Best Practices Guide

Modernizing Desktop Applications

DigitalWorkplace.pro

Introduction

In today's rapidly evolving technological landscape, desktop applications remain a cornerstone of enterprise productivity, user workflows, and specialized software solutions.

However, as user expectations shift toward seamless experiences, cloud integration, and cross-platform compatibility, the need to modernize legacy desktop applications has never been more pressing. Modernization is not merely about updating outdated software development life cycle — it's about reimagining how desktop applications can leverage contemporary tools, frameworks, and architectures to deliver enhanced performance, scalability, and maintainability while preserving the robustness that users rely on.

This guide, *Modernizing Desktop Applications: Technical Best Practices*, is designed to equip developers, architects, and IT leaders with actionable strategies and proven methodologies to transform aging desktop applications into future-ready solutions. From assessing the current state of your application to implementing cutting-edge technologies like containerization, microservices, and modern UI frameworks, this resource distills industry insights into a practical roadmap. Whether you're extending the life of a decades-old codebase or preparing for a full architectural overhaul, these best practices will help you balance innovation with stability, ensuring your application meets the demands of today's users while remaining adaptable for tomorrow's challenges.

In the chapters that follow, we'll explore key considerations—such as evaluating modernization drivers, selecting the right tools and platforms, optimizing performance, and ensuring security—while providing real-world examples and step-by-step guidance. By adopting these best practices, you can unlock new opportunities, reduce technical debt, and deliver desktop applications that stand the test of time. Let's embark on this journey to breathe new life into the software that powers your business and your users' success.

Assessing the Need for Modernization

Before embarking on the journey of modernizing a desktop application, it's critical to establish a clear understanding of why modernization is necessary and what goals you aim to achieve. Not every application requires a complete overhaul—some may benefit from incremental updates, while others demand a full rewrite to remain viable. This section outlines a structured approach to evaluating your application's current state, identifying pain points, and aligning modernization efforts with business and technical objectives.

Step 1: Evaluate the Current Application

Begin by conducting a thorough assessment of your existing desktop application. Key areas to examine include:

- **Performance**: Are users experiencing slow load times, crashes, or resource inefficiencies? Benchmark the application against modern standards to identify bottlenecks.
- **Compatibility**: Does the application run smoothly on current operating systems (e.g., Windows 11, macOS Sonoma) and hardware configurations? Are there dependencies on outdated libraries or frameworks?
- **User Experience**: Is the interface intuitive and aligned with contemporary design principles, or does it feel clunky and dated compared to modern alternatives?
- **Maintenance**: How difficult is it to fix bugs, add features, or integrate with new systems? A high level of technical debt often signals the need for modernization.
- **Security**: Does the application meet current security standards, such as encryption protocols, secure authentication, and protection against vulnerabilities like SQL injection or cross-site scripting?

Step 2: Identify Business Drivers

Modernization should align with organizational goals. Engage stakeholders to pinpoint the business case for updating the application. Common drivers include:

- **Cost Reduction**: Legacy systems often incur high maintenance costs due to obsolete hardware, licensing fees, or specialized expertise.
- **Competitive Advantage**: Modernized applications can offer new features, better performance, or integration with emerging technologies like AI and cloud services.
- **User Retention**: An outdated application risks alienating users who expect responsive, visually appealing, and feature-rich software.
- **Regulatory Compliance**: Evolving standards (e.g., GDPR, HIPAA) may necessitate updates to ensure data privacy and security.

Microsoft's Application Virtualization (App-V) solution has long been a cornerstone for enterprise organizations seeking to streamline application delivery and management. Introduced in 2006 as part of the Microsoft Desktop Optimization Pack (MDOP), App-V enabled IT teams to virtualize applications, isolating them from the underlying operating system and other software to reduce conflicts, simplify deployments, and enhance compatibility. By packaging applications into portable, self-contained units, App-V allowed centralized management and delivery without requiring traditional installations, making it a popular choice for large-scale enterprises with complex IT environments.

As of March 23, 2025, App-V remains widely used due to its integration with System Center Configuration Manager (SCCM) and its ability to support legacy Windows applications in highly regulated industries like finance, healthcare, and government. Its strengths—such as offline functionality, minimal end-user disruption, and compatibility with Windows desktops—cemented its position as the most commonly adopted application virtualization solution for enterprises over the past two decades.

However, Microsoft has signaled the end of the road for App-V. The company announced that App-V would reach its end-of-life (EOL), with mainstream support for the latest version (App-V 5.1) having ended on January 10, 2023, and extended support scheduled to conclude on **April 14, 2026**. After this date, Microsoft will no longer provide security updates, patches, or technical support, leaving organizations vulnerable to emerging threats and compatibility issues with modern Windows versions (e.g., Windows 11). Additionally, Microsoft has shifted its focus to cloud-based management solutions like Microsoft Intune and Azure Virtual Desktop (AVD), integrating lightweight virtualization alternatives such as MSIX and Windows 365, which align with hybrid and remote work trends.

This impending EOL underscores a critical need for desktop application modernization. Enterprises relying on App-V face several challenges that necessitate a proactive transition:

1. **Security Risks**: Without updates post-2026, App-V deployments will become susceptible to unpatched vulnerabilities, a significant concern for compliance-driven organizations.
2. **Compatibility Gaps**: As Windows evolves, App-V's lack of support for newer OS features (e.g., Windows 11's enhanced security models) limits its viability.
3. **Operational Inefficiency**: Maintaining an aging virtualization infrastructure diverts resources from innovation, increasing technical debt.
4. **User Expectations**: Modern workforces demand seamless, cloud-integrated experiences that App-V, designed for a pre-cloud era, struggles to deliver.

The shift away from App-V highlights broader trends in desktop application management—moving from on-premises, siloed solutions to agile, cloud-native platforms. Modernization offers enterprises an opportunity to replace App-V with alternatives like Intune for application delivery, MSIX for packaging, or AVD for full desktop virtualization. These solutions provide benefits such as cross-platform support, real-time updates, and integration with Microsoft 365 security features, aligning with the needs of distributed workforces.

In summary, App-V's historical dominance in enterprise application virtualization is nearing its end, with its EOL marking a pivotal moment for organizations to rethink their desktop application strategies. Modernizing away from App-V is not just a response to obsolescence—it's a chance to enhance security, scalability, and user experience, ensuring applications remain robust and relevant in a cloud-first world. Enterprises must act swiftly to assess their App-V dependencies, explore successors, and implement a migration plan to stay ahead of this inevitable transition.

With Microsoft App-V approaching its end-of-life on April 14, 2026, and the broader push toward modern desktop application management, MSIX has emerged as Microsoft's recommended successor for application packaging and delivery. Introduced in 2018, MSIX combines the best of MSI (Windows Installer) and App-V, offering a universal packaging format that supports modern deployment via Microsoft Intune, the Microsoft Store, or other management tools. However, MSIX isn't a one-size-fits-all solution, and its adoption has faced challenges like complexity, limited legacy app support, and a steep learning curve. For organizations seeking alternatives to MSIX, several options exist, each with unique strengths tailored to specific use cases. Below, we explore the leading MSIX alternatives for application virtualization, packaging, and delivery.

---

1. App-V (Legacy Continuation with Third-Party Support)

- **Overview**: Although Microsoft is phasing out App-V, some third-party vendors and IT teams may extend its life through custom support or hybrid approaches.
- **Strengths**:
    - Familiar to enterprises with existing App-V deployments.
    - Robust isolation for legacy applications.
    - Offline functionality for environments without constant connectivity.
- **Weaknesses**:
    - No official updates or security patches post-2026.
    - Ties organizations to an aging infrastructure.
    - Limited integration with modern cloud tools like Intune.
- **Use Case**: Temporary stopgap for organizations needing more time to migrate away from App-V while maintaining compatibility with complex legacy apps.
- **Tools**: Leverage existing App-V infrastructure with tools like Flexera's AdminStudio for continued packaging support.

---

2. Citrix App Layering

- **Overview**: A virtualization solution from Citrix that separates applications into manageable layers, deployable across physical, virtual, or cloud environments.
- **Strengths**:
    - Granular control over app layers, enabling easy updates and rollbacks.
    - Strong integration with Citrix Virtual Apps and Desktops for enterprise-scale delivery.
    - Supports legacy apps alongside modern ones.
- **Weaknesses**:
    - Requires a Citrix ecosystem, increasing costs and complexity.
    - Less lightweight than MSIX for standalone app packaging.
- **Use Case**: Enterprises already using Citrix for VDI (Virtual Desktop Infrastructure) or needing advanced layering for complex app portfolios.
- **Tools**: Citrix App Layering console, paired with Citrix Provisioning Services.

---

3. VMware App Volumes

- **Overview**: Part of VMware's Workspace ONE and Horizon suites, App Volumes delivers applications as virtualized "volumes" attached to desktops in real time.
- **Strengths**:
    - Dynamic app delivery without modifying the base OS image.
    - Strong support for VDI and hybrid environments.
    - Simplifies management of large app catalogs.
- **Weaknesses**:
    - Tied to VMware infrastructure, requiring significant investment.
    - Not ideal for standalone desktop deployments outside VDI.

- **Use Case**: Organizations with VMware Horizon deployments or those prioritizing VDI over traditional desktops.
- **Tools**: VMware App Volumes Manager and Dynamic Environment Manager (DEM).

---

4. Turbo.net (Formerly Spoon)

- **Overview**: A cloud-enabled application virtualization platform that packages and streams apps without requiring OS-level installation.
- **Strengths**:
    - Lightweight and portable app containers.
    - Cross-platform support (Windows, macOS, Linux).
    - Cloud streaming option reduces local resource demands.
- **Weaknesses**:
    - Smaller enterprise adoption compared to Microsoft or Citrix solutions.
    - May require additional configuration for complex dependencies.
- **Use Case**: Small to mid-sized organizations or teams needing a flexible, cloud-friendly alternative with minimal infrastructure overhead.
- **Tools**: Turbo Studio for packaging, Turbo Server for centralized management.

---

5. Advanced Installer with App Virtualization

- **Overview**: A third-party tool traditionally used for MSI packaging, now offering virtualization features as an alternative to MSIX.
- **Strengths**:
    - Simple conversion of MSI/EXE installers into virtualized packages.
    - Familiar workflow for teams accustomed to traditional packaging.
    - No dependency on Microsoft-specific ecosystems.
- **Weaknesses**:

    - Less integrated with modern cloud management tools like Intune.
    - Virtualization features are not as mature as dedicated solutions.
- **Use Case**: Organizations seeking a straightforward, MSI-like experience with virtualization capabilities for legacy apps.
- **Tools**: Advanced Installer Enterprise edition with Virtualization Pack.

---

6. Cloudhouse Containers

- **Overview**: A specialized solution for packaging and running legacy Windows applications on modern systems without rewriting code.
- **Strengths**:
    - Excels at modernizing old apps (e.g., Windows XP/7-era software) for Windows 10/11.
    - Strong isolation and dependency management.
    - Lightweight compared to full VDI solutions.
- **Weaknesses**:
    - Niche focus on legacy apps limits broader applicability.
    - Smaller vendor ecosystem and community support.
- **Use Case**: Enterprises with critical legacy applications incompatible with MSIX or modern OS versions.
- **Tools**: Cloudhouse Compatibility Containers toolkit.

---

7. Docker for Windows (Containerization)

- **Overview**: While traditionally associated with server-side workloads, Docker can containerize Windows desktop applications for consistent deployment.
- **Strengths**:
    - Industry-standard containerization with broad tooling support.

    - Ensures environment consistency across development and production.
    - Open-source and highly customizable.
- **Weaknesses**:
    - Steep learning curve for desktop app teams unfamiliar with containers.
    - Overhead for simple apps compared to MSIX.
    - Limited native integration with Intune (requires custom workflows).
- **Use Case**: DevOps-oriented teams or organizations already using Docker, needing to modernize desktop apps alongside server workloads.
- **Tools**: Docker Desktop for Windows, paired with custom scripts for deployment.

Comparison Table

Solution	Best For	Cloud Integration	Legacy Support	Complexity
App-V (Legacy)	Temporary App-V continuity	Low	High	Low
Citrix App Layering	Citrix VDI environments	Moderate	High	High
VMware App Volumes	VMware VDI deployments	Moderate	High	High
Turbo.net	Lightweight cloud delivery	High	Moderate	Moderate
Advanced Installer	MSI-like simplicity	Low	High	Low

Cloudhouse	Legacy app modernization	Low	Very High	Moderate
Docker	Containerized workflows	High	Moderate	High

Key Considerations When Choosing an MSIX Alternative

- **Legacy Compatibility**: If your portfolio includes older apps, prioritize solutions like Cloudhouse or Advanced Installer over Docker.
- **Cloud Readiness**: For hybrid or remote workforces, Turbo.net or VMware App Volumes offer stronger cloud integration than standalone tools.
- **Infrastructure Alignment**: Leverage existing investments (e.g., Citrix, VMware) to reduce costs and training needs.
- **Scalability**: Larger enterprises may favor Citrix or VMware, while smaller teams might opt for Turbo.net or Advanced Installer.
- **Cost**: Open-source options like Docker contrast with premium solutions like Citrix, impacting budget decisions.

Conclusion

While MSIX offers a modern, Microsoft-backed packaging standard, its alternatives provide flexibility for organizations with diverse needs. Whether preserving legacy apps (Cloudhouse), scaling VDI (Citrix/VMware), or embracing cloud-native delivery (Turbo.net), these options ensure enterprises can move beyond App-V and MSIX limitations. Assess your app catalog, IT infrastructure, and modernization goals to select the best fit, and pilot test your chosen solution to validate compatibility and performance before full deployment.

Microsoft's App-V and MSIX are both application packaging and delivery solutions designed to simplify deployment and management in enterprise environments, but they cater to different eras and paradigms of desktop application management. App-V, a virtualization-focused technology, has been a staple for nearly two decades, while MSIX, introduced in 2018, represents Microsoft's modern vision for a universal packaging format. With App-V reaching its end-of-life on April 14, 2026, understanding their differences is crucial for organizations planning a transition. Below is a detailed comparison of MSIX and App-V across key dimensions.

---

1. Core Technology and Approach

- **App-V**:
  - **Type**: Application virtualization.
  - **Approach**: Virtualizes applications into isolated containers that run in a sandboxed environment, separate from the OS and other apps. The app doesn't install traditionally but streams components as needed.
  - **Key Feature**: Isolation prevents conflicts by redirecting file and registry changes to a virtual layer.
- **MSIX**:
  - **Type**: Application packaging.
  - **Approach**: Packages apps into a single, installable container that integrates natively with the OS. It uses a declarative installation model rather than virtualization.
  - **Key Feature**: Combines MSI simplicity with AppX (Universal Windows Platform) features, enabling clean installs/uninstalls without sandboxing.

Comparison: App-V focuses on runtime isolation, ideal for legacy apps prone to conflicts, while MSIX emphasizes streamlined packaging and native integration, aligning with modern app deployment.

---

2. Deployment and Management

- **App-V**:
    - **Deployment**: Typically managed via System Center Configuration Manager (SCCM) or standalone servers. Supports streaming from a central server or local caching for offline use.
    - **Management**: Requires an App-V client on devices and a sequencing process to virtualize apps.
    - **Scalability**: Suited for large enterprises with on-premises infrastructure.
- **MSIX**:
    - **Deployment**: Deployable via Microsoft Intune, SCCM, Microsoft Store, or manual installation. Supports cloud-based and sideloading options.
    - **Management**: No client prerequisite beyond Windows 10/11; managed through modern MDM (Mobile Device Management) tools.
    - **Scalability**: Designed for hybrid and cloud environments, with broader deployment flexibility.

Comparison: App-V's deployment is tied to traditional infrastructure like SCCM, while MSIX integrates with cloud-native tools, offering greater agility and less overhead.

---

3. Compatibility

- **App-V**:

    - **OS Support**: Works on Windows 7 through Windows 11, though support wanes post-2026.
    - **App Support**: Excels with legacy Win32 apps, including those with complex dependencies or conflicts.
    - **Limitations**: Struggles with modern UWP apps and lacks native Windows 11 optimization.
- **MSIX**:
    - **OS Support**: Native to Windows 10 (version 1709+) and Windows 11.
    - **App Support**: Supports Win32, UWP, and .NET apps, but older apps may require repackaging or conversion (e.g., via MSIX Packaging Tool).
    - **Limitations**: Limited compatibility with apps requiring deep system integration or kernel-level drivers.

Comparison: App-V is a lifeline for legacy apps, while MSIX targets modern apps and OS versions, sometimes requiring extra effort for older software.

---

4. Performance and Resource Usage

- **App-V**:
    - **Performance**: Virtualization adds overhead due to streaming and sandboxing, potentially slowing startup times or increasing resource use.
    - **Storage**: Virtualized apps can be cached locally, consuming disk space proportional to usage.
- **MSIX**:
    - **Performance**: Native installation reduces runtime overhead, offering faster startup and execution compared to virtualization.
    - **Storage**: Efficient packaging with deduplication (e.g., via MSIX App Attach in Azure Virtual Desktop) minimizes disk footprint.

Comparison: MSIX outperforms App-V in speed and efficiency, as it avoids virtualization layers, though App-V's isolation can mitigate app-specific performance issues.

---

5. Security

- **App-V**:
    - **Security**: Isolation enhances security by containing app changes, reducing system-wide vulnerabilities. However, no updates post-2026 will expose unpatched risks.
    - **Updates**: Managed manually or via SCCM, lacking modern automation.
- **MSIX**:
    - **Security**: Signed packages ensure authenticity, and native integration leverages Windows security features (e.g., Defender). Cleaner uninstalls reduce residual risks.
    - **Updates**: Supports automatic updates via the Microsoft Store or Intune, improving patch management.

Comparison: MSIX offers stronger long-term security with modern update mechanisms, while App-V's isolation is a short-term advantage overshadowed by its EOL.

---

6. User Experience

- **App-V**:
    - **Experience**: Users see virtualized apps as native, but initial streaming may delay access. Offline caching ensures availability without internet.
    - **Uninstall**: Leaves no system traces but requires App-V client cleanup if fully removed.
- **MSIX**:

- **Experience**: Seamless installation and execution, indistinguishable from traditional apps. Faster access with no streaming delay.
- **Uninstall**: Clean removal with no leftovers, adhering to modern app lifecycle standards.

Comparison: MSIX delivers a smoother, more modern user experience, while App-V's virtualization can introduce minor usability quirks.

---

7. Development and Packaging

- **App-V**:
  - **Process**: Requires sequencing—an involved process of capturing app behavior in a virtual environment using the App-V Sequencer.
  - **Tools**: App-V Sequencer, SCCM integration.
  - **Complexity**: High learning curve for sequencing complex apps.
- **MSIX**:
  - **Process**: Packaging via the MSIX Packaging Tool, which converts MSI/EXE installers or captures installs. Simpler declarative model.
  - **Tools**: MSIX Packaging Tool, Visual Studio, Intune.
  - **Complexity**: Easier for modern apps, but legacy conversion can be challenging.

Comparison: MSIX simplifies packaging for new apps, while App-V's sequencing is more robust for legacy complexity at the cost of effort.

---

8. Future-Proofing

- **App-V**:

    - **Status**: End-of-life on April 14, 2026; no further development or support.
    - **Roadmap**: Deprecated in favor of MSIX and cloud solutions like Intune and Azure Virtual Desktop.
- **MSIX**:
    - **Status**: Actively developed, with ongoing enhancements (e.g., MSIX App Attach).
    - **Roadmap**: Core to Microsoft's modern management strategy, integrated with Windows 11 and beyond.

Comparison: MSIX is future-ready, while App-V is a legacy technology nearing obsolescence.

Summary Table

Aspect	App-V	MSIX
Technology	Virtualization	Packaging
Deployment	SCCM, streaming	Intune, Store, sideloading
Compatibility	Legacy Win32	Modern Win32, UWP
Performance	Higher overhead	Lower overhead
Security	Isolation, EOL risks	Signed, modern updates
User Experience	Streaming delays	Native, seamless

Packaging	Sequencing, complex	Conversion, simpler
Future Support	Ends 2026	Ongoing development

Conclusion

App-V and MSIX serve distinct purposes: App-V excels at isolating and delivering legacy applications in traditional on-premises setups, while MSIX streamlines packaging and deployment for modern apps in cloud-integrated environments. For organizations still using App-V, MSIX offers a forward-looking replacement with better performance, security, and scalability—albeit with potential hurdles for legacy app conversion. As App-V's EOL looms, transitioning to MSIX (or its alternatives) is essential for maintaining a secure, efficient, and future-proof application management strategy. Evaluate your app portfolio and infrastructure to determine if MSIX meets your needs or if a hybrid approach is required during migration.

MSIX App Attach is an advanced feature of the MSIX packaging framework that enables dynamic, on-demand delivery of applications to Windows devices, particularly in virtualized or cloud-based environments like Azure Virtual Desktop (AVD) and Windows 365. Introduced by Microsoft to enhance application deployment efficiency, MSIX App Attach leverages the MSIX packaging format to separate app delivery from the underlying operating system image, allowing administrators to attach and detach applications to user sessions without traditional installation. This approach streamlines management, reduces resource overhead, and aligns with modern, cloud-native desktop strategies. Below is a detailed explanation of MSIX App Attach, its mechanics, benefits, and use cases.

What is MSIX App Attach?

MSIX App Attach extends the MSIX framework by enabling applications packaged as MSIX containers to be "attached" to a Windows environment at runtime, rather than being fully installed on the device or virtual machine (VM). The application is stored as a virtual disk (e.g., VHD, VHDX, or CIM file) and mounted dynamically when a user logs in or requests the app. Once attached, the app behaves as if it were natively installed, integrating seamlessly with the OS, but it can be detached just as easily, leaving no residual footprint.

This contrasts with traditional MSIX deployment (which installs the app onto the device) and App-V (which virtualizes apps in a sandbox). MSIX App Attach is essentially a hybrid model: it uses MSIX's clean packaging while borrowing virtualization concepts to deliver apps on-demand without modifying the base OS image.

How MSIX App Attach Works

The process involves several key steps and components:

---

1. **Packaging the Application**:
    - An application is packaged into an MSIX container using tools like the MSIX Packaging Tool. This creates a single, signed package with all app files, dependencies, and metadata.
    - The MSIX package is then converted into a virtual disk format (e.g., VHDX) that can be mounted by Windows.
2. **Storing the App**:
    - The virtual disk is stored in a central location, such as Azure Files, a network share, or a local storage pool, accessible to the target devices or VMs.
3. **Attaching the App**:
    - When a user logs into a session (e.g., on Azure Virtual Desktop), the system mounts the virtual disk containing the MSIX package.
    - The OS registers the app's metadata (e.g., shortcuts, file associations) without copying files to the local disk, making it instantly available to the user.
4. **Running the App**:
    - The application runs natively, leveraging the mounted disk for its files and runtime needs. It integrates with the desktop, Start menu, and other OS features as a standard app.
5. **Detaching the App**:
    - Upon logout or policy trigger, the virtual disk is unmounted, and the app is unregistered from the session, leaving the OS unchanged.

Key Components

- **MSIX Package**: The core application container, ensuring consistency and security.
- **Virtual Disk (VHD/VHDX/CIM)**: Stores the MSIX package and mounts it to the system.

- **Host Pool (in AVD)**: The virtualized environment where apps are attached to user sessions.
- **Management Tools**: Azure Virtual Desktop, Intune, or custom scripts to orchestrate attachment/detachment.

Benefits of MSIX App Attach

1. **Efficient Resource Use**:
    - Reduces disk space by avoiding full installations on each device or VM. A single virtual disk can serve multiple users.
    - Minimizes OS image bloat, as apps are not baked into the base image.
2. **Simplified Management**:
    - Centralized app storage and updates—update the virtual disk once, and all users get the latest version.
    - No need to rebuild or redeploy OS images for app changes.
3. **Dynamic Delivery**:
    - Apps can be attached or detached based on user roles, policies, or session demands, enhancing flexibility in multi-user environments.
4. **Clean Application Lifecycle**:
    - Inherits MSIX's clean install/uninstall behavior, avoiding registry clutter or leftover files common in traditional deployments.
5. **Scalability**:
    - Ideal for cloud environments like AVD, where rapid provisioning of apps across thousands of sessions is critical.
6. **Security**:
    - Signed MSIX packages ensure authenticity, and detachment ensures no persistent changes to the OS, reducing attack surfaces.

Limitations and Challenges

1. **Compatibility**:
    - Primarily supports Win32 and UWP apps packaged as MSIX. Legacy apps requiring deep system integration (e.g., drivers) may need repackaging or alternative solutions.
2. **Infrastructure Dependency**:
    - Requires a virtualized setup (e.g., AVD) or custom scripting for physical devices, limiting its use in traditional desktop scenarios.
3. **Learning Curve**:
    - Packaging and managing virtual disks add complexity compared to standard MSIX or App-V workflows.
4. **Windows Version**:
    - Requires Windows 10 Enterprise (version 2004+) or Windows 11, with full functionality in AVD environments.

Use Cases

- **Azure Virtual Desktop (AVD)**:
    - Deliver apps to remote users without bloating VM images, ideal for call centers or remote workforces.
- **Multi-User Environments**:
    - Attach specific apps to shared Windows Server RDS (Remote Desktop Services) sessions based on user roles.
- **Rapid Provisioning**:
    - Quickly deploy updated apps to thousands of virtual desktops without downtime.
- **Dev/Test Scenarios**:

- Attach apps to temporary VMs for testing, then detach without affecting the base system.

Comparison to App-V

Aspect	MSIX App Attach	App-V
Delivery	Mounts virtual disk on-demand	Streams virtualized app
Installation	No install, native runtime	No install, sandboxed runtime
Isolation	Minimal, integrates with OS	Full virtualization
Management	Intune, AVD, cloud-focused	SCCM, on-premises focus
Future Support	Actively developed	EOL April 2026
Use Case	Cloud/VDI environments	Legacy app isolation

MSIX App Attach trades App-V's heavy virtualization for a lightweight, cloud-optimized delivery model, sacrificing some isolation for better performance and integration.

Getting Started

1. **Package the App**: Use the MSIX Packaging Tool to create an MSIX package from an MSI or EXE installer.

---

2. **Create a Virtual Disk**: Convert the MSIX into a VHDX using tools like Hyper-V Manager or scripts (e.g., Microsoft's sample PowerShell scripts).
3. **Configure Storage**: Upload the VHDX to Azure Files or a network share.
4. **Set Up AVD**: In the AVD portal, configure MSIX App Attach under *Applications* > *MSIX Packages*, linking the virtual disk and assigning it to host pools.
5. **Test**: Validate app availability and performance in a pilot session.

Conclusion

MSIX App Attach is a powerful evolution of Microsoft's application delivery strategy, bridging traditional packaging with cloud-scale virtualization. It's particularly valuable in Azure Virtual Desktop and Windows 365 environments, where dynamic app provisioning enhances efficiency and user experience. While it doesn't fully replace App-V's isolation capabilities, it offers a modern, scalable alternative for organizations embracing cloud-native desktop management. As App-V nears EOL, MSIX App Attach stands out as a future-ready solution for delivering apps in a rapidly evolving IT landscape.

MSIX is Microsoft's modern application packaging format, designed to streamline deployment, ensure clean installations, and support both traditional Win32 and Universal Windows Platform (UWP) applications across Windows 10 and 11. To create MSIX packages, developers and IT professionals rely on specialized tools that convert existing installers (e.g., MSI, EXE) or build new packages from scratch. These tools vary in complexity, features, and target audience, catering to scenarios from simple repackaging to advanced enterprise workflows. Below, we explore the primary MSIX packaging tools, their capabilities, and how they fit into modern desktop application management.

---

1. MSIX Packaging Tool

- **Overview**: Microsoft's official, free tool for creating and editing MSIX packages, available via the Microsoft Store.
- **Key Features**:
    - Converts MSI, EXE, App-V, and script-based installers into MSIX packages.
    - Supports both GUI and command-line interfaces for automation.
    - Captures app installations in real time using a clean machine or VM.
    - Allows customization of package metadata (e.g., display name, version).
    - Signs packages with a certificate for deployment security.
- **How It Works**:
    - Install the tool on a Windows 10/11 device.
    - Choose "Create new package" or "Convert existing package."
    - For conversion, point to an installer (e.g., `setup.msi`) and specify output details.
    - For capture, run the installer on a monitored system, then save the resulting MSIX.
- **Strengths**:
    - Free and natively supported by Microsoft.
    - Simple for basic conversions and small-scale use.

    - Integrates with Intune and SCCM for deployment.
- **Weaknesses**:
    - Limited advanced editing (e.g., no deep dependency management).
    - Requires a clean environment for accurate capture, adding setup overhead.
    - Manual process for complex apps with custom install logic.
- **Use Case**: Ideal for IT admins or small teams repackaging straightforward Win32 apps for Intune or Microsoft Store deployment.
- **Availability**: Microsoft Store or GitHub (command-line version).

---

2. Visual Studio (with MSIX Project Templates)

- **Overview**: Microsoft's flagship IDE, which includes built-in support for creating MSIX packages as part of app development workflows.
- **Key Features**:
    - Native MSIX project templates for UWP and Windows App SDK (WinUI 3) apps.
    - Configures package manifests (e.g., `AppxManifest.xml`) via a visual editor.
    - Builds, signs, and tests MSIX packages directly from the IDE.
    - Supports integration with Azure DevOps for CI/CD pipelines.
- **How It Works**:
    - Create a new UWP or Windows App SDK project in Visual Studio.
    - Edit the package manifest to define app capabilities, icons, and dependencies.
    - Build the project, generating an MSIX package in the output folder.
    - Sign the package with a certificate (self-signed or trusted).
- **Strengths**:
    - Seamless for developers building new apps from scratch.
    - Rich debugging and testing tools within the IDE.
    - Supports modern frameworks like .NET and WinUI.

- **Weaknesses**:
    - Not designed for repackaging existing apps (e.g., legacy EXEs).
    - Requires coding knowledge, less suited for IT admins.
    - License cost for Visual Studio Professional/Enterprise editions.
- **Use Case**: Best for developers creating new Windows apps or modernizing existing codebases with MSIX as the target format.
- **Availability**: Visual Studio 2022 (Community, Professional, or Enterprise editions).

---

3. Advanced Installer (MSIX Extension)

- **Overview**: A third-party tool with a dedicated MSIX module, traditionally known for MSI packaging but expanded to support modern formats.
- **Key Features**:
    - Converts MSI, EXE, and App-V packages to MSIX with a user-friendly GUI.
    - Advanced editing of package manifests, dependencies, and file associations.
    - Supports repackaging via installation monitoring or direct import.
    - Integrates with Intune and SCCM for deployment workflows.
    - Offers MSIX modification without repackaging (e.g., updating certificates).
- **How It Works**:
    - Open Advanced Installer and select the MSIX project type.
    - Import an existing installer or capture a new installation.
    - Customize settings (e.g., shortcuts, capabilities) in the visual editor.
    - Build and sign the MSIX package.
- **Strengths**:
    - Intuitive interface with more control than the MSIX Packaging Tool.
    - Handles complex apps with dependencies or custom scripts.
    - Bridges legacy (MSI) and modern (MSIX) workflows.
- **Weaknesses**:
    - Paid tool (requires Architect or Enterprise edition for full MSIX features).

    - Less lightweight than Microsoft's free offerings.
- **Use Case**: Enterprises needing a robust, all-in-one solution for repackaging legacy apps and managing large-scale MSIX deployments.
- **Availability**: Advanced Installer website (free trial available).

---

4. InstallShield (MSIX Support)

- **Overview**: A veteran packaging tool from Flexera, updated to include MSIX capabilities alongside its MSI and App-V heritage.
- **Key Features**:
    - Converts legacy installers (MSI, EXE) to MSIX packages.
    - Visual editor for package manifests and app properties.
    - Supports hybrid projects (e.g., MSI + MSIX outputs from one source).
    - Integrates with SCCM and modern deployment systems.
- **How It Works**:
    - Create a new MSIX project or import an existing installer.
    - Configure package details (e.g., version, publisher) in the UI.
    - Build the MSIX package and sign it with a certificate.
- **Strengths**:
    - Mature platform with extensive enterprise support.
    - Ideal for teams transitioning from MSI/App-V to MSIX.
    - Handles complex install logic and prerequisites.
- **Weaknesses**:
    - Expensive licensing model.
    - Overkill for simple apps or small teams.
- **Use Case**: Large organizations with existing InstallShield workflows modernizing to MSIX while supporting legacy deployment needs.
- **Availability**: Flexera website (paid, with trial option).

---

5. PowerShell and MSIX SDK

- **Overview**: A script-based approach using Microsoft's MSIX Software Development Kit (SDK) and PowerShell for advanced users or automation.
- **Key Features**:
  - Programmatic creation and modification of MSIX packages.
  - Includes tools like `MakeAppx.exe` (packs files into MSIX) and `SignTool.exe` (signs packages).
  - Fully customizable via scripting for batch processing.
  - Open-source components available via the MSIX SDK.
- **How It Works**:
  - Prepare app files and a manifest (e.g., `AppxManifest.xml`).
  - Use `MakeAppx.exe pack` to create an MSIX package from a directory.
  - Sign the package with `SignTool.exe` and a certificate.
  - Automate with PowerShell scripts for bulk operations.
- **Strengths**:
  - Free and highly flexible for custom workflows.
  - Enables CI/CD integration (e.g., GitHub Actions, Azure Pipelines).
  - No GUI dependency, ideal for server-based automation.
- **Weaknesses**:
  - Requires scripting expertise and manual manifest editing.
  - No built-in capture or conversion for legacy installers.
- **Use Case**: DevOps teams or advanced users automating MSIX packaging in CI/CD pipelines or bulk-processing scenarios.
- **Availability**: MSIX SDK on GitHub; PowerShell pre-installed on Windows.

Comparison Table

Tool	Best For	Cost	Ease of Use	Legacy Conversion	Automation
MSIX Packaging Tool	Simple repackaging	Free	Moderate	Yes	Limited
Visual Studio	New app development	Free/Paid	High (devs)	No	High
Advanced Installer	Enterprise repackaging	Paid	High	Yes	Moderate
InstallShield	Legacy-to-modern transition	Paid	High	Yes	Moderate
PowerShell/MSIX SDK	Automation & scripting	Free	Low	No	Very High

---

Key Considerations When Choosing a Tool

1. **App Type**:
    - New UWP/WinUI apps → Visual Studio.
    - Legacy Win32 apps → Advanced Installer or MSIX Packaging Tool.
2. **Team Skillset**:
    - Developers prefer Visual Studio or PowerShell; IT admins lean toward GUI tools like Advanced Installer.
3. **Scale**:
    - Small projects can use the free MSIX Packaging Tool; enterprises benefit from InstallShield or Advanced Installer's advanced features.

---

4. **Budget**:
    - Free options (MSIX Packaging Tool, PowerShell) suit cost-conscious teams; paid tools offer richer functionality.
5. **Deployment Target**:
    - Intune/AVD deployments pair well with any tool, but automation (PowerShell) shines for CI/CD.

Practical Workflow Example (Using MSIX Packaging Tool)

1. **Setup**: Install the MSIX Packaging Tool from the Microsoft Store.
2. **Convert an MSI**:
    - Launch the tool, select “Application package,” and browse to `app.msi`.
    - Specify an output path (e.g., `app.msix`).
    - Sign with a self-signed certificate or trusted CA.
3. **Deploy**: Upload `app.msix` to Intune or sideload on a test device.
4. **Test**: Verify app functionality and clean uninstallation.

Conclusion

MSIX packaging tools cater to a spectrum of needs, from the free and straightforward MSIX Packaging Tool to the developer-centric Visual Studio and enterprise-grade Advanced Installer or InstallShield. For automation enthusiasts, PowerShell with the MSIX SDK offers unmatched flexibility. Selecting the right tool depends on your app portfolio, team expertise, and deployment goals. As MSIX becomes central to Microsoft’s ecosystem (e.g., Intune, Azure Virtual Desktop), mastering these tools is key to modernizing desktop applications efficiently and effectively.

The MSIX conversion process transforms existing application installers—such as MSI, EXE, App-V, or even script-based setups—into the MSIX packaging format, enabling modern deployment, management, and compatibility with Windows 10 and 11. MSIX offers benefits like clean installs/uninstalls, digital signing, and integration with tools like Microsoft Intune and Azure Virtual Desktop. The conversion process varies depending on the source installer and complexity of the application, but it generally involves capturing or repackaging the app's files, dependencies, and configuration into an MSIX container. Below is a detailed explanation of the MSIX conversion process, focusing on the most common approach using the MSIX Packaging Tool, along with key steps, considerations, and best practices.

Overview of the MSIX Conversion Process

The goal is to take an application in its current form (e.g., an MSI file or legacy installer) and encapsulate it into an MSIX package—a single, signed file (`.msix`) that includes the app's binaries, dependencies, and metadata. The process can be manual (capturing an installation) or semi-automated (converting an existing package), depending on the tool and source. Microsoft's MSIX Packaging Tool is the primary utility for this, offering both GUI and command-line options.

Step-by-Step MSIX Conversion Process

Here's how to convert an application using the MSIX Packaging Tool, the most widely used free tool from Microsoft:

1. Prepare the Environment

- **Requirements**:
    - Windows 10 (version 1709 or later) or Windows 11.
    - MSIX Packaging Tool installed (available via the Microsoft Store).

    - A clean machine or virtual machine (VM) to avoid interference from existing software.
- **Why Clean?**: Pre-installed apps or conflicting registry entries can pollute the capture process, leading to an inaccurate MSIX package.
- **Setup**:
    - Use a VM (e.g., Hyper-V, VMware) with a fresh Windows install.
    - Take a snapshot/checkpoint for easy rollback after conversion.

2. Launch the MSIX Packaging Tool

- Open the tool from the Start menu.
- Select one of three options based on your scenario:
    - **Create application package**: Convert by capturing an installation (most common for EXE/script-based installers).
    - **Convert application package**: Directly convert an MSI or App-V package.
    - **Modify existing package**: Edit an existing MSIX (not a full conversion).

3. Capture or Convert the Application

- **Option A: Capture an Installation (EXE/Script-Based)**:
    1. **Start Monitoring**: Choose “Create application package” > “Create package on this computer” > Next.
    2. **Specify Installer**: Point to the installer (e.g., `setup.exe`) and provide command-line arguments if needed (e.g., `/silent`).
    3. **Run Installer**: The tool monitors file system and registry changes as the installer runs. Complete the installation process manually if it’s interactive.
    4. **Stop Monitoring**: After installation, click “Next” to finalize the capture.
- **Option B: Convert an MSI/App-V Package**:
    1. Choose “Convert application package” > Next.
    2. Browse to the MSI file (e.g., `app.msi`) or App-V package (`.appv`).
    3. The tool extracts files and metadata directly, skipping the capture step.

- **Output**: The tool generates a temporary package structure with files, registry entries, and a default manifest.

4. Customize the Package

- **Package Information**:
  - Enter details like *Name*, *Display Name*, *Publisher*, and *Version*. These appear in the Start menu and app metadata.
  - Example: Publisher format is `CN=YourCompanyName`.
- **Entry Points**:
  - Specify the executable (e.g., `app.exe`) that launches the app. For multiple entry points, add additional executables.
- **Dependencies**:
  - Manually include external dependencies (e.g., .NET Framework) if not captured. This may require adding files to the package folder.
- **Manifest Editing**:
  - Optionally edit the `AppxManifest.xml` file (e.g., to add capabilities like file associations or permissions). Use a text editor or the tool's basic editor.

5. Sign the Package

- **Why Sign?**: MSIX packages must be digitally signed for deployment via Intune, SCCM, or sideloading.
- **Process**:
  - Select "Create signed package" in the tool.
  - Use an existing certificate (e.g., from a trusted CA) or generate a self-signed certificate for testing:
    - Command: `New-SelfSignedCertificate -Type CodeSigningCert -Subject "CN=Test" -KeyUsage DigitalSignature -CertStoreLocation Cert:\CurrentUser\My`
  - Provide the certificate file (`.pfx`) and password if required.
- **Output**: A signed `.msix` file (e.g., `app.msix`).

6. Save and Test the Package

- **Save**: Choose an output location for the `.msix` file.
- **Test**:
    - Install manually: Double-click the `.msix` file on a test machine (requires sideloading enabled: Settings > Update & Security > For developers).
    - Deploy via Intune: Upload to the Intune admin center and assign to a test group.
    - Verify functionality, shortcuts, and uninstall behavior.

Key Considerations During Conversion

1. **Dependencies**:
    - If the app requires runtimes (e.g., Visual C++ Redistributables), ensure they're included or pre-installed on target devices.
    - MSIX doesn't inherently virtualize dependencies like App-V; they must be explicit in the package or system.
2. **Complex Installers**:
    - Apps with custom logic (e.g., registry tweaks, driver installs) may fail to convert cleanly. Manual adjustments or repackaging may be needed.
3. **Silent Installation**:
    - For automation, use installers with silent switches (e.g., `setup.exe /quiet`) during capture to avoid interactive prompts.
4. **File System and Registry**:
    - MSIX enforces strict rules (e.g., no writes to `C:\Windows`). Modify apps or use Package Support Framework (PSF) fixes for compatibility.
5. **Licensing**:
    - Ensure the app's license allows repackaging and redistribution in MSIX form.

Tools for Conversion

While the MSIX Packaging Tool is the standard, other tools can streamline or enhance the process:

- **Advanced Installer**: Offers a GUI for conversion with dependency management and manifest editing.
- **InstallShield**: Converts MSI/EXE to MSIX with enterprise-grade options.
- **PowerShell/MSIX SDK**: Automates conversion using `MakeAppx.exe` for scripted workflows.
    - Example: `MakeAppx pack /d "C:\AppFiles" /p "app.msix"`

Common Challenges and Solutions

1. **Challenge**: App fails post-conversion due to missing dependencies.
    - **Solution**: Add dependencies manually to the package or deploy them separately via Intune.
2. **Challenge**: Legacy app modifies restricted OS locations.
    - **Solution**: Apply the Package Support Framework (PSF) to redirect file/registry calls (e.g., using `Fixup` tools like PSF Launcher).
3. **Challenge**: Interactive installer requires user input.
    - **Solution**: Find silent install switches in the app's documentation or contact the vendor.

Example Workflow (Converting an MSI)

---

1. **Setup**: Launch MSIX Packaging Tool on a clean Windows 11 VM.
2. **Convert**: Select “Convert application package,” input `notepadplusplus.msi`.
3. **Customize**: Set Display Name to “Notepad++,” Publisher to `CN=NotepadPlus`.
4. **Sign**: Use a self-signed certificate (`notepad.pfx`).
5. **Output**: Save as `notepadplusplus.msix`.
6. **Test**: Sideload on a test device and confirm it launches correctly.

Benefits of MSIX Conversion

- Clean uninstalls with no residual files/registry entries.
- Seamless deployment via Intune, SCCM, or Microsoft Store.
- Signed packages enhance security and trust.
- Supports modern features like MSIX App Attach in Azure Virtual Desktop.

Conclusion

The MSIX conversion process bridges legacy applications with modern management by repackaging them into a standardized, efficient format. Using the MSIX Packaging Tool, the process is accessible yet requires careful preparation—especially for complex or legacy apps. By addressing dependencies, signing requirements, and compatibility issues, organizations can successfully convert their app portfolios, preparing them for cloud-based delivery and future Windows ecosystems. For advanced needs, tools like Advanced Installer or PowerShell offer additional flexibility, ensuring MSIX fits into any modernization strategy.

Below is a list of third-party vendors that offer application packaging solutions, along with the key features they provide that enhance the application packaging process. These vendors streamline, automate, and improve the efficiency, compatibility, and deployment of packaged applications, often going beyond native Microsoft tools like the MSIX Packaging Tool. This list focuses on widely recognized solutions as of March 23, 2025, based on their established offerings in the market.

---

1. Flexera (AdminStudio)

- **Overview**: A leading application packaging and readiness solution designed for enterprise-scale management of desktop applications.
- **Enhancement Features**:
    - **Automated Packaging**: Automatically downloads, tests, and repackages thousands of third-party installers, reducing manual effort.
    - **Compatibility Testing**: Identifies and mitigates deployment issues (e.g., OS compatibility, conflicts) before distribution.
    - **Broad Format Support**: Creates MSI, MSIX, App-V, and virtualized packages from a single workflow, enhancing flexibility.
    - **Integration**: Publishes packages directly to Microsoft SCCM, Intune, VMware Workspace ONE, and other systems management tools.
    - **Customization Wizards**: Nearly 1,000 wizards simplify complex app deployments, saving time on custom configurations.
    - **Application Readiness Automation**: Continuously updates packages for new software versions, reducing maintenance overhead.

---

2. Advanced Installer

- **Overview**: A versatile packaging tool known for its user-friendly interface, supporting both MSI and MSIX packaging with a focus on enterprise needs.
- **Enhancement Features**:
    - **Visual Repackaging**: Captures installations or imports MSI/EXE files with a GUI, simplifying conversion to MSIX.
    - **Dependency Management**: Automatically detects and includes dependencies, ensuring complete packages.
    - **MSIX Modification**: Edits existing MSIX packages (e.g., certificates, manifests) without repackaging, enhancing flexibility.
    - **PowerShell Automation**: Scripts repetitive tasks for bulk packaging, boosting efficiency.
    - **App Virtualization**: Offers lightweight virtualization options alongside MSIX, bridging legacy and modern deployment.
    - **Direct Deployment**: Integrates with Intune and SCCM for seamless package distribution.

---

3. InstallShield (Flexera)

- **Overview**: A long-standing packaging solution updated to support MSIX alongside traditional MSI and App-V formats, ideal for legacy-to-modern transitions.
- **Enhancement Features**:
    - **Hybrid Packaging**: Builds MSI, EXE, MSIX, and App-V packages from a single source, reducing rework.
    - **Advanced Manifest Editing**: Provides detailed control over MSIX manifests for custom capabilities and file associations.
    - **Silent Install Support**: Automates silent installations for complex apps, improving deployment consistency.

- **Legacy Conversion**: Converts older installers to MSIX with minimal manual intervention, easing modernization.
- **Enterprise Integration**: Deploys packages to SCCM and other management platforms with built-in support.
- **Validation Tools**: Ensures package integrity and compliance before deployment, reducing errors.

---

4. Patch My PC

- **Overview**: A third-party patch management and application packaging solution that integrates with Microsoft SCCM and Intune.
- **Enhancement Features**:
  - **Automated App Packaging**: Creates and updates application packages for initial deployment, not just patching, saving hours of manual work.
  - **Extensive Catalog**: Supports over 8,100 customers with a library of pre-packaged third-party apps, reducing custom packaging needs.
  - **Silent Deployment**: Configures silent install commands automatically for supported apps, enhancing automation.
  - **Cloud-Only Option**: Offers Patch My PC Cloud for Intune, enabling packaging without on-premises infrastructure.
  - **Custom App Support**: Allows users to add and deploy custom apps using the same workflow as its catalog, increasing versatility.
  - **Real-Time Updates**: Monitors vendor releases and refreshes packages monthly, ensuring up-to-date deployments.

---

5. Raynet (RayPack Studio)

- **Overview**: A comprehensive packaging suite that supports multiple formats, including MSIX, MSI, and App-V, with a focus on automation and compatibility.
- **Enhancement Features**:
    - **Bulk Conversion**: Automates conversion of existing packages (MSI, App-V) to MSIX using virtual environments (e.g., VMware, Hyper-V).
    - **Pre-Conversion Screening**: Uses RayQC Advanced to test app compatibility with Windows 10/11 editions, minimizing post-deployment issues.
    - **Collision Detection**: Identifies potential conflicts between apps or OS versions during packaging, enhancing stability.
    - **Unified Workflow**: Creates MSIX, MSI, patches (MSP), and transforms (MST) from one platform, streamlining processes.
    - **Customizable Templates**: Speeds up packaging with reusable configurations for common app types.
    - **Enterprise Deployment**: Integrates with SCCM and other tools for large-scale rollouts.

---

6. Liquit (Setup Commander)

- **Overview**: Part of Liquit's Release & Patch Management suite, this tool simplifies repackaging legacy setups into modern formats like MSIX.
- **Enhancement Features**:
    - **Repackaging Wizard**: Wraps legacy setups into MSIX or MSIX App Attach (VHD/CIM) formats with minimal effort.
    - **Setup Store**: Provides a library of downloadable, customizable vendor setups for quick repackaging, saving sourcing time.
    - **Automation**: Fully automates packaging and validation, reducing manual steps.

- **MSIX App Attach Support**: Converts packages for use in Windows Virtual Desktop or multi-session environments, enhancing cloud compatibility.
- **Integration**: Connects with enterprise management solutions for seamless deployment.
- **Speed**: Significantly accelerates the repackaging process compared to manual methods.

---

7. PACE Suite (PACE Packager Hub)

- **Overview**: A modern packaging solution focused on flexibility and support for multiple packaging formats, including MSIX.
- **Enhancement Features**:
  - **Universal Packaging**: Supports MSIX, MSI, App-V, and ThinApp from a single tool, offering format versatility.
  - **Snapshot-Free Capture**: Uses PackageAware technology to capture setups without a clean VM, simplifying the process.
  - **Scripting Support**: Enhances automation with scripting options for complex installers.
  - **Quality Assurance**: Validates packages against Windows standards, reducing deployment risks.
  - **Collaboration Tools**: Manages team workflows for large-scale packaging projects, improving coordination.
  - **Direct Publishing**: Deploys packages to SCCM or other systems with built-in connectors.

Key Enhancements Across Vendors

These third-party solutions enhance the application packaging process in several ways compared to basic tools like the MSIX Packaging Tool:

- **Automation**: Reduce manual effort with bulk processing, silent install detection, and update scheduling.
- **Compatibility**: Address legacy app challenges with advanced dependency management and conflict resolution.
- **Flexibility**: Support multiple formats (MSIX, MSI, App-V) and deployment targets (Intune, SCCM, VDI).
- **Efficiency**: Speed up packaging with pre-built libraries, wizards, and reusable templates.
- **Enterprise Scale**: Enable large-scale deployments with integration into existing management systems and robust testing features.

Conclusion

These vendors—Flexera (AdminStudio and InstallShield), Advanced Installer, Patch My PC, Raynet, Liquit, and PACE Suite—offer powerful enhancements to the application packaging process, making them valuable for organizations modernizing desktop applications. The choice depends on specific needs: Patch My PC excels in automation and patching, Advanced Installer and InstallShield offer robust legacy support, while Raynet and Liquit focus on enterprise-scale flexibility and cloud readiness. Each tool goes beyond Microsoft's native offerings by addressing real-world complexities like legacy compatibility, scalability, and deployment efficiency, ensuring a smoother transition to modern formats like MSIX.

Below is a comparison of third-party vendors offering MSIX packaging solutions—Flexera (AdminStudio and InstallShield), Advanced Installer, Patch My PC, Raynet (RayPack Studio), Liquit (Setup Commander), and PACE Suite (PACE Packager Hub). These vendors enhance the MSIX packaging process beyond Microsoft's native MSIX Packaging Tool by providing advanced features tailored to enterprise needs, legacy app support, automation, and integration with modern deployment systems. This comparison evaluates their offerings across key criteria relevant to application packaging as of March 23, 2025.

Comparison Criteria

1. **Core MSIX Features**: Ability to create, convert, and edit MSIX packages.
2. **Legacy App Support**: Handling of MSI, EXE, App-V, and other older formats.
3. **Automation**: Tools for bulk processing, scripting, or automated updates.
4. **Enterprise Integration**: Compatibility with SCCM, Intune, and other management platforms.
5. **Ease of Use**: User interface and workflow simplicity.
6. **Additional Features**: Unique capabilities enhancing the packaging process.
7. **Cost**: Pricing model and value proposition.

---

1. Flexera (AdminStudio)

- **Core MSIX Features**: Full MSIX creation, conversion from MSI/App-V/EXE, and manifest editing.
- **Legacy App Support**: Strong support for MSI, App-V, and EXE with automated repackaging and conflict detection.
- **Automation**: High—downloads, tests, and repackages thousands of apps; supports CI/CD workflows.

- **Enterprise Integration**: Direct publishing to SCCM, Intune, VMware Workspace ONE, and more.
- **Ease of Use**: Moderate—GUI with wizards, but complex for beginners due to extensive options.
- **Additional Features**: Compatibility testing, nearly 1,000 app-specific wizards, continuous update automation.
- **Cost**: High—enterprise-grade pricing, typically subscription-based (e.g., $5,000+ annually per user).
- **Best For**: Large enterprises needing comprehensive app readiness and deployment automation.

---

2. Flexera (InstallShield)

- **Core MSIX Features**: MSIX creation and conversion from MSI/EXE, with detailed manifest customization.
- **Legacy App Support**: Excellent—converts legacy installers to MSIX, supports hybrid MSI+MSIX projects.
- **Automation**: Moderate—silent install automation, but less extensive than AdminStudio.
- **Enterprise Integration**: Built-in support for SCCM and other deployment tools.
- **Ease of Use**: High—intuitive GUI with a long history of user familiarity.
- **Additional Features**: Hybrid packaging (MSI/MSIX from one source), validation tools for compliance.
- **Cost**: High—standalone license (~$2,000-$4,000) or bundled with AdminStudio.
- **Best For**: Teams transitioning from MSI/App-V to MSIX with existing InstallShield expertise.

---

3. Advanced Installer

- **Core MSIX Features**: Converts MSI/EXE/App-V to MSIX, edits manifests, and supports MSIX modification.
- **Legacy App Support**: Robust—visual repackaging and dependency detection for legacy apps.
- **Automation**: Moderate—PowerShell scripting for bulk tasks, but primarily GUI-driven.
- **Enterprise Integration**: Direct deployment to Intune and SCCM.
- **Ease of Use**: High—user-friendly interface with drag-and-drop features.
- **Additional Features**: Lightweight virtualization option, dependency auto-inclusion, MSIX editing without repackaging.
- **Cost**: Moderate—starts at ~$500/year (Professional) to ~$2,500/year (Enterprise with MSIX features).
- **Best For**: Mid-sized teams or enterprises needing an affordable, versatile packaging solution.

---

4. Patch My PC

- **Core MSIX Features**: Creates MSIX packages for custom apps, though focused on pre-packaged catalog deployment.
- **Legacy App Support**: Limited—primarily enhances existing installers rather than full repackaging.
- **Automation**: Very High—automates packaging, updates, and deployment for over 8,100 pre-packaged apps.
- **Enterprise Integration**: Seamless with SCCM and Intune; offers a cloud-only Intune option.

- **Ease of Use**: High—simplified workflow for catalog apps, moderate for custom packaging.
- **Additional Features**: Extensive pre-packaged app library, monthly update refreshes, silent install automation.
- **Cost**: Moderate—subscription-based (~$1-$2 per device/year), cost-effective for large deployments.
- **Best For**: Organizations prioritizing automation and third-party app updates over custom packaging.

---

5. Raynet (RayPack Studio)

- **Core MSIX Features**: Full MSIX creation, conversion from MSI/App-V, and manifest customization.
- **Legacy App Support**: Strong—bulk conversion with virtual environment support (e.g., VMware, Hyper-V).
- **Automation**: High—bulk processing and scripting for large-scale packaging.
- **Enterprise Integration**: Integrates with SCCM and other enterprise tools.
- **Ease of Use**: Moderate—GUI-driven but requires technical knowledge for advanced features.
- **Additional Features**: Collision detection, pre-conversion compatibility screening (RayQC), reusable templates.
- **Cost**: High—enterprise pricing, typically quote-based (~$3,000+ annually per user).
- **Best For**: Enterprises needing detailed compatibility analysis and bulk MSIX conversions.

---

6. Liquit (Setup Commander)

- **Core MSIX Features**: Converts legacy setups to MSIX and MSIX App Attach (VHD/CIM) formats.
- **Legacy App Support**: Excellent—wraps complex setups into MSIX with minimal effort.
- **Automation**: High—fully automated packaging and validation processes.
- **Enterprise Integration**: Connects with Intune, SCCM, and cloud platforms.
- **Ease of Use**: High—repackaging wizard and Setup Store simplify workflows.
- **Additional Features**: Setup Store (vendor setup library), MSIX App Attach support for VDI, rapid processing.
- **Cost**: Moderate to High—subscription-based, varies by scale (e.g., ~$1,000-$3,000/year).
- **Best For**: Teams modernizing apps for cloud/VDI environments like Azure Virtual Desktop.

---

7. PACE Suite (PACE Packager Hub)

- **Core MSIX Features**: Creates and converts to MSIX, with manifest editing capabilities.
- **Legacy App Support**: Strong—supports MSI, App-V, ThinApp, and EXE conversions.
- **Automation**: High—scripting and PackageAware (snapshot-free capture) for efficiency.
- **Enterprise Integration**: Direct publishing to SCCM and other systems.
- **Ease of Use**: Moderate—GUI with collaboration tools, but technical setup required.
- **Additional Features**: Snapshot-free capture, team collaboration workflows, package validation.
- **Cost**: Moderate—starts at ~$1,200/year per user, scalable for teams.

- **Best For**: Collaborative teams needing flexible, multi-format packaging without VM overhead.

Comparison Table

Vendor	Core MSIX	Legacy Support	Automation	Integration	Ease of Use	Unique Features	Cost
AdminStudio	Excellent	Excellent	Very High	Excellent	Moderate	Compatibility testing, wizards	High
InstallShield	Excellent	Excellent	Moderate	Excellent	High	Hybrid packaging, validation	High
Advanced Installer	Excellent	Excellent	Moderate	Excellent	High	Dependency mgmt, virtualization	Moderate
Patch My PC	Good	Limited	Very High	Excellent	High	App catalog, update automation	Moderate
RayPack Studio	Excellent	Excellent	High	Excellent	Moderate	Collision detection, screening	High
Liquit	Excellent	Excellent	High	Excellent	High	Setup Store, MSIX App Attach	Moderate-High

---

PACE Suite	Excellent	Excellent	High	Excellent	Moderate	Snapshot-free, collaboration	Moderate

---

Analysis and Recommendations

1. **Best for Enterprise Scale**:
    - **AdminStudio** excels with automation, testing, and broad integration, ideal for large organizations with diverse app portfolios.
    - **RayPack Studio** offers similar scale with a focus on compatibility and bulk processing.
2. **Best for Legacy Transitions**:
    - **InstallShield** and **Advanced Installer** shine with robust legacy support and intuitive workflows, perfect for teams modernizing MSI/App-V apps.
3. **Best for Automation**:
    - **Patch My PC** leads with its pre-packaged library and update automation, though it's less flexible for custom apps.
    - **PACE Suite** offers strong automation without VM dependency.
4. **Best for Cloud/VDI**:
    - **Liquit** stands out with MSIX App Attach support and rapid cloud-ready packaging.
5. **Best Value**:
    - **Advanced Installer** and **PACE Suite** balance cost and features for mid-sized teams.
    - **Patch My PC** is cost-effective for automated third-party app management.

Conclusion

Each MSIX vendor enhances the packaging process differently: Flexera's AdminStudio and InstallShield target enterprise complexity, Advanced Installer offers affordability and ease, Patch My PC prioritizes automation, Raynet focuses on compatibility, Liquit excels in cloud/VDI scenarios, and PACE Suite provides flexibility. The choice depends on your organization's scale, legacy app needs, deployment targets (e.g., Intune, SCCM), and budget. For a comprehensive solution, AdminStudio leads; for cost-effective versatility, Advanced Installer or PACE Suite are strong contenders. Evaluate your app catalog and infrastructure to select the vendor that best aligns with your modernization goals.

SCCM

Migrating applications from Microsoft System Center Configuration Manager (SCCM, now part of Microsoft Configuration Manager) to Microsoft Intune is a key step in modernizing desktop application management, enabling cloud-based control, and supporting hybrid or remote workforces. This process involves planning, preparation, conversion, deployment, and validation to ensure a seamless transition with minimal disruption to end users. Below is a step-by-step guide to migrating applications from SCCM to Intune.

---

1. Planning and Assessment

Before migrating, assess your current SCCM environment to understand the scope and requirements:

- **Inventory Applications**: Identify all applications managed in SCCM, including their deployment types (e.g., MSI, EXE, scripts), dependencies, detection methods, and

---

installation requirements. Use SCCM reports or tools like PowerShell scripts to export this data.

- **Prioritize Applications**: Categorize applications by criticality, usage frequency, and complexity. Start with widely used, business-critical apps to ensure continuity.
- **Evaluate Intune Readiness**: Check if applications are compatible with Intune's Win32 app model, which supports traditional Windows applications. Some legacy apps may require repackaging or updates.
- **Define Goals**: Decide whether to fully replace SCCM with Intune or use co-management, where SCCM and Intune work together. Co-management allows a gradual shift of workloads.

---

2. Set Up the Environment

Prepare your infrastructure to support the migration:

- **Enable Co-Management (Optional)**: If transitioning gradually, configure co-management in SCCM. In the SCCM console, go to *Administration* > *Cloud Services* > *Cloud Attach*, and enable co-management. Assign the "Client Apps" workload to Intune or a pilot group.
- **Configure Intune**: Ensure Intune is set up with an active Microsoft Entra ID (formerly Azure AD) tenant, proper licensing (e.g., Microsoft 365 or EMS), and the MDM authority set to Intune.
- **Enroll Devices**: Enroll Windows devices in Intune via Microsoft Entra ID Join or Hybrid Join. Devices must be co-managed or fully Intune-managed to receive migrated apps.

---

3. Convert Applications to Intune Format

SCCM applications need to be converted into a format Intune can deploy, primarily the `.intunewin` format for Win32 apps:

- **Gather Installation Files**: Locate the source files, scripts, and dependencies for each SCCM application from its content distribution points.
- **Repackage for Intune**: Use the *Microsoft Win32 Content Prep Tool* to convert installation files into `.intunewin` packages:
    1. Download the tool from Microsoft's GitHub repository.
    2. Run the tool with commands like:

       ```
       IntuneWinAppUtil.exe -c <source_folder> -s <installer_file> -o <output_folder>
       ```
    3. This creates a single `.intunewin` file containing all necessary components.
- **Preserve Detection Rules**: In SCCM, detection methods (e.g., registry keys, file versions) ensure an app is installed correctly. Manually note these rules, as they'll need to be recreated in Intune.
- **Handle Dependencies**: If an app relies on other software (e.g., .NET Framework), package those dependencies separately or use Intune's dependency feature.

---

4. Deploy Applications in Intune

Upload and configure the converted applications in the Intune admin center:

- **Add the App**:
    1. Log in to the Microsoft Intune admin center (`endpoint.microsoft.com`).
    2. Navigate to *Apps* > *All Apps* > *Add*, and select "Windows app (Win32)" as the app type.
    3. Upload the `.intunewin` file.
- **Configure App Information**: Enter details like name, description, publisher, and category.

- **Set Install Command**: Specify the installation command (e.g., `setup.exe /silent`) and uninstall command, mirroring SCCM behavior.
- **Define Detection Rules**: Recreate SCCM detection methods in Intune (e.g., file exists, registry key value) to verify installation success.
- **Assign the App**: Assign the app to user or device groups, choosing “Required” (mandatory install) or “Available” (optional via Company Portal).
- **Test Deployment**: Deploy to a pilot group first to validate functionality.

---

5. Validation and Testing

Ensure the migrated applications work as expected:

- **Test on Pilot Devices**: Verify installation, functionality, and user experience on a small set of co-managed or Intune-managed devices.
- **Compare with SCCM**: Confirm that the Intune-deployed app behaves identically to its SCCM counterpart, checking features, settings, and dependencies.
- **Monitor Logs**: Check the Intune Management Extension logs (`C:\ProgramData\Microsoft\IntuneManagementExtension\Logs`) on devices for errors or issues.

---

6. Full Migration and Cleanup

Once validated, roll out the migration and retire SCCM components:

- **Scale Deployment**: Assign apps to broader user or device groups in Intune.
- **Retire SCCM Deployments**: Remove the application deployments from SCCM for migrated devices to avoid conflicts, especially if using co-management.

- **Uninstall SCCM Client (Optional)**: For a full Intune transition, uninstall the SCCM client from devices after confirming Intune management is sufficient.
- **Document Changes**: Record the migration process, including app configurations and lessons learned, for future reference.

Tools and Tips

- **Automation Tools**: Use community tools like the *Win32App Migration Tool* or commercial solutions (e.g., SoftwareCentral, Rimo3) to automate conversion and deployment.
- **Handle Legacy Apps**: For apps incompatible with Intune (e.g., requiring complex task sequences), consider repackaging with tools like Advanced Installer or maintaining them in SCCM under co-management.
- **User Communication**: Inform users about the transition, especially if self-service via the Company Portal is enabled.

Benefits and Considerations

- **Benefits**: Intune offers cloud-based management, support for remote devices, and integration with Microsoft 365 security features like Conditional Access.
- **Challenges**: Intune lacks SCCM's advanced features like task sequences or bare-metal OS deployment, so plan accordingly if these are critical.

By following this structured approach, you can efficiently migrate applications from SCCM to Intune, modernizing your application management while maintaining reliability and user satisfaction.
