

New to Telerik UI for Blazor? [Start a free 30-day trial](#)

# Chunk Upload

Updated on Jun 11, 2026

Chunk upload works by splitting a file into smaller parts (chunks) and sending them one by one in several sequential requests. These chunks are then reassembled at the remote endpoint into the final file.

## Enable Chunk Upload

To enable the chunk upload feature, add the `<UploadChunkSettings>` tag inside inside `<UploadSettings>`. See the [UploadChunkSettings API reference](#) for all available parameters.

```
<TelerikUpload>
  <UploadSettings>
    <UploadChunkSettings AutoRetryAfter="100"
                        Enabled="true"
                        MaxAutoRetries="1"
                        MetadataField="chunkMetadata"
                        Resumable="true"
                        Size="@((1024 * 1024))" />
  </UploadSettings>
</TelerikUpload>
```

## Meta Data

When posting files in chunks, the Upload component sends each chunk together with serialized meta data to the controller. To deserialize this meta data, define a class with the following properties. You can name the class, according to your preferences.

```
using using System.Runtime.Serialization;

[DataContract]
public class ChunkMetadata
{
    [DataMember(Name = "fileId" )]
    public string FileId { get; set; } = string.Empty;

    [DataMember(Name = "fileName" )]
    public string FileName { get; set; } = string.Empty;

    [DataMember(Name = "fileSize" )]
    public long FileSize { get; set; }

    [DataMember(Name = "contentType")]
    public string ContentType { get; set; } = string.Empty;

    [DataMember(Name = "chunkIndex")]
    public long ChunkIndex { get; set; }

    [DataMember(Name = "totalChunks")]
    public long TotalChunks { get; set; }
}
```

## Controller Implementation

The controller method that receives file chunks should be similar to the regular [controller method that receives complete files](#). The major differences are:

- The Save action method must expect an additional `string` argument with a name that matches the value of the `UploadChunkSettings MetadataField` parameter. By default, that is `"chunkMetadata"`.
- The action method must obtain the uploaded file name from the `FileName` property of the meta data object, instead of the `FileName` property of the `IFormFile` argument.
- The `FileStream FileMode` must be `Append` instead of `Create` for all chunks, except the first one. Check the `ChunkIndex` property of the meta data object.

The Upload sends the next chunk only after the previous one has been uploaded successfully.

As always, make sure that the Save action method name is consistent with the Upload `SaveUrl` parameter value.

### Chunk upload controller action method

```
using System.Runtime.Serialization;
using System.Runtime.Serialization.Json;

public async Task<IActionResult> SaveChunk(IFormFile files ,
[FromForm] string chunkMetadata)
{
    DataContractJsonSerializer dcSerializer =
new<typeof(ChunkMetadata)>();
    MemoryStream ms = new(Encoding.UTF8.GetBytes(chunkMetadata));

    string saveLocation =
Path.Combine(HostingEnvironment.WebRootPath, metadata.FileName);

    using FileStream fs = new(saveLocation, metadata.ChunkIndex == 0
? FileMode.Create : FileMode.Append);
    await files.CopyToAsync(fs);

    // ...
}
```

## Events

The Upload exposes events related to chunk uploading. See the examples in the [Upload Events](#) article.

- **OnPause**—fires when the user clicks on the Pause button during chunk upload.
- **OnResume**—fires when the user clicks on the Resume button during chunk upload.

The Upload renders Pause and Resume buttons only when `Resumable` is `true` in the `UploadChunkSettings`, which is by default.

## Example

The `UploadController` class below assumes that the project name and namespace is `TelerikBlazorApp`.

Make sure to enable controller routing in the app startup file (`Program.cs`). In this case, the file includes the following lines:

- `builder.Services.AddControllers();`
- `app.MapDefaultControllerRoute();`.

Also see:

- Section [Implement Controller Methods](#)
- Page [Upload Troubleshooting](#)

### Using Chunk Upload

```
Home.razor  UploadController.cs  ChunkMetadata.cs  Program.cs

using Microsoft.AspNetCore.Mvc;
using System.Runtime.Serialization.Json;
using System.Text;
using TelerikBlazorApp.Data;

namespace TelerikBlazorApp.Controllers;

[Route("api/[controller]/[action]")]
public class UploadController : ControllerBase
{
    public IWebHostEnvironment HostingEnvironment { get; set; }

    public UploadController(IWebHostEnvironment hostingEnvironment)
    {
        HostingEnvironment = hostingEnvironment;
    }

    [HttpPost]
    public async Task<IActionResult> SaveChunk(IFormFile files ,
[FromForm] string chunkMetadata)
    {
        if (files != null)
        {
            try
            {
                DataContractJsonSerializer dcSerializer =
new<typeof(ChunkMetadata)>();
                using MemoryStream ms =
new(Encoding.UTF8.GetBytes(chunkMetadata));

                if (dcSerializer.ReadObject(ms) is not ChunkMetadata
metadata)
                {
                    throw new NullReferenceException("Chunk metadata
serialization failed.");
                }

                var rootPath = HostingEnvironment.WebRootPath; //
save to wwwroot
                string saveLocation = Path.Combine(rootPath,
metadata.FileName);

                using FileStream fs = new(saveLocation,
metadata.ChunkIndex == 0 ? FileMode.Create : FileMode.Append);
                await files.CopyToAsync(fs);

                Response.StatusCode = 201;
            }
            catch (Exception ex)
            {
                Response.StatusCode = 500;
                await Response.WriteAsync($"Chunk upload failed.
Exception: {ex.Message}");
                return new EmptyResult();
            }
        }

        return new EmptyResult();
    }

    [HttpPost]
    public async Task<IActionResult> Remove([FromForm] string files )
    // "files" matches the Upload RemoveField value
    {
        if (!string.IsNullOrEmpty(files ))
        {
            try
            {
                var rootPath = HostingEnvironment.WebRootPath; //
delete from wwwroot
                var fileLocation = Path.Combine(rootPath, files );

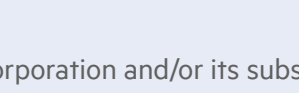
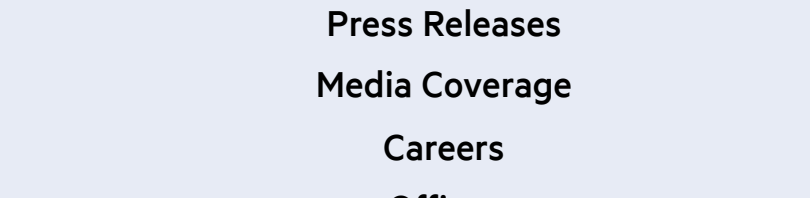
                if (System.IO.File.Exists(fileLocation ))
                {
                    System.IO.File.Delete(fileLocation );
                }
            }
            catch (Exception ex)
            {
                Response.StatusCode = 500;
                await Response.WriteAsync($"Delete failed. Exception:
{ex.Message}");
            }
        }

        return new EmptyResult();
    }
}
```

## See Also

- [Blazor Upload](#)

### Contact Us



Telerik and Kendo UI are part of Progress product portfolio. Progress is the leading provider of application development and digital experience technologies.

Company

Technology

Awards

Press Releases

Media Coverage

Careers

Offices

Copyright © 2026 Progress Software Corporation and/or its subsidiaries or affiliates. All Rights Reserved.

Progress and certain product names used herein are trademarks or registered trademarks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and/or other countries. See [Trademarks](#) for appropriate markings. All rights in any other trademarks contained herein are reserved by their respective owners and their inclusion does not imply an endorsement, affiliation, or sponsorship as between Progress and the respective owners.

[Terms of Use](#) | [Site Feedback](#) | [Privacy Center](#) | [Trust Center](#)

DO NOT SELL OR SHARE MY PERSONAL INFORMATION

