Leveraging OS virtualization for your repackaging and testing infrastructure

Victor Ciura
Senior Software Engineer
Advanced Installer

Session code: 123

Agenda

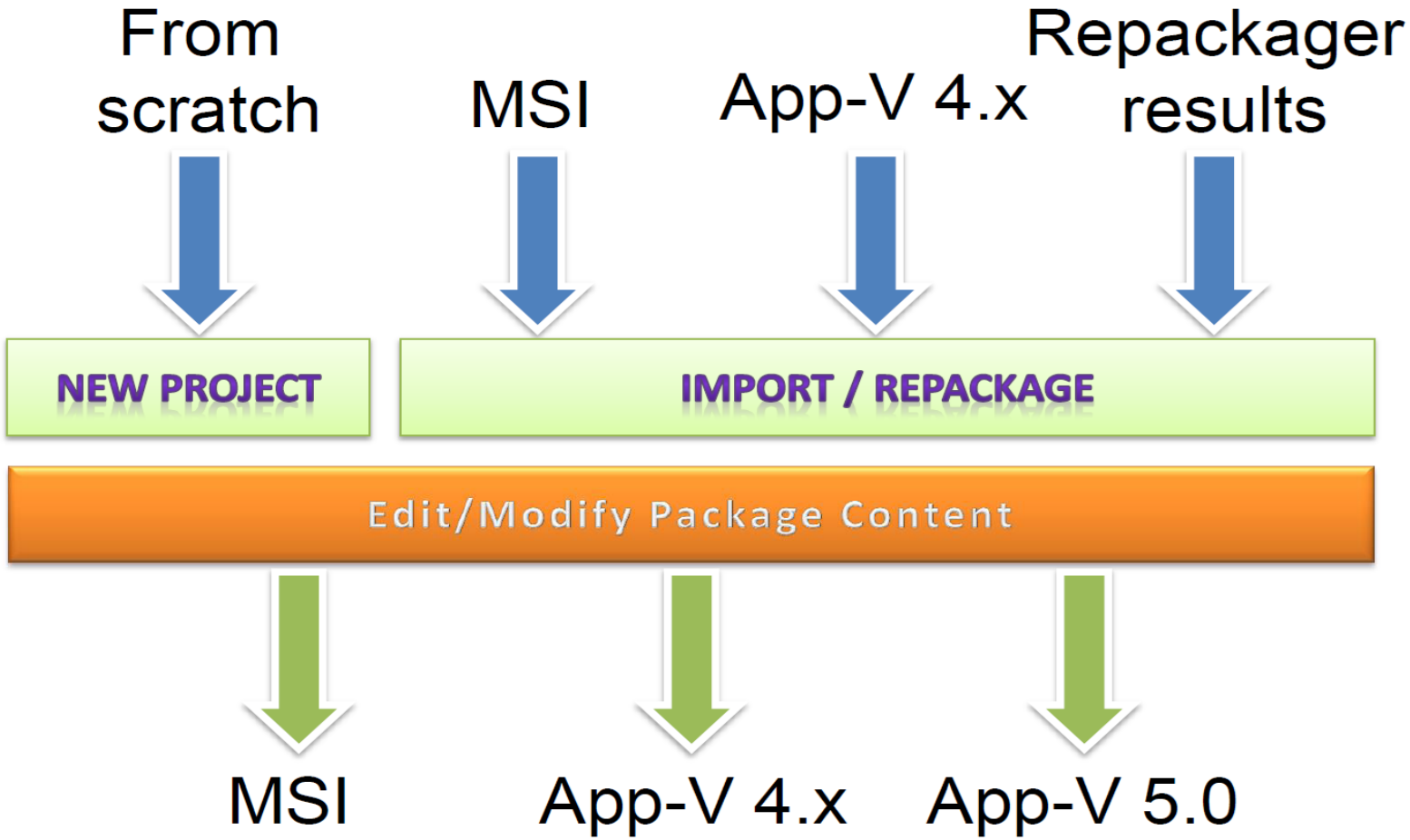
- Advanced Installer overview
- Packaging workflow
- Repackager overview
- Repackaging challenges & pitfalls
- Repackaging in a VM
- Repackager Automation/CLI
- DEMO: repackage in VM
- Package testing in a VM
- VM Launcher Automation/CLI
- DEMO: test package in VM
- Roadmap
- Q & A

Advanced Installer

- Comprehensive packager IDE
- Built on industry standards
  - no proprietary engines
- High-level, easy to use UI
  - No scripting languages required
  - Automation friendly (MSBuild, Ant, PowerShell)
- A lot of industry experience (since 2003)
- Excellent technical support team
- Agile release cycle (guided by strong customer feedback)

Advanced Installer

Packaging Workflow



Repackaging Overview

Why ?

- Converting an installation kit to another format: EXE => MSI => App-V
- Migrating from another packaging tool
- Coalescing multiple packages and updates
- Customization of application installation and post-installation options

Repackaging Tool

- Performs an instrumented install of the source package
- Records the changes on target machine OS
- A smart scanning algorithm detects and preserves high-level constructs like:
  - Services
  - Drivers
  - File Associations
  - Environment Variables
  - Assemblies
  - COM (classes, interfaces, typelibs)
  - Scheduled Tasks
  - Windows Firewall Settings
  - Control Panel Applets
- Builds new package(s)

Repackaging challenges & pitfalls

➢ Results Accuracy: ***missing resources***

We need a clean machine (fresh OS install).

Solution:

- repackage in a VM
- by leveraging VM snapshots, you get a non-intrusive, automated and reproducible process

Repackaging challenges & pitfalls

➢ Results Accuracy: ***false positives***

Solution:

- very fined tuned **filters**, customizable from a friendly GUI
- integrated automatic system **noise scan**
- file & registry *real-time monitoring* for triaging system changes on a **per-process** basis

Repackaging challenges & pitfalls

➢ ***Information Loss***

Solution:

Post-processing detected system changes into high-level package data like:

- Services
- Drivers
- File Associations
- Environment Variables
- Assemblies
- COM (classes, interfaces, typelibs)
- Scheduled Tasks
- Windows Firewall Settings
- Control Panel Applets

Repackaging challenges & pitfalls

➢ ***Unsolved Problems***

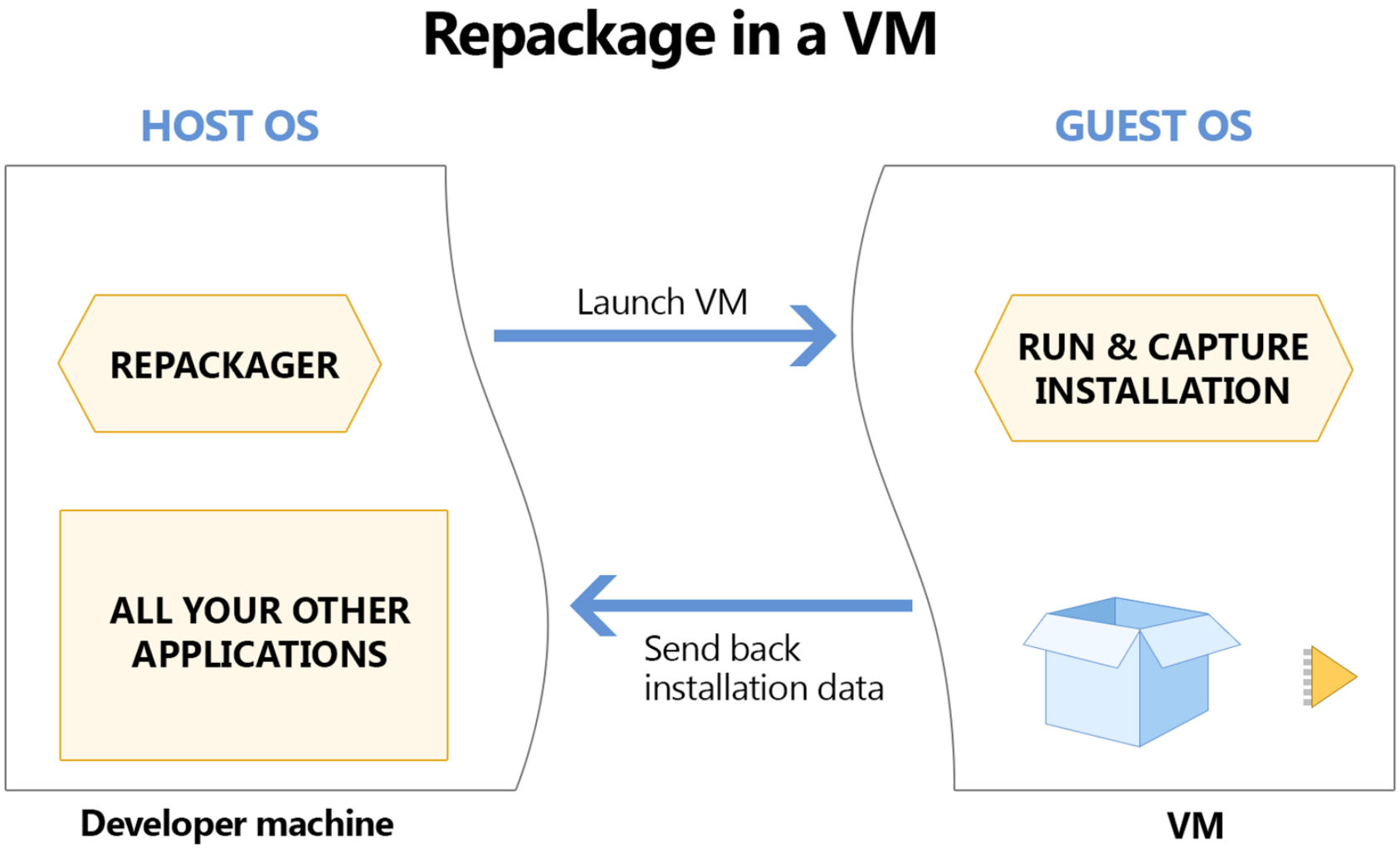
- original package organization into features is lost (also, component dependencies)
- package custom actions (just side-effects)
- resources installed conditionally for each OS
- installer logic driven by custom input, or detected environment settings
- localized resources are captured just for the current language

Why Repackage in VMs ?

- Deploy the application in a controlled environment (clean or preconfigured OS image)
- Use legacy OS (Windows XP) for older applications
- Faster system scans
- Lower false positives due to system background activity
- Repeatability by leveraging VM snapshots
- Parallelize repackaging operations using multiple VM images

http://www.advancedinstaller.com/user-guide/repackaging-VMs.html

Advanced Installer



Automation (CLI)

Powerful Repackager CLI:

/ProductName	- The name of the repackaged product.
/ProductVersion	- The repackaged product's version.
/ProductCompany	- The repackaged product's company name.
/SetupPath	- The file path of the application setup which will be repackaged.
/SetupCmdLine	- Command line which will be passed to the setup chosen to be repackaged.
/SaveSnapshots	- Save system snapshots on disk after installation capture is completed.
/PreSnapshot	- Specify a file path to an existing saved system snapshot.
/NoiseScan	- Perform a system noise scan without installing a package.
/NoiseFile	- Specify a file path to an existing saved system noise recording.
/OutputPath	- The output path for the repackaged solution (resources and repackaging results).
/Profile	- Specify a custom profile to use for repackaging.
/ListProfiles	- Displays a list with the available profiles.
/PromptAfterPreScan	- Prompt and wait after first system scan.
/PromptBeforePostScan	- Prompt and wait before second system scan.
/UiAutomation	- Automate package installation by instrumenting the UI.
/NoUI	- Repackager will run in console mode (no wizard).

Automation (Windows PowerShell)

```
# repackage a product in a VMware Workstation VM (optional: load a previous snapshot)
$pi = Start-Process -Wait -PassThru -FilePath "C:\Program Files (x86)\Caphyon\Advanced Installer 11.6\bin\x64\Repackager.exe"
 -ArgumentList `
    "/NoUI /SetupPath `"D:\Test\ProductName-SetupFiles\ProductName.msi`" /VmPath `"C:\VMware VMs\Windows XP\Windows XP.vmx`"
    /VmLoadSnapshot `"Initial\SnapshotA`" /PromptBeforePostScan /VmUser caphyon /VmPassword pass "
Write-Output "Repackage in VM returned: "  $pi.ExitCode

# repackage a product in a Hyper-V VM (optional: load a previous snapshot)
$pi = Start-Process -Wait -PassThru -FilePath "C:\Program Files (x86)\Caphyon\Advanced Installer 11.6\bin\x64\Repackager.exe"
 -ArgumentList `
    "/NoUI /SetupPath `"D:\Test\ProductName-SetupFiles\ProductName.msi`" /VmPath 598409EA-9A76-4A3F-92F7-F3D42638FACD
    /VmLoadSnapshot `"Initial\SnapshotA`" /PromptBeforePostScan "
Write-Output "Repackage in VM returned: "  $pi.ExitCode

# repackage a product in a VMware Workstation VM (specify a disk location for the resources and repackaging results)
$pi = Start-Process -Wait -PassThru -FilePath "C:\Program Files (x86)\Caphyon\Advanced Installer 11.6\bin\x64\Repackager.exe"
 -ArgumentList `
    "/NoUI /SetupPath `"D:\Test\ProductName-SetupFiles\ProductName.msi`" /VmPath `"C:\VMware VMs\Windows XP\Windows XP.vmx`"
    /VmLoadSnapshot `"Initial\SnapshotA`" /OutputPath `"D:\Test\Installation Captures`" /VmUser caphyon /VmPassword pass "
Write-Output "Repackage in VM returned: "  $pi.ExitCode

# repackage a product in a Hyper-V VM (optional: use a custom Repackager Profile)
$pi = Start-Process -Wait -PassThru -FilePath "C:\Program Files (x86)\Caphyon\Advanced Installer 11.6\bin\x64\Repackager.exe"
 -ArgumentList `
    "/NoUI /SetupPath `"D:\Test\ProductName-SetupFiles\ProductName.msi`" /VmPath 598409EA-9A76-4A3F-92F7-F3D42638FACD
    /VmLoadSnapshot `"Initial\SnapshotA`" /Profile `"My Custom Filters`" "
Write-Output "Repackage in VM returned: "  $pi.ExitCode
```

DEMO

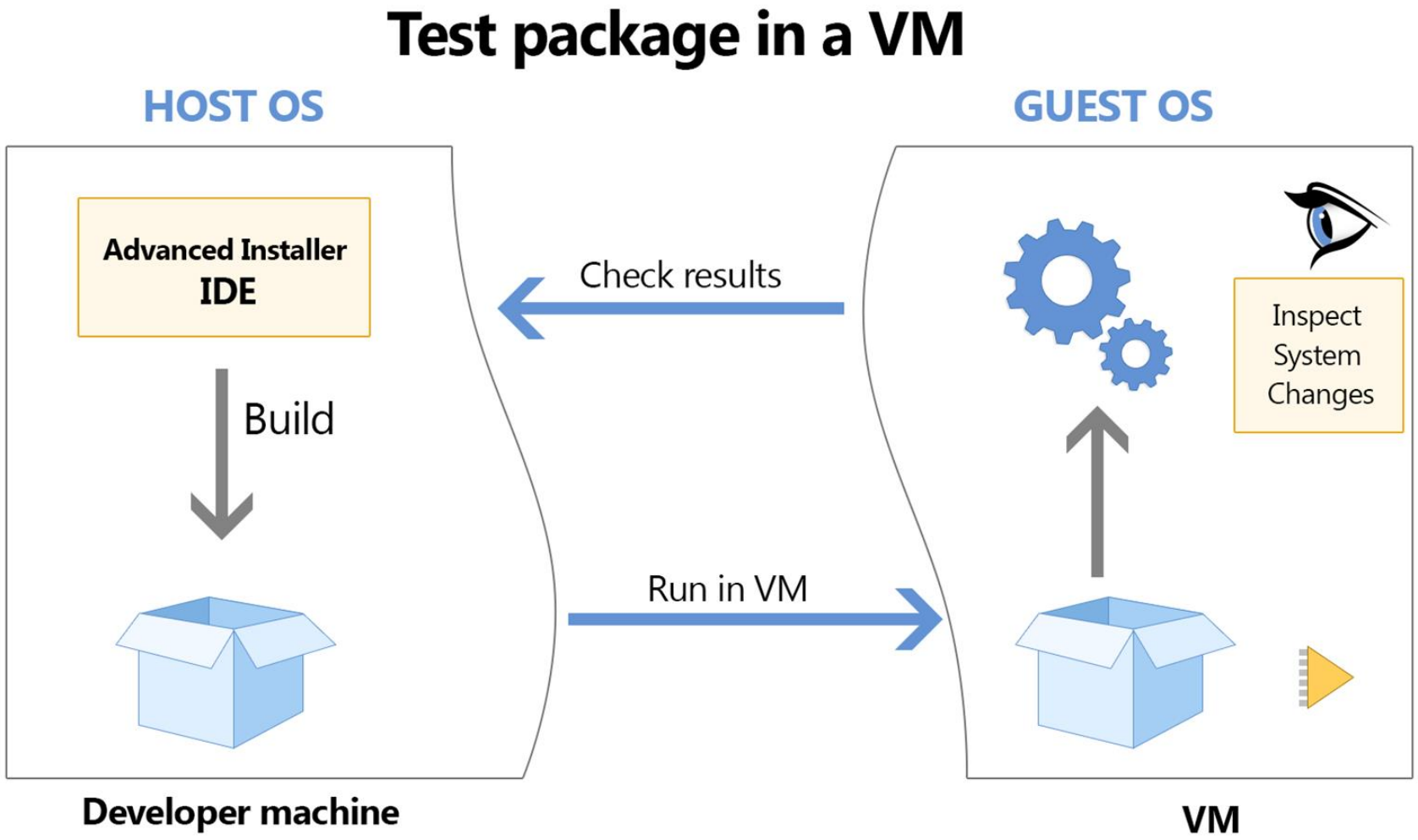
Repackage in a virtual machine.

Package testing in VMs

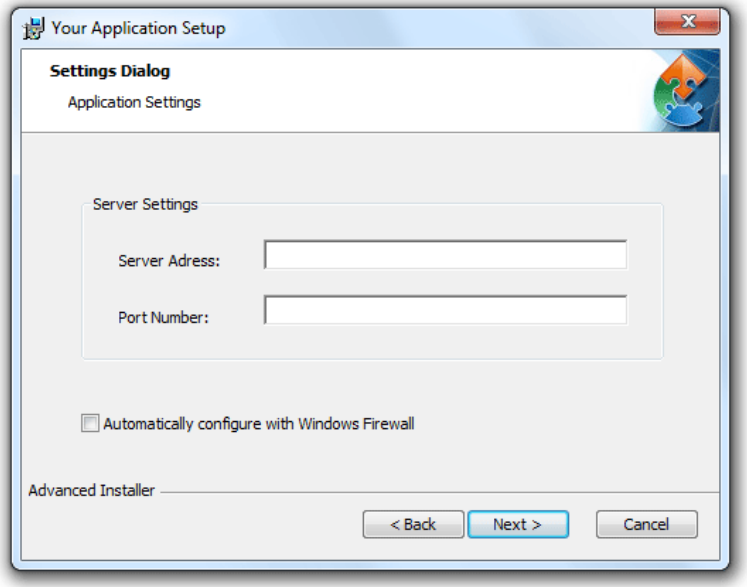
Why ?

- Quickly test your package on multiple OS versions (that you plan to support)
- Deploy the application in a controlled environment
  (clean or preconfigured OS image)
- Repeatability by leveraging VM snapshots
- You won't wreck your development machine when testing half-done installers
- Integrated within Advanced Installer IDE & build workflow (one-click)

Package testing workflow



Automation (GUI)



Automates the package installation by invoking UI controls in order to walk-through the installation steps.

Will automatically scroll text controls and invoke buttons like: **Agree**, **OK**, **Next**, **Continue**, **Install**, **Finish**, **Close**.

Tips:

- Disable automatic updates and background programs that might display popups during the automated install.
- You should not activate other application windows until the installation process has ended.
- The automation tool will attempt to close any license agreement dialog in the package UI, so make sure you accept the license terms of the repackaged application.

Automation (CLI)

Powerful VM Launcher CLI:

`/SetupPath`	- The file path of the application setup which will be tested.
`/SetupCmdLine`	- Command line which will be passed to the setup chosen to be tested.
`/UiAutomation`	- Automate package installation by instrumenting the UI.
`/NoUI`	- VM Launcher will run in console mode.
`/Profile`	- Specify a VM profile to use for testing.
`/EditProfiles`	- Display UI for configuring VM profiles.
`/VmPath`	- The path to the VM configuration file (VMware: *.vmx), or VM identifier (Hyper-V: GUID)
`/VmLoadSnapshot`	- Load a specific VM snapshot to use for the task (instead of the current VM state)
`/VmCreateSnapshot`	- Take a snapshot of the current VM state (for recovery), before running the task
`/VmListSnapshots`	- Display all snapshots for the selected VM
`/VmUser`	- Username to be used for login into the Guest OS
`/VmPassword`	- Password for the user account used in the Guest OS

Automation (Windows PowerShell)

```
# test a package by running it in VM (using a VM configuration profile)
$pi = Start-Process -Wait -PassThru -FilePath "C:\Program Files (x86)\Caphyon\Advanced Installer 11.6\bin\x86\VmLauncher.exe"
 -ArgumentList `
    "/NoUI /SetupPath `"D:\Test\ProductName-SetupFiles\ProductName.msi`" /Profile {0ECC5124-3B73-40B3-B928-B272E40B79AD} "
Write-Output "Run in VM returned: "  $pi.ExitCode

# test a package by running it in a VMware Workstation VM (optional: load a previous snapshot)
$pi = Start-Process -Wait -PassThru -FilePath "C:\Program Files (x86)\Caphyon\Advanced Installer 11.6\bin\x86\VmLauncher.exe"
 -ArgumentList `
    "/NoUI /SetupPath `"D:\Test\ProductName-SetupFiles\ProductName.msi`" /VmPath `"C:\VMware VMs\Windows XP\Windows XP.vmx`"
    /VmLoadSnapshot `"Initial\SnapshotA`" /VmUser caphyon /VmPassword pass "
Write-Output "Run in VM returned: "  $pi.ExitCode

# test a package by running it in a Hyper-V VM (optional: load a previous snapshot)
$pi = Start-Process -Wait -PassThru -FilePath "C:\Program Files (x86)\Caphyon\Advanced Installer 11.6\bin\x86\VmLauncher.exe"
 -ArgumentList `
    "/NoUI /SetupPath `"D:\Test\ProductName-SetupFiles\ProductName.msi`" /VmPath 598409EA-9A76-4A3F-92F7-F3D42638FACD
    /VmLoadSnapshot `"Initial\SnapshotA`" "
Write-Output "Run in VM returned: "  $pi.ExitCode

# display all snapshots of a VMware Workstation VM
$pi = Start-Process -Wait -PassThru -FilePath "C:\Program Files (x86)\Caphyon\Advanced Installer 11.6\bin\x86\VmLauncher.exe"
 -ArgumentList `
    "/VmListSnapshots `"C:\VMware VMs\Windows XP\Windows XP.vmx`" "
Write-Output "Run in VM returned: "  $pi.ExitCode

# display all snapshots of a Hyper-V VM
$pi = Start-Process -Wait -PassThru -FilePath "C:\Program Files (x86)\Caphyon\Advanced Installer 11.6\bin\x86\VmLauncher.exe"
 -ArgumentList `
    "/VmListSnapshots 598409EA-9A76-4A3F-92F7-F3D42638FACD "
Write-Output "Run in VM returned: "  $pi.ExitCode
```

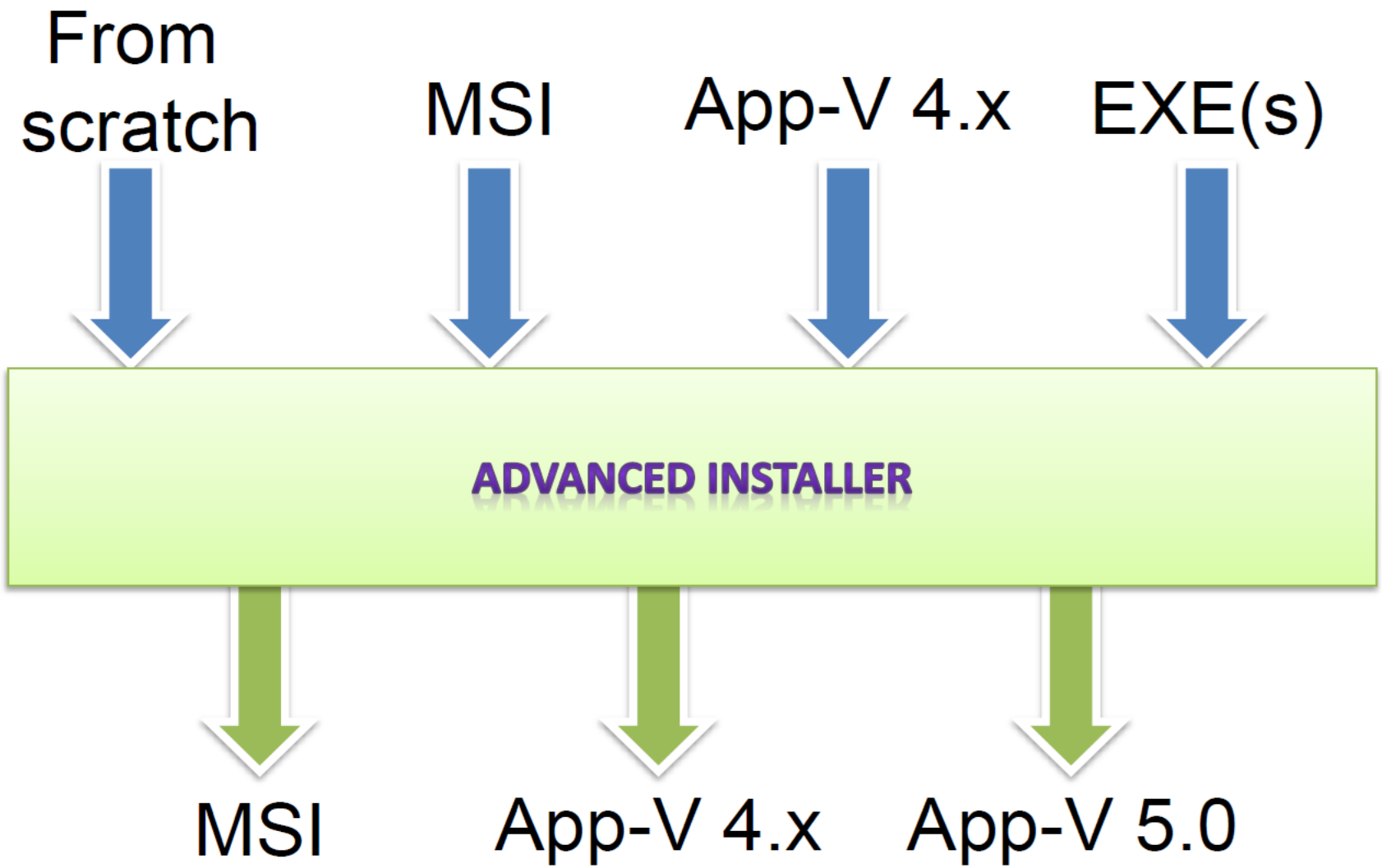
DEMO

Test a built package in VM.

Roadmap

- Support for more virtualization solutions:
  - VMware vSphere, ESX servers
  - Hyper-V remote VMs
- Further improvements in Repackager accuracy.
- Support for more VM testing scenarios.
- PowerShell interface for Advanced Installer automation.

One tool does it all.
Fast, easy to use and powerful.



Thank you for your time

www.advancedinstaller.com

Tweet us:
https://twitter.com/advinst

Mail us:
support@advancedinstaller.com
victor@caphyon.com

Session code: 123