**File Operations in JavaScript FileManager control**

02 Dec 2022 / 15 minutes to read

The file manager component is used to browse, manage, and organize the files and folders in a file system through a web application. All basic file operations like creating a new folder, uploading and downloading of files in the file system, and deleting and renaming of existing files and folders are available in the file manager component. Additionally, previewing of image files is also provided in the file manager component.

The following table represents the basic operations available in the file manager and their corresponding functions.

| **Operation Name** | **Function** |
| --- | --- |
| read | Read the details of files or folders available in the given path from the file system, to display the files for the user to browse the content. |
| create | Creates a new folder in the current path of the file system. |
| delete | Removes the file or folder from the file server. |
| rename | Rename the selected file or folder in the file system. |
| search | Searches for items matching the search string in the current and child directories. |
| details | Gets the detail of the selected item(s) from the file server. |
| copy | Copy the selected file or folder in the file system. |
| move | Cut the selected file or folder in the file server. |
| upload | Upload files to the current path or directory in the file system. |
| download | Downloads the file from the server and the multiple files can be downloaded as ZIP files. |

The *CreateFolder*, *Remove*, and *Rename* actions will be reflected in the file manager only after the successful response from the server.

**Folder Upload support**

To perform the directory(folder) upload in File Manager, set [directoryUpload](https://ej2.syncfusion.com/documentation/api/file-manager/uploadSettings/#directoryupload) as true within the uploadSettings property. The directory upload feature is supported for the following file service providers:

* Physical file service provider.
* Azure file service provider.
* NodeJS file service provider.
* Amazon file service provider.

In the following example, directory upload is enabled/disabled on DropDownButton selection.

**Source**

Preview

**app.ts**

index.html

import { FileManager, Toolbar, NavigationPane, DetailsView, ContextMenu } from '@syncfusion/ej2-filemanager';

import { DropDownButton, ItemModel } from '@syncfusion/ej2-splitbuttons';

FileManager.Inject(Toolbar, NavigationPane, DetailsView, ContextMenu);

/\*\*

\* File Manager folder upload sample

\*/

let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';

let fileObject: FileManager = new FileManager({

ajaxSettings: {

url: hostUrl + 'api/FileManager/FileOperations',

getImageUrl: hostUrl + 'api/FileManager/GetImage',

uploadUrl: hostUrl + 'api/FileManager/Upload',

downloadUrl: hostUrl + 'api/FileManager/Download'

}

});

fileObject.appendTo('#file');

let items: ItemModel[] = [{ text: 'Folder' }, { text: 'Files' }];

let drpDownBtn: DropDownButton = new DropDownButton({

items: items,

select: (args) => {

if (args.item.text === 'Folder') {

fileObject.uploadSettings.directoryUpload = true;

} else {

fileObject.uploadSettings.directoryUpload = false;

}

setTimeout(function () {

let uploadBtn: HTMLElement = document.querySelector('.e-file-select-wrap button');

uploadBtn.click();

}, 100);

}

},'#file\_tb\_upload');

document.getElementById('file\_tb\_upload').onclick = function (args) {

args.stopPropagation();

};

**Physical file service provider**

To achieve the directory upload in the physical file service provider, use the below code snippet in IActionResult Upload method in the Controllers/FileManagerController.cs file.

[Route("Upload")]

public IActionResult Upload(string path, IList<IFormFile> uploadFiles, string action)

{

FileManagerResponse uploadResponse;

foreach (var file in uploadFiles)

{

var folders = (file.FileName).Split('/');

// checking the folder upload

if (folders.Length > 1)

{

for (var i = 0; i < folders.Length - 1; i++)

{

string newDirectoryPath = Path.Combine(this.basePath + path, folders[i]);

if (!Directory.Exists(newDirectoryPath))

{

this.operation.ToCamelCase(this.operation.Create(path, folders[i]));

}

path += folders[i] + "/";

}

}

}

uploadResponse = operation.Upload(path, uploadFiles, action, null);

if (uploadResponse.Error != null)

{

Response.Clear();

Response.ContentType = "application/json; charset=utf-8";

Response.StatusCode = Convert.ToInt32(uploadResponse.Error.Code);

Response.HttpContext.Features.Get<IHttpResponseFeature>().ReasonPhrase = uploadResponse.Error.Message;

}

return Content("");

}

Refer to the [GitHub](https://github.com/SyncfusionExamples/ej2-aspcore-file-provider/blob/master/Controllers/FileManagerController.cs#L76) for more details

And also add the below code snippet in FileManagerResponse Upload method in Models/PhysicalFileProvider.cs file.

string[] folders = name.Split('/');

string fileName = folders[folders.Length - 1];

var fullName = Path.Combine((this.contentRootPath + path), fileName);

Refer to the [GitHub](https://github.com/SyncfusionExamples/ej2-aspcore-file-provider/blob/master/Models/PhysicalFileProvider.cs#L1185) for more details.

**Azure file service provider**

For Azure file service provider, no customizations are needed for directory upload with server side and this will work with the below default upload method code.

Refer to the [GitHub](https://github.com/essential-studio/azure-aspcore-file-provider/blob/master/Controllers/AzureProviderController.cs#L89) for more details.

**NodeJS file service provider**

To perform the directory upload in the NodeJS file service provider, use the below code snippet in app.post method in the filesystem-server.js file.

var folders = (req.body.filename).split('/');

var filepath = req.body.path;

var uploadedFileName = folders[folders.length - 1];

// checking the folder upload

if (folders.length > 1)

{

for (var i = 0; i < folders.length - 1; i++)

{

var newDirectoryPath = path.join(contentRootPath + filepath, folders[i]);

if (!fs.existsSync(newDirectoryPath)) {

fs.mkdirSync(newDirectoryPath);

(async () => {

await FileManagerDirectoryContent(req, res, newDirectoryPath).then(data => {

response = { files: data };

response = JSON.stringify(response);

});

})();

}

filepath += folders[i] + "/";

}

fs.rename('./' + uploadedFileName, path.join(contentRootPath, filepath + uploadedFileName), function (err) {

if (err) {

if (err.code != 'EBUSY') {

errorValue.message = err.message;

errorValue.code = err.code;

}

}

});

}

Refer to the [GitHub](https://github.com/SyncfusionExamples/ej2-filemanager-node-filesystem/blob/master/filesystem-server.js#L788) for more details.

**Amazon file service provider**

To perform the directory upload in the Amazon file service provider, use the below code snippet in IActionResult AmazonS3Upload method in the Controllers/AmazonS3ProviderController.cs file.

foreach (var file in uploadFiles)

{

var folders = (file.FileName).Split('/');

// checking the folder upload

if (folders.Length > 1)

{

for (var i = 0; i < folders.Length - 1; i++)

{

if (!this.operation.checkFileExist(path, folders[i]))

{

this.operation.ToCamelCase(this.operation.Create(path, folders[i], dataObject));

}

path += folders[i] + "/";

}

}

}

Refer to the [GitHub](https://github.com/essential-studio/amazon-s3-aspcore-file-provider/blob/development/Controllers/AmazonS3ProviderController.cs#L83) for more details.

And also add the below code snippet in AsyncUpload method in Models/AmazonS3FileProvider.cs file.

string[] folders = file.FileName.Split('/');

string name = folders[folders.Length - 1];

Refer to the [GitHub](https://github.com/essential-studio/amazon-s3-aspcore-file-provider/blob/development/Models/AmazonS3FileProvider.cs#L585) for more details.

**File operation request and response Parameters**

The default parameters available in file operation request from the file manager and the corresponding response parameters required by the file manager are listed as follows.

**Read**

The following table represents the request parameters of *read* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| action | String | read | Name of the file operation. |
| path | String | - | Relative path from which the data has to be read. |
| showHiddenItems | Boolean | - | Defines show or hide the hidden items. |
| data | FileManagerDirectoryContent | - | Details about the current path (directory). |

*Refer*[*File request and response contents*](https://ej2.syncfusion.com/documentation/file-manager/file-operations/#file-request-and-response-contents)*for the contents of data*.

*Example:*

{

action: "read",

path: "/",

showHiddenItems: false,

data: []

}

The following table represents the response parameters of *read* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| cwd | FileManagerDirectoryContent | - | Path (Current Working Directory) details. |
| files | FileManagerDirectoryContent[] | - | Details of files and folders present in given path or directory. |
| error | ErrorDetails | - | Error Details |

*Refer*[*File request and response contents*](https://ej2.syncfusion.com/documentation/file-manager/file-operations/#file-request-and-response-contents)*for the contents of cwd, files, and error*.

*Example:*

{

cwd:

{

name:"Download",

size:0,

dateModified:"2019-02-28T03:48:19.8319708+00:00",

dateCreated:"2019-02-27T17:36:15.812193+00:00",

hasChild:false,

isFile:false,

type:"",

filterPath:"//Download//"

},

files:[

{

name:"Sample Work Sheet.xlsx",

size:6172,

dateModified:"2019-02-27T17:23:50.9651206+00:00",

dateCreated:"2019-02-27T17:36:15.8151955+00:00",

hasChild:false,

isFile:true,

type:".xlsx",

filterPath:"//Download//"

}

],

error:null,

details:null

}

**Create**

The following table represents the request parameters of *create* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| action | String | create | Name of the file operation. |
| path | String | - | Relative path in which the folder has to be created. |
| name | String | - | Name of the folder to be created. |
| data | FileManagerDirectoryContent | - | Details about the current path (directory). |

*Refer*[*File request and response contents*](https://ej2.syncfusion.com/documentation/file-manager/file-operations/#file-request-and-response-contents)*for the contents of data*

*Example:*

{

action: "create",

data: [

{

dateCreated: "2019-02-27T17:36:15.6571949+00:00",

dateModified: "2019-03-12T10:17:31.8505975+00:00",

filterPath: "/",

hasChild: true,

isFile: false,

name: files,

nodeId: "fe\_tree",

size: 0,

type: ""

}

],

name: "Hello",

path: "/"

}

The following table represents the response parameters of *create* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| files | FileManagerDirectoryContent[] | - | Details of the created folder |
| error | ErrorDetails | - | Error Details |

*Refer*[*File request and response contents*](https://ej2.syncfusion.com/documentation/file-manager/file-operations/#file-request-and-response-contents)*for the contents of files and error*.

*Example:*

{

cwd: null,

files: [

{

dateCreated: "2019-03-15T10:25:05.3596171+00:00",

dateModified: "2019-03-15T10:25:05.3596171+00:00",

filterPath: null,

hasChild: false,

isFile: false,

name: "New",

size: 0,

type: ""

}

],

details: null,

error: null

}

**Rename**

The following table represents the request parameters of *rename* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| action | String | rename | Name of the file operation. |
| path | String | - | Relative path in which the item is located. |
| name | String | - | Current name of the item to be renamed. |
| newname | String | - | New name for the item. |
| data | FileManagerDirectoryContent | - | Details of the item to be renamed. |

*Refer*[*File request and response contents*](https://ej2.syncfusion.com/documentation/file-manager/file-operations/#file-request-and-response-contents)*for the contents of data*.

*Example:*

{

action: "rename",

data: [

{

dateCreated: "2019-03-20T05:22:34.621Z",

dateModified: "2019-03-20T08:45:56.000Z",

filterPath: "/Pictures/Nature/",

hasChild: false,

iconClass: "e-fe-image",

isFile: true,

name: "seaviews.jpg",

size: 95866,

type: ".jpg"

}

],

newname: "seaview.jpg",

name: "seaviews.jpg",

path: "/Pictures/Nature/"

}

The following table represents the response parameters of *rename* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| files | FileManagerDirectoryContent[] | - | Details of the renamed item. |
| error | ErrorDetails | - | Error Details |

*Refer*[*File request and response contents*](https://ej2.syncfusion.com/documentation/file-manager/file-operations/#file-request-and-response-contents)*for the contents of files and error*.

*Example:*

{

cwd:null,

files:[

{

name:"seaview.jpg",

size:95866,

dateModified:"2019-03-20T08:45:56+00:00",

dateCreated:"2019-03-20T05:22:34.6214847+00:00",

hasChild:false,

isFile:true,

type:".jpg",

filterPath:"//Pictures//Nature//seaview.jpg"

}

],

error:null,

details:null

}

**Delete**

The following table represents the request parameters of *delete* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| action | String | delete | Name of the file operation. |
| path | String | - | Relative path where the items to be deleted are located. |
| names | String[] | - | List of the items to be deleted. |
| data | FileManagerDirectoryContent | - | Details of the item to be deleted. |

*Refer*[*File request and response contents*](https://ej2.syncfusion.com/documentation/file-manager/file-operations/#file-request-and-response-contents)*for the contents of data*.

*Example:*

{

action: "delete",

path: "/Hello/",

names: ["New"],

data: []

}

The following table represents the response parameters of *delete* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| files | FileManagerDirectoryContent[] | - | Details about the deleted item(s). |
| error | ErrorDetails | - | Error Details |

*Refer*[*File request and response contents*](https://ej2.syncfusion.com/documentation/file-manager/file-operations/#file-request-and-response-contents)*for the contents of files and error*.

*Example:*

{

cwd: null,

details: null,

error: null,

files: [

{

dateCreated: "2019-03-15T10:13:30.346309+00:00",

dateModified: "2019-03-15T10:13:30.346309+00:00",

filterPath: "/Hello/folder",

hasChild: true,

isFile: false,

name: "folder",

size: 0,

type: ""

}

]

}

**Details**

The following table represents the request parameters of *details* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| action | String | details | Name of the file operation. |
| path | String | - | Relative path where the items are located. |
| names | String[] | - | List of the items to get details. |
| data | FileManagerDirectoryContent | - | Details of the selected item. |

*Refer*[*File request and response contents*](https://ej2.syncfusion.com/documentation/file-manager/file-operations/#file-request-and-response-contents)*for the contents of data*.

*Example:*

{

action: "details",

path: "/FileContents/",

names: ["All Files"],

data: []

}

The following table represents the response parameters of *details* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| details | FileManagerDirectoryContent | - | Details of the requested item(s). |
| error | ErrorDetails | - | Error Details |

*Refer*[*File request and response contents*](https://ej2.syncfusion.com/documentation/file-manager/file-operations/#file-request-and-response-contents)*for the contents of details and error*.

*Example:*

{

cwd:null,

files:null,

error:null,

details:

{

name:"All Files",

location:"//Files//FileContents//All Files",

isFile:false,

size:"679.8 KB",

created:"3/8/2019 10:18:37 AM",

modified:"3/8/2019 10:18:39 AM",

multipleFiles:false

}

}

**Search**

The following table represents the request parameters of *search* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| action | String | search | Name of the file operation. |
| path | String | - | Relative path to the directory where the files should be searched. |
| showHiddenItems | Boolean | - | Defines show or hide the hidden items. |
| caseSensitive | Boolean | - | Defines search is case sensitive or not. |
| searchString | String | - | String to be searched in the directory. |
| data | FileManagerDirectoryContent | - | Details of the searched item. |

*Example:*

{

action: "search",

path: "/",

searchString: "\*nature\*",

showHiddenItems: false,

caseSensitive: false,

data: []

}

The following table represents the response parameters of *search* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| cwd | FileManagerDirectoryContent | - | Path (Current Working Directory) details. |
| files | FileManagerDirectoryContent[] | - | Files and folders in the searched directory that matches the search input. |
| error | ErrorDetails | - | Error Details |

*Refer*[*File request and response contents*](https://ej2.syncfusion.com/documentation/file-manager/file-operations/#file-request-and-response-contents)*for the contents of cwd, files and error.*

*Example:*

{

cwd:

{

name:files,

size:0,

dateModified:"2019-03-15T10:07:00.8658158+00:00",

dateCreated:"2019-02-27T17:36:15.6571949+00:00",

hasChild:true,

isFile:false,

type:"",

filterPath:"//"

},

files:[

{

name:"Nature",

size:0,

dateModified:"2019-03-08T10:18:42.9937708+00:00",

dateCreated:"2019-03-08T10:18:42.5907729+00:00",

hasChild:true,

isFile:false,

type:"",

filterPath:"//FileContents//Nature"

}

],

error:null,

details:null

}

**Copy**

The following table represents the request parameters of *copy* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| action | String | copy | Name of the file operation. |
| path | String | - | Relative path to the directory where the files should be copied. |
| names | String[] | - | List of files to be copied. |
| targetPath | String | - | Relative path where the items to be pasted are located. |
| data | FileManagerDirectoryContent | - | Details of the copied item. |
| renameFiles | String[] | - | Details of the renamed item. |

*Example:*

{

action: "copy",

path: "/",

names: ["6.png"],

renameFiles: ["6.png"],

targetPath: "/Videos/"

}

The following table represents the response parameters of *copy* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| cwd | FileManagerDirectoryContent | - | Path (Current Working Directory) details. |
| files | FileManagerDirectoryContent[] | - | Details of copied files or folders |
| error | ErrorDetails | - | Error Details |

*Refer*[*File request and response contents*](https://ej2.syncfusion.com/documentation/file-manager/file-operations/#file-request-and-response-contents)*for the contents of cwd, files and error.*

*Example:*

{

cwd:null,

files:[

{

path:null,

action:null,

newName:null,

names:null,

name:"justin.mp4",

size:0,

previousName:"album.mp4",

dateModified:"2019-06-21T06:58:32+00:00",

dateCreated:"2019-06-24T04:22:14.6245618+00:00",

hasChild:false,

isFile:true,

type:".mp4",

id:null,

filterPath:"//"

}

],

error:null,

details:null

}

**Move**

The following table represents the request parameters of *move* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| action | String | move | Name of the file operation. |
| path | String | - | Relative path to the directory where the files should be copied. |
| names | String[] | - | List of files to be moved. |
| targetPath | String | - | Relative path where the items to be pasted are located. |
| data | FileManagerDirectoryContent | - | Details of the moved item. |
| renameFiles | String[] | - | Details of the renamed item. |

*Example:*

{

action: "move",

path: "/",

names: ["6.png"],

renameFiles: ["6.png"],

targetPath: "/Videos/"

}

The following table represents the response parameters of *copy* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| cwd | FileManagerDirectoryContent | - | Path (Current Working Directory) details. |
| files | FileManagerDirectoryContent[] | - | Details of cut files or folders |
| error | ErrorDetails | - | Error Details |

*Refer*[*File request and response contents*](https://ej2.syncfusion.com/documentation/file-manager/file-operations/#file-request-and-response-contents)*for the contents of cwd, files and error.*

*Example:*

{

cwd:null,

files:[

{

path:null,

action:null,

newName:null,

names:null,

name:"justin biber.mp4",

size:0,

previousName:"justin biber.mp4",

dateModified:"2019-06-21T06:58:32+00:00",

dateCreated:"2019-06-24T04:26:49.2690476+00:00",

hasChild:false,

isFile:true,

type:".mp4",

id:null,

filterPath:"//Videos//"

}

],

error:null,

details:null

}

**Upload**

The following table represents the request parameters of *Upload* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| action | String | Save | Name of the file operation. |
| path | String | - | Relative path to the location where the file has to be uploaded. |
| uploadFiles | IList<IFormFile> | - | File that are uploaded. |

*Example:*

uploadFiles: (binary),

path: /,

action: Save,

data: {

path:null,

action:null,

newName:null,

names:null,

name:"Downloads",

size:0,

previousName:null,

dateModified:"2019-07-22T11:23:46.7153977 00:00",

dateCreated:"2019-07-22T11:26:13.9047229 00:00",

hasChild:false,

isFile:false,

type:"",

id:null,

filterPath:"//",

targetPath:null,

renameFiles:null,

uploadFiles:null,

caseSensitive:false,

searchString:null,

showHiddenItems:false,

\_fm\_iconClass:null,

\_fm\_id:"fe\_tree\_1",

\_fm\_pId:null,

\_fm\_selected:false,

\_fm\_icon:null,

data:null,

targetData:null,

permission:null

}

The upload response is an empty string.

**Download**

The following table represents the request parameters of *download* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| action | String | download | Name of the file operation |
| path | String | - | Relative path to location where the files to download are present. |
| names | String[] | - | Name list of the items to be downloaded. |
| data | FileManagerDirectoryContent | - | Details of the download item. |

*Example:*

{

action:"download",

path:"/",

names:["1.png"],

data:[

{

path:null,

action:null,

newName:null,

names:null,

name:"1.png",

size:49792,

previousName:null,

dateModified:"2019-07-22T12:15:45.0972405+00:00",

dateCreated:"2019-07-22T12:15:45.0816042+00:00",

hasChild:false,

isFile:true,

type:".png",

id:null,

filterPath:"//",

targetPath:null,

renameFiles:null,

uploadFiles:null,

caseSensitive:false,

searchString:null,

showHiddenItems:false,

\_fm\_iconClass:"e-fe-image",

\_fm\_id:null,

\_fm\_pId:null,

\_fm\_selected:false,

\_fm\_icon:null,

data:null,

targetData:null,

permission:null,

\_fm\_created:"2019-07-22T12:15:45.081Z",

\_fm\_modified:"2019-07-22T12:15:45.097Z",

\_fm\_imageUrl:"https://ej2-aspcore-service.azurewebsites.net/api/FileManager/GetImage?path=/1.png",

\_fm\_imageAttr:

{

alt:"1.png"

},

\_fm\_htmlAttr:

{

class:"e-large-icon",

title:"1.png"

}

}

]

}

Downloads the requested items from the file server in response.

**GetImage**

The following table represents the request parameters of *GetImage* operations.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| path | String | - | Relative path to the image file |

Return the image as a file stream in response.

The request from the file manager can be customized using the beforeSend event. Additional information can be passed to the file manager in file operation response and can be used in customization.

**File request and response contents**

The following table represents the contents of *data, cwd, and files* in the file manager request and response.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| name | String | - | File name |
| dateCreated | String | - | Date in which file was created (UTC Date string). |
| dateModified | String | - | Date in which file was last modified (UTC Date string). |
| filterPath | String | - | Relative path to the file or folder. |
| hasChild | Boolean | - | Defines this folder has any child folder or not. |
| isFile | Boolean | - | Say whether the item is file or folder. |
| size | Number | - | File size |
| type | String | - | File extension |

The following table represents the contents of *error* in the file manager request and response.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| code | String | - | Error code |
| message | String | - | Error message |
| fileExists | String[] | - | List of duplicate file names |

The following table represents the contents of *details* in the file manager request and response.

| **Parameter** | **Type** | **Default** | **Explanation** |
| --- | --- | --- | --- |
| name | String | - | File name |
| dateCreated | String | - | Date in which file was created (UTC Date string). |
| dateModified | String | - | Date in which file was last modified (UTC Date string). |
| filterPath | String | - | Relative path to the file or folder. |
| hasChild | Boolean | - | Defines this folder has any child folder or not. |
| isFile | Boolean | - | Say whether the item is file or folder. |
| size | Number | - | File size |
| type | String | - | File extension |
| multipleFiles | Boolean | - | Say whether the details are about single file or multiple files. |

**Action Buttons**

The file manager has several menu buttons to access the file operations. The list of menu buttons available in the file manager is given in the following table.

| **Menu Button** | **Behaviour** |
| --- | --- |
| SortBy | Opens the sub menu to choose the sorting order and sorting parameter. |
| View | Opens the sub menu to choose the View. |
| Open | Navigates to the selected folder. Opens the preview for image files. |
| Refresh | Initiates the read operation for the current directory and displays the updated directory content. |
| NewFolder | Opens the new folder dialog box to receive the name for the new folder. |
| Rename | Opens the rename dialog box to receive the new name for the selected item. |
| Delete | Opens the delete dialog box to confirm the removal of the selected items from the file system. |
| Upload | Opens the upload box to select the items to upload to the file system. |
| Download | Downloads the selected item(s). |
| Details | Get details about the selected items and display them in details dialog box. |
| SelectAll | Selects all the files and folders displayed in the view section. |

The action menu buttons are present in the toolbar and context menu. The toolbar contains the buttons based on the selected items count, while the context menu will appear with a list based on the target.

**Toolbar**

The toolbar can be divided into two sections as right and left. Whenever the toolbar buttons exceed the size, the buttons present in the left section of the toolbar will be moved to the toolbar popup.

The following table provides the toolbar buttons that appear based on the selection.

|  |  |  |
| --- | --- | --- |
| **Selected Items Count** | **Left section** | **Right section** |
| 0 (none of the item ) | * SortBy * Refresh * NewFolder * Upload | * View * Details |
| 1 (single item selected) | * Delete * Download * Rename | * Selected items count * View * Details |
| >1 (multiple selection) | * Delete * Download | * Selected items count * View * Details |

**Context menu**

The following table provides the default context menu item and the corresponding target areas.

|  |  |  |
| --- | --- | --- |
| **Menu Name** | **Menu Items** | **Target** |
| Layout | * SortBy * View * Refresh * NewFolder * Upload * Details * Select all | * Empty space in the view section (details view and large icon view area). * Empty folder content. |
| Folders | * Open * Delete * Rename * Downloads * Details | * Folders in treeview, details view, and large icon view. |
| Files | * Open * Delete * Rename * Downloads * Details | * Files in details view and large icon view. |