
?>
```

Write a new file

```
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';
$contents  = 'file api content';
$config = ['visibility'=>'public'];

// @param string $path     The path of the new file.
// @param string $contents The file contents.
// @param array  $config   An optional configuration array.
// @throws FileExistsException
// @return bool True on success, false on failure.
$isSuccessful = $storage->write($path, $contents, $config);
?>
```

Write a new file using a stream

```php
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';
$resource  = fopen('local/file','r');
$config = ['visibility'=>'public'];

// @param string   $path     The path of the new file.
// @param resource $resource The file handle.
// @param array    $config   An optional configuration array.
// @throws InvalidArgumentException If $resource is not a file handle.
// @throws FileExistsException
// @return bool True on success, false on failure.

$isSuccessful = $storage->writeStream($path, $resource, $config);
?>
```

Rename a file

```php
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';
$newpath =  'PF.Base/file/new-file-api-renamed.txt';

// @param string $path    Path to the existing file.
// @param string $newpath The new path of the file.
// @throws FileExistsException   Thrown if $newpath exists.
// @throws FileNotFoundException Thrown if $path does not exist.
// @return bool True on success, false on failure.

$isSuccessful = $storage->rename($path, $newpath);
?>
```

Copy a file

```php
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';
$newpath =  'PF.Base/file/new-file-api-renamed.txt';

// @param string $path    Path to the existing file.
// @param string $newpath The new path of the file.
// @throws FileExistsException   Thrown if $newpath exists.
// @throws FileNotFoundException Thrown if $path does not exist.
// @return bool True on success, false on failure.

$isSuccessful = $storage->copy($path, $newpath);
?>
```

Delete a file

```php
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';

// @param string $path
// @throws FileNotFoundException
// @return bool True on success, false on failure.

$isSuccessful = $storage->delete($path);
?>
```

Delete a directory

```php
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$dirname =  'PF.Base/file/sub-dir/';

// @param string $dirname
// @throws RootViolationException Thrown if $dirname is empty.
// @return bool True on success, false on failure.

$isSuccessful = $storage->deleteDir($dirname);

?>
```

Create a directory

```php
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$dirname =  'PF.Base/file/sub-dir/';
$config= ['visibility'=>'public'];

// @param string $dirname The name of the new directory.
// @param array  $config  An optional configuration array.
*
// @return bool True on success, false on failure.

$isSuccessful = $storage->createDir($dirname, $config);
?>
```

Set file visibility

```php
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';
$visibility = 'public';

// @param string $path       The path to the file.
// @param string $visibility One of 'public' or 'private'.
// @throws FileNotFoundException
// @return bool True on success, false on failure.

$isSuccessful = $storage->setVisibility($path, $visibility);

?>
```

Read and delete a file

```
<?php

$storage = Phpfox::getLib('storage')->getStorage();
$path =  'PF.Base/file/new-file-api.txt';

// @param string $path The path to the file.
// @throws FileNotFoundException
// @return string|false The file contents, or false on failure.

$fileContent = $storage->readAndDelete($path);
?>
```