PH7X SYSTEMS ®

0.1.0 · CHAPTER 16

# Decide every SPFx package before its sites move

SharePoint Migration Field Guide · Chapter 16

pH7x Systems

2026-11-07

# 16. Decide every SPFx package before its sites move

> The catalogue holds the package, never its source. A package nobody can rebuild can be kept or replaced, never changed.

SharePoint Framework (SPFx) solutions are the estate's client-side code: web parts and extensions that run in the browser with the rights of whoever opens the page, delivered as an `.sppkg` package that an administrator approves in an app catalogue (enterprise guidance, checked on 6 November 2026). Chapter 15 left them to this chapter, because their questions differ: which version built each one, where its scripts load from, what it calls, and who can still build it.

> **Which SharePoint Framework packages does the estate run, built with which version and calling what, and which of them run unchanged at the destination?**

The decision this chapter leads to is a disposition for every package (keep, rebuild for the version the destination supports, or retire), recorded with its version, its source, what it calls and its owner, before the content moves.

## Which packages this chapter is for

Every `.sppkg` in the app catalogues of a SharePoint Server 2016 farm with Feature Pack 2 or a 2019 farm, moving to Subscription Edition by attach (Chapter 23) or into SharePoint Online by copy (Chapters 8 and 9), and the rarer reverse: an online package that a farm must run. Add-ins rebuilt as SPFx are Chapter 18's.

## What each platform runs

| WHAT | 2016 WITH FEATURE PACK 2 | 2019 | SUBSCRIPTION EDITION | SHAREPOINT ONLINE |
|---|---|---|---|---|
| Framework versions | v1.0 to v1.1 | v1.0 to v1.4.1 | v1.0 to v1.5, or v1.5.1 from Version 23H1 | All |
| Web parts | Classic pages | Classic and modern pages | Classic and modern pages | Yes |
| Extensions | None | Modern pages | Modern pages | Yes |
| Library components (v1.9.1) and later types | None | None | None | Yes |
| `AadHttpClient` and `MSGraphClient` (v1.4.1) | None | None documented | None documented | Once each scope is approved |
| Deployment to every site at once | None | Pages disagree | Pages disagree | Yes |
| Documented toolchain | Node.js 6.17.1, generator 1.6.0, gulp | Node.js 8.17.0, generator 1.10.0, gulp | As 2019 | v1.24.1, Node.js 22, 24 or 26, Heft |

Sources: compatibility reference, SPFx on 2016, SPFx on 2019 and Subscription Edition, Version 23H1, the v1.4.1 and v1.9.1 release notes, tenant-scoped deployment and the enterprise guidance, checked on 6 November 2026.

Microsoft's pages disagree on Subscription Edition: the compatibility table gives v1.0 to v1.5, Version 23H1 adds support for v1.5.1, and the development page still gives it "all the same dependencies and requirements" as 2019, whose version is v1.4.1. We read v1.5.1 as the ceiling from 23H1, and v1.4.1 as the version every page supports. On deploying to every site, the tenant-scoped page includes SharePoint 2019, while the enterprise guidance says "Tenant-wide deployment is available only in SharePoint Online": read what each site has installed.

A farm "only supports the version that matches the server-side dependencies of the deployed packages" (SPFx on 2019 and Subscription Edition), so a package running on 2016 was built at v1.1 or earlier, and one on 2019 at v1.4.1 or earlier. **On Subscription Edition, the version never stops a farm package**, and SharePoint Online accepts every version. What stops packages is where their scripts load from, what they call, and who can still build them.

## Who can still build each package

Changing a package means building it again with its target's toolchain. The generator "starting with v1.13.0, only supports projects for SharePoint Online"; Microsoft documents workarounds for building v1.4.1 on Node.js 12.18.1, 14.17.1 or 16.15.0, and its Subscription Edition page gives no setup for v1.5.1.

Online, "Starting with SPFx v1.24, the gulp-based toolchain is unsupported" (Heft toolchain, checked on 6 November 2026). Keeping packages on a farm and rebuilding others online means two build environments.

Moving a farm project to a current version is work of its own: "Manually updating package.json alone will result in build failures" (v1.4.1 release notes). The same notes suggest the CLI for Microsoft 365's `m365 spfx project upgrade` command to list the changes; it is community-maintained, with no Microsoft SLA.

## Find every package, its version and what it calls

Each web application has at most one catalogue, a site collection (Update-SPAppCatalogConfiguration, checked on 6 November 2026), whose Apps for SharePoint library holds the packages (enterprise guidance). From an elevated SharePoint Management Shell on the old farm, outside business hours:

```
# Reads only. Saves every .sppkg in an Apps for SharePoint library and lists them in packages.csv.
Write-Host "Saving the SPFx packages of every app catalogue on this farm to .\packages"
$folder = (New-Item -ItemType Directory -Force .\packages).FullName
$rows = foreach ($site in Get-SPSite -Limit All) {
    $library = $site.RootWeb.Lists.TryGetList("Apps for SharePoint")
    if ($library) { foreach ($item in $library.Items) { if ($item.File.Name -like "*.sppkg") {
        $saved = "$($site.ID)-$($item.File.Name)"
        [IO.File]::WriteAllBytes("$folder\$saved", $item.File.OpenBinary())
        [pscustomobject]@{ Catalogue = $site.Url; Package = $item.File.Name; Saved = $saved } } }
    $site.Dispose()
}
$rows | Export-Csv .\packages.csv -NoTypeInformation
```

We have not run it against a farm: run it on a test farm first. Four answers come from elsewhere:

- **Version.** Microsoft documents reading it from the project: `.yo-rc.json` holds the generator version that created it, `package.json` the framework packages, `npm-shrinkwrap.json` any locked versions (SPFx on 2019 and Subscription Edition). Without the source, we do not know a documented way: record the farm's ceiling and "source unknown".
- **Script addresses.** From v1.4 a package carries its scripts by default, served by the catalogue (enterprise guidance); a 2016 package cannot, so its files sit on a CDN, a server or a farm site (SPFx on 2016). `cdnBasePath`, `externals` and scripts loaded in code name the rest (CSP in SharePoint Online, checked on 6 November 2026).
- **Calls.** Each API and service, and the identity each relies on: the farm's sign-in is Chapter 21, the other end Chapter 22.
- **Use.** A solution is added to a site "in the same way as for SharePoint Add-ins" (enterprise guidance), so we would expect the Get-SPAppInstance inventory of Chapter 18 (checked on 6 November 2026) to include it. We have not confirmed that: compare a sample with Site Contents, and have the owner name the pages that depend on each package.

## What an attach carries, and what a copy leaves

| PART | ATTACHED TO SUBSCRIPTION EDITION | COPIED INTO SHAREPOINT ONLINE |
|---|---|---|
| **Catalogue and packages** | Arrive with the catalogue's content database | Uploaded to the tenant catalogue or a site collection catalogue |
| **Web application's catalogue link** | Set again with `Update-SPAppCatalogConfiguration` | Not needed |
| **App Management** | Built again: not among the service applications upgraded from a database | Not needed |
| **Scripts hosted outside (every 2016 package)** | Work while their address does | Reach users' browsers and pass the policy below |
| **Sites that installed a package** | Arrive with their content databases | Install it again, per site or for every site |

Sources: Update-SPAppCatalogConfiguration, tenant-scoped deployment, Add-SPOSiteCollectionAppCatalog and the enterprise guidance, checked on 6 November 2026; create the farm, checked on 7 November 2026.

An attach upgrades "only the content for your environment and not the configuration settings" (create the farm). Microsoft's SPFx steps on a farm start with App Management and "other dependent service applications", then the catalogue (SPFx on 2019 and Subscription Edition); the old App Management database comes across only where Chapter 19 moves it. You reapply blocked file types by hand (create the farm), which matters where a 2016 farm was changed to accept the `.json` files of SPFx assets. Whether packages in an attached catalogue load without being deployed again, the pages above do not say, and we do not know: we would set the catalogue first, then open, as a user, a page that uses each package.

## What a farm package meets online

Microsoft dated enforcement of a Content Security Policy in SharePoint Online from 1 March 2026, deferrable by a tenant to 1 June 2026. Scripts inside a package load; `cdnBasePath` and `externals` addresses become trusted script sources when the solution is installed; any other script needs an entry, and enforcement blocks inline script, on modern pages only (CSP in SharePoint Online, checked on 6 November 2026). The same page still describes reporting mode, so test each page that uses a package with `csp=enforce` in its address.

For farms, Microsoft lists web parts and extensions only, and `AadHttpClient` and `MSGraphClient` arrived in v1.4.1 "for preview usage within SharePoint Online" (v1.4.1 release notes). A package rebuilt to call Microsoft Graph or another Entra ID-secured API lists its scopes in `webApiPermissionRequests`; deployment "creates permission requests", which a Global Administrator approves on the API access page for Microsoft APIs, or an Application Administrator

for third-party ones (Entra ID-secured APIs, API access, checked on 6 November 2026). Each approval is a delegated grant to the SharePoint Online Client Extensibility Web Application Principal (a newer principal in some tenants), shared by every SPFx component and script in the tenant; denial does not block deployment, and removing the package does not revoke the grant. We recommend recording each package's narrowest scope and its approver before the copy: only that record says which packages rely on a grant.

Domain-isolated web parts, a v1.8 feature no farm runs, stopped working on 2 April 2026 (retirement, both checked on 6 November 2026), and the SharePoint Framework Online workbench, deprecated on 13 May 2026, retires on 1 December 2026, so rebuilt web parts are tested on real pages (enterprise guidance).

The reverse is narrower: an online package above a farm's ceiling does not run there, and rebuilt at v1.4.1 or v1.5.1 it loses library components, later component types and the Entra ID clients; what it did with Graph becomes Chapter 22's question.

## Keep, rebuild or retire

**Keep** a package that ran on the old farm and runs at the destination: on Subscription Edition once the catalogue is rebuilt and its script addresses resolve; online once it loads under `csp=enforce` and calls only what the tenant approves. Keeping one without its source is a decision never to change it.

**Rebuild for the destination's version** when the package is needed and keeping fails, or it must change and nobody can build it. For Subscription Edition we would build at v1.4.1 unless v1.5.1 is needed, since every Microsoft page supports the first; online, at v1.24.1 with Heft. Rebuild from the source, or from a specification its owner agrees, as Chapter 18 does for add-ins; for a bought package, from its supplier.

**Retire** it when no site installs it, the product now does its job, or its owner releases it in writing. Removing a solution "won't remove any data or information created by the actual client-side solution" (enterprise guidance): retiring settles the package, not its data.

We recommend recording each disposition with version, source, build environment, script addresses, calls, scopes and the owner's agreement before the content moves: the old farm is the last place every package still runs as built.

## Out of scope

Farm solutions are Chapter 15, elevated code Chapter 17, add-ins Chapter 18, sign-in Chapter 21, the systems a package calls Chapter 22 and installing what you keep Chapter 23. SPFx development, Teams, Viva and Copilot components, and migration tooling (Chapter 29) are not covered.

**Before you continue**

- [ ] Every app catalogue is found and every `.sppkg` in it saved, with the web application it serves
- [ ] Each package names its framework version from its source, or "source unknown"
- [ ] Each package names where its scripts load from and every API or service it calls
- [ ] Each package names the sites that installed it, the pages that use it and an owner who confirmed them
- [ ] For an attach: App Management, catalogue links and blocked file types rebuilt, every kept package loaded on the rehearsal farm
- [ ] For SharePoint Online: every kept package loaded with `csp=enforce`, every requested scope with an approver
- [ ] Each package kept, rebuilt for the destination's version or retired, with its reason and owner's agreement, before its sites move