

W  **L** 20042758  **ME**  **TO** **AUST**  **N**



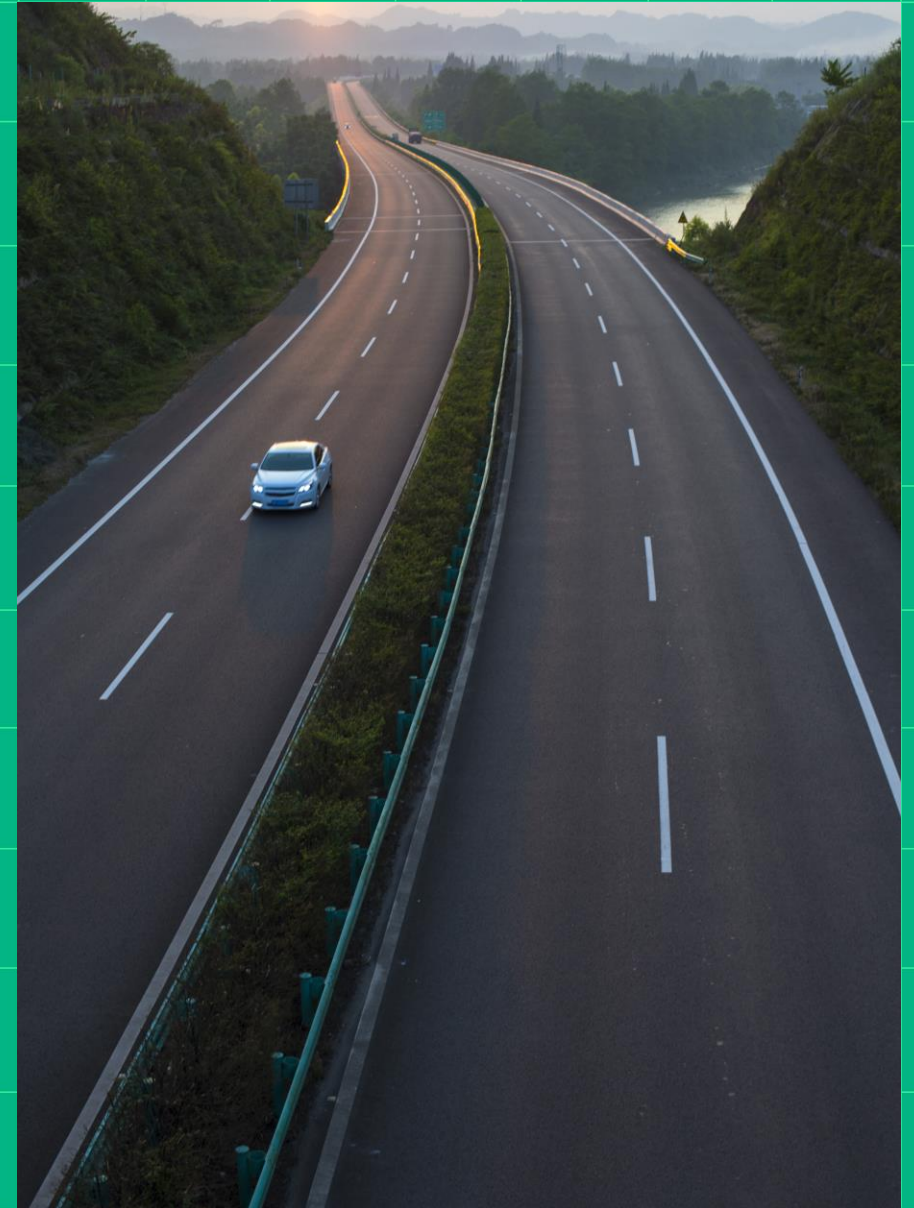
CONNECT

Get the Most Out of NI's Package Management Capabilities

Scott Richardson

May 24, 2023

10:30-11:30 am



Agenda

- Past
 - Why did we create NI Package Management framework?
- Present
 - What can a package contain?
 - What really is a package installer?
 - How can I customize and automate installations?
 - How does NIPM use feeds?
- Future
 - What is in NIPM's future?

Why did we create NI Package Management framework?

History and rationale...

Why did we create NI Package Manager framework?

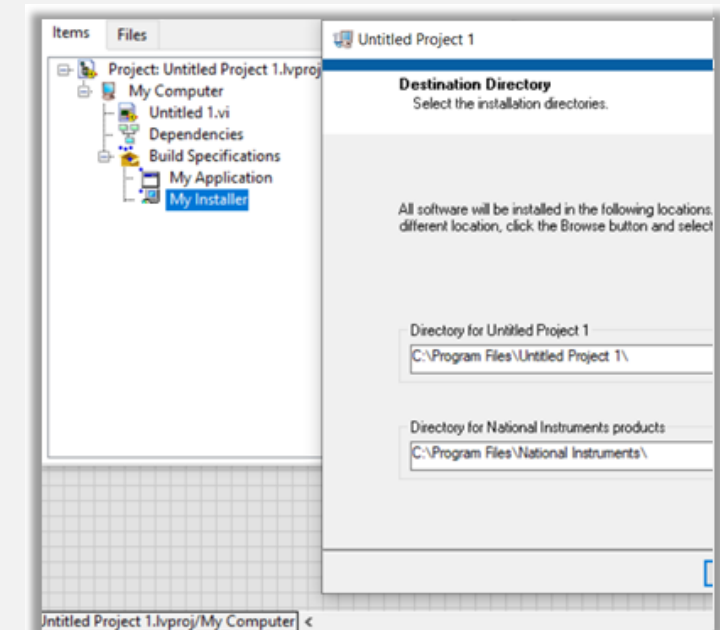
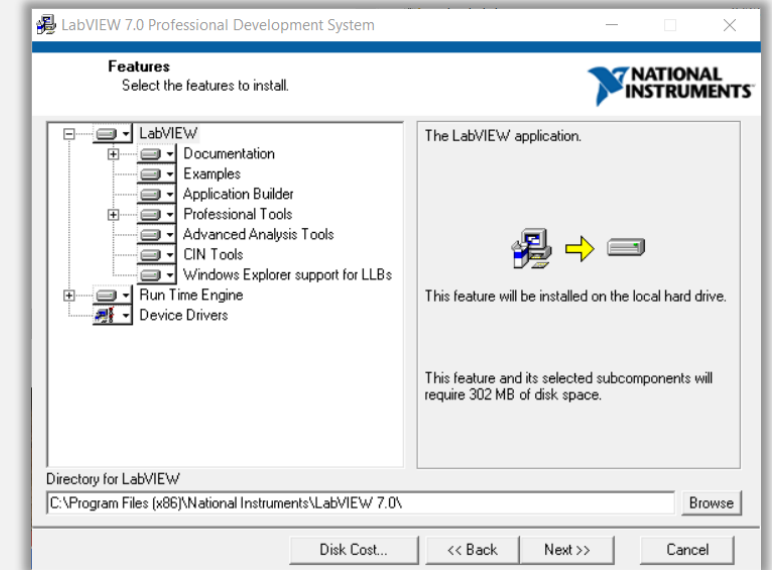
NI's pre-2019 installer framework

In 2003, NI created an **MSI-based** installer framework

- Consistent UX and reliability
- Used for all NI software installers
- Componentization, dependencies, and versioning
- Supported displaying EULAs and license activation

In 2005, customers could create custom installers

- First added “installer builder” capability to LabVIEW 8.0
- Later added to CVI, TestStand, Measurement Studio
- NI and customers benefited when NI introducing new features



Why did we create NI Package Manager framework?

Installer framework shortcomings...

By 2018, NI recognize the following missing needs:

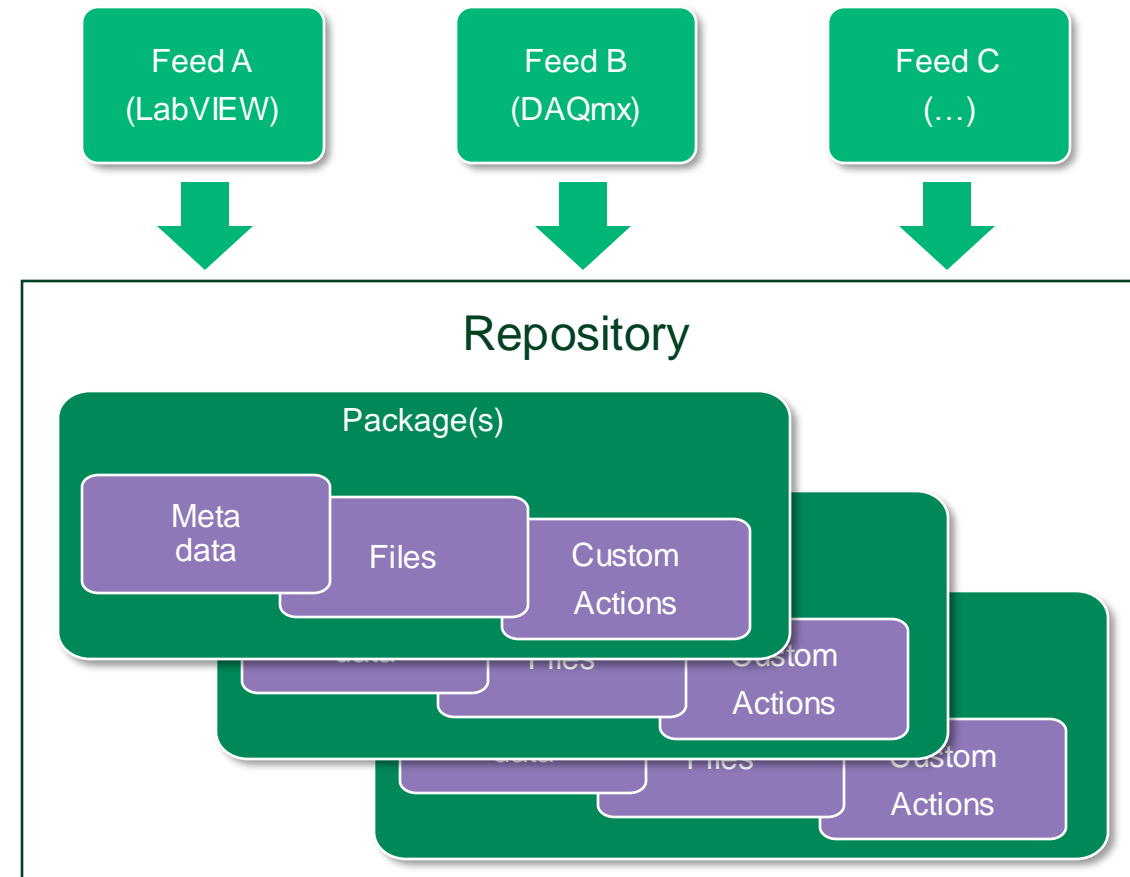
- × Users cannot **manage installed components**
- × No support for **web-based distribution**
- × Only “suite” software could **selectively download**
- × Limitations with **upgrading and patching** components
- × Limitations with component **dependency relationships**
- × Ill-suited to support **remote system management**
- × No support of **low-level custom workflows**



What is NI's Package Management Framework?

In 2019, NI released **framework** to package, distribute, & manage software on Windows

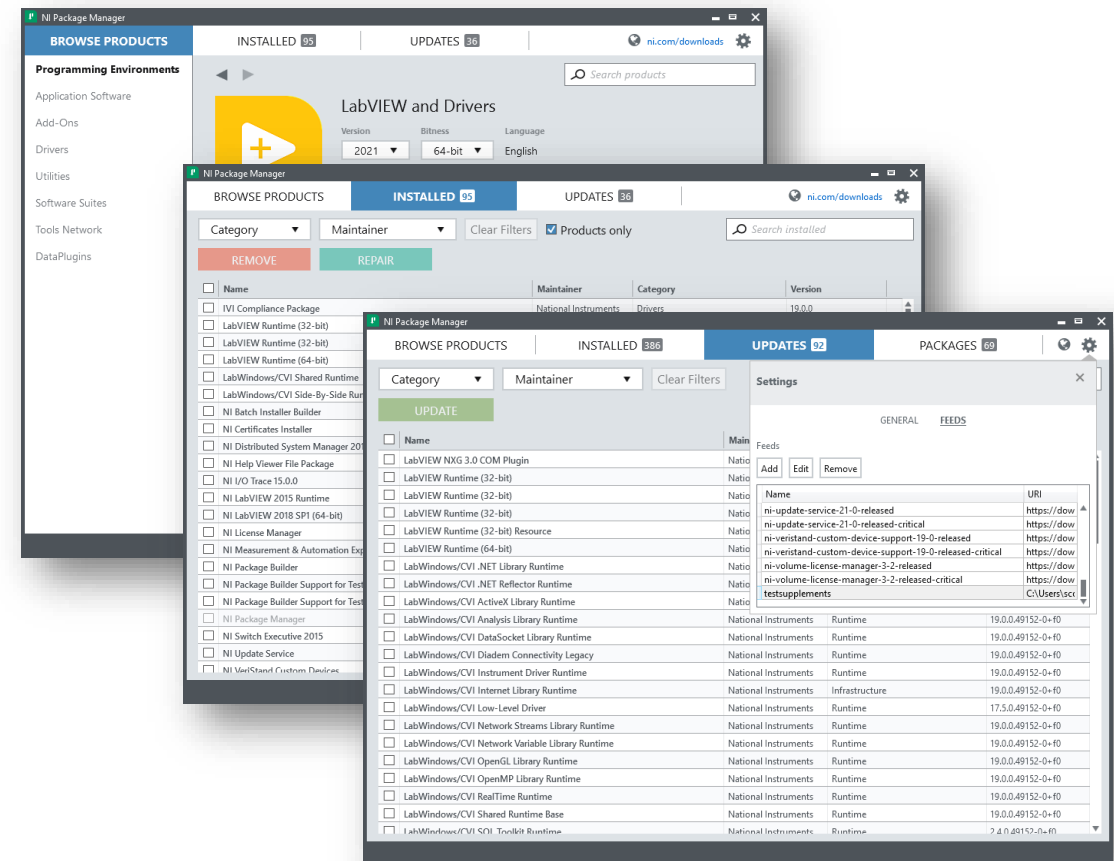
- Includes:
 - **Package** – “zip” file that represents a software component that contains:
 - Metadata: name, description, version, etc.
 - Files and their install paths
 - Dependency information to other packages
 - **Feed** – list of packages and where to find them
 - **Repository** – collection (or pool) of packages in directories, network, or cloud
 - **NI Package Manager** – tools for installing and managing packages on a system
- All NI software now built and deployed using this framework.
- NI has published over 7,300 times with 100K+ packages



What is NI Package Manager?

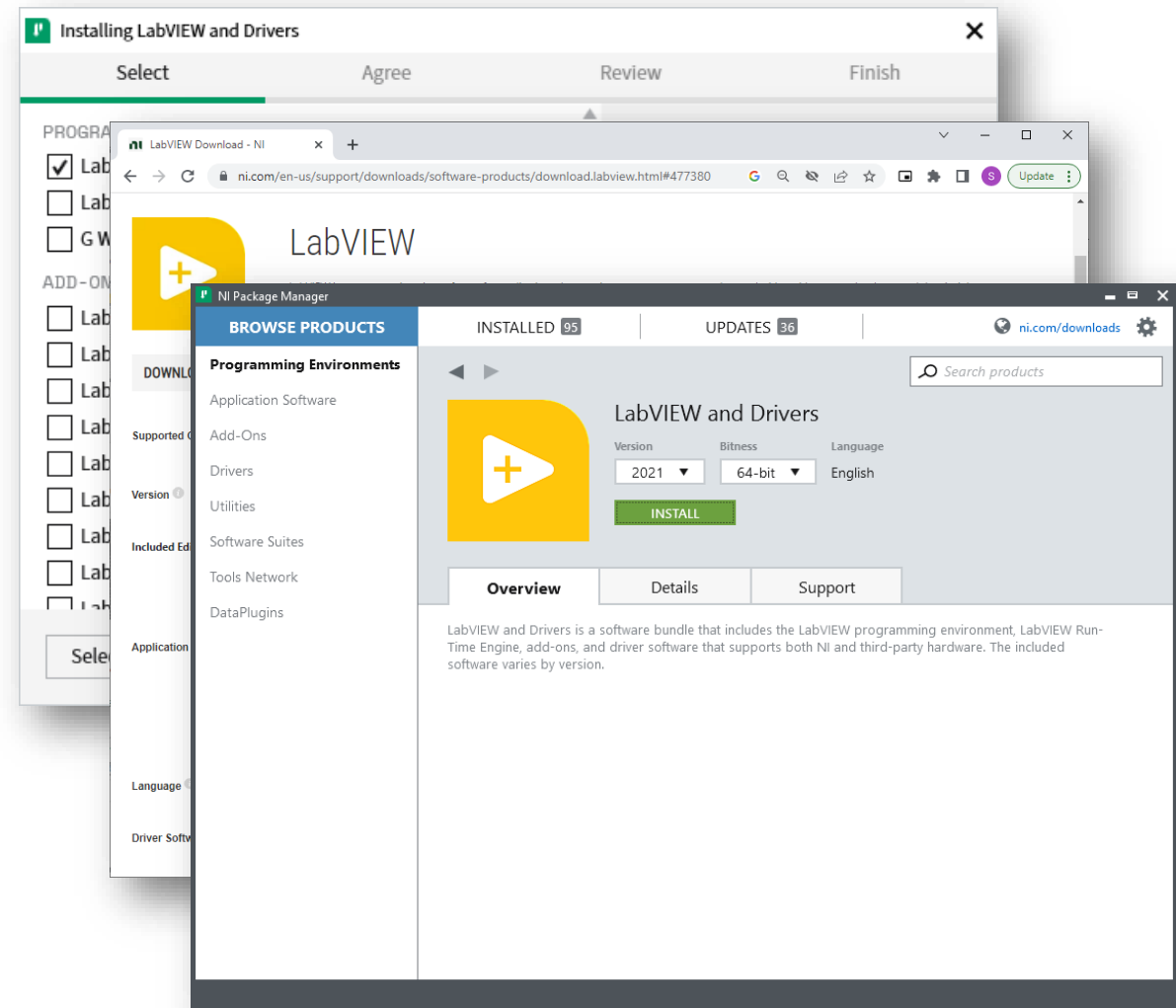
Use GUI to install and manage software

- Browse for software products from ni.com
- Lists installed packages and discover recommendations not yet installed
- List available updates (patches) to install
- Repair or remove installed packages
- Configuration settings
 - Flexible filtering of list of packages
 - Update registered feeds for system
 - Options to show more package information



Packages might be cool, but what about installers?

- Package-based installers supported
 - Interactive workflow displays a "wizard" GUI
 - Prompts user to install NI Package Manager (if not already installed)
 - Prompts user to select software to install
- How launched?
 - Ni.com downloads
 - Online installer
 - Small executable that downloads packages from ni.com
 - Offline installer
 - Large ISO, full installer, and no internet required
 - NIPM's Browse Products tab
 - Select product and version to install
 - Downloads packages from ni.com





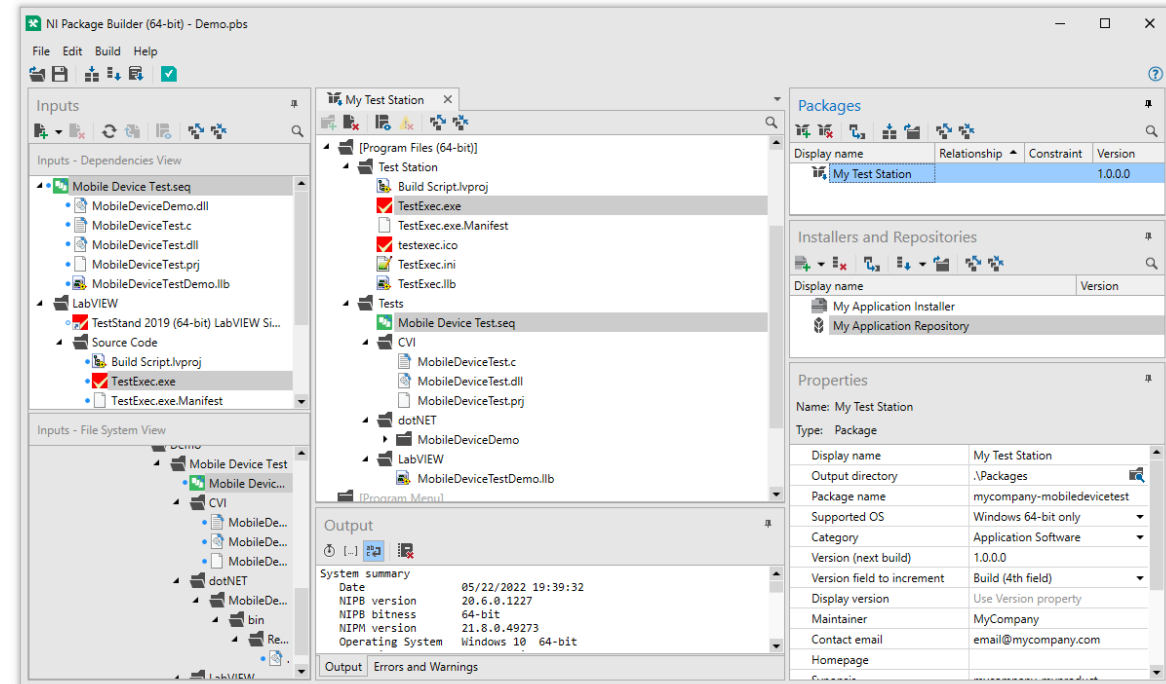
What tools can I use to build packages and installers?

What tools can I use to build packages and installers?

Package and Installer Build Tooling

Tools to manage project dependencies for development and automate build processes

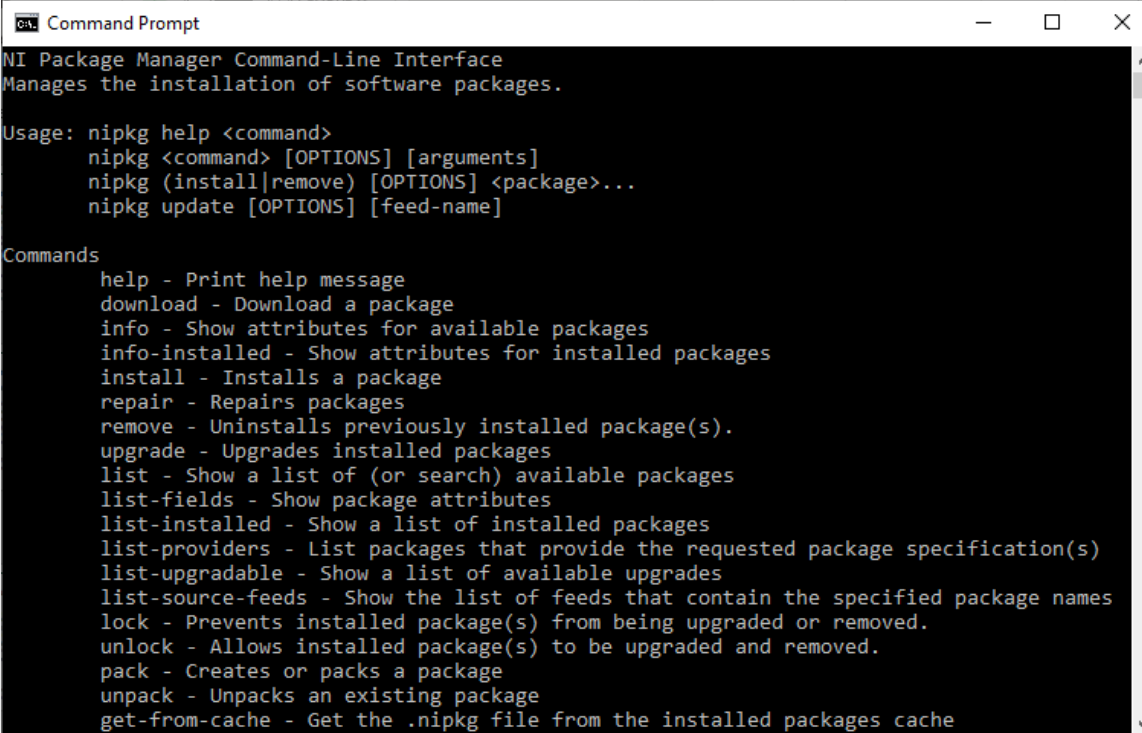
- LabVIEW's "Package" build specifications
 - Create package, feed, and installer
 - Include application and dependencies
 - Upload packages to SystemLink
 - Use CLI/API to automate builds
- TestStand's Deployment Utility
 - Create package, feed and installer
 - Include sequence files, code modules, and dependencies
 - Use CLI to automate builds
- NI Package Builder
 - Create multiple packages, feeds, and installers
 - Supports TestStand deployment workflows
 - Create "suite" installer from other installers
 - Simple CLI to automate building solutions



What is NIPM CLI?

Use nipkg.exe to perform and automate low-level commands for managing packages

- Install, update, repair, and remove packages
- Update registered feeds for system
- List installed and available packages
- Download packages from registered feeds
- Pack (create) and unpack packages
- Create and edit feeds, and list feed contents
- Full help available (`nipkg help`)



```
Command Prompt
NI Package Manager Command-Line Interface
Manages the installation of software packages.

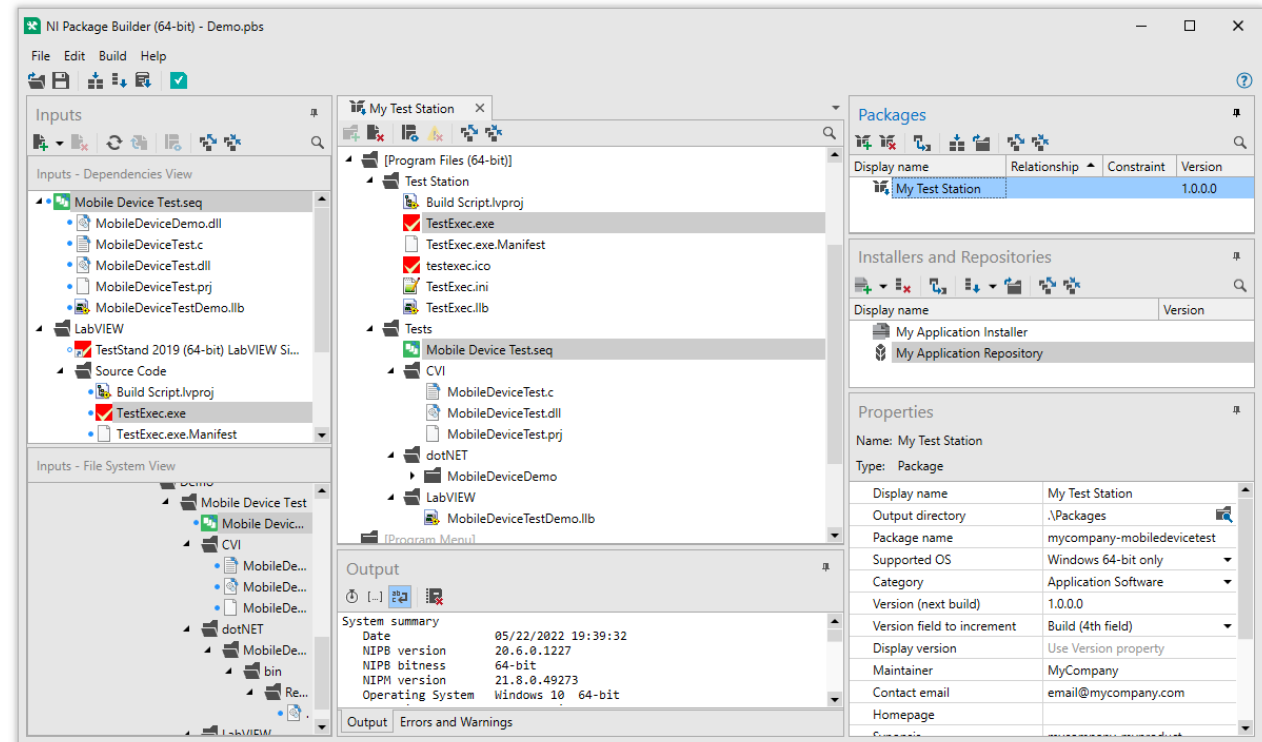
Usage: nipkg help <command>
       nipkg <command> [OPTIONS] [arguments]
       nipkg (install|remove) [OPTIONS] <package>...
       nipkg update [OPTIONS] [feed-name]

Commands
  help - Print help message
  download - Download a package
  info - Show attributes for available packages
  info-installed - Show attributes for installed packages
  install - Installs a package
  repair - Repairs packages
  remove - Uninstalls previously installed package(s).
  upgrade - Upgrades installed packages
  list - Show a list of (or search) available packages
  list-fields - Show package attributes
  list-installed - Show a list of installed packages
  list-providers - List packages that provide the requested package specification(s)
  list-upgradable - Show a list of available upgrades
  list-source-feeds - Show the list of feeds that contain the specified package names
  lock - Prevents installed package(s) from being upgraded or removed.
  unlock - Allows installed package(s) to be upgraded and removed.
  pack - Creates or packs a package
  unpack - Unpacks an existing package
  get-from-cache - Get the .nipkg file from the installed packages cache
```

Why did we create NI Package Manager framework?

Demo...

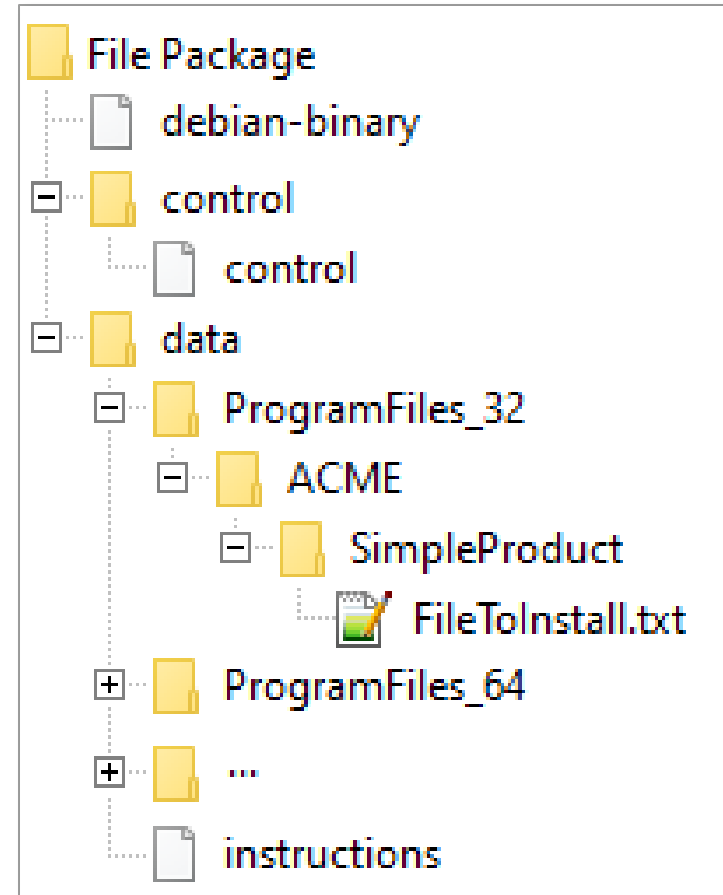
Creating packages



What can a package contain?

File packages

- Contains:
 - Control file (attributes)
 - Metadata: name, description, version, etc.
 - Dependency information to other packages
 - Files and their install paths
 - Base directory is OS “target root”
e.g., System, ProgramFiles
 - Followed by installation directory structure
 - Instructions file (optional)
 - Additional actions taken after installing files
- Built by LabVIEW, TestStand and NI Package Builder



Control File

Important attributes:

- *Plugin* – (EULA | File | WinInst)
- *Package* and *DisplayName*
- *Description*
- *Section* – name for “grouping”, i.e., category
- *Versions*
 - *Version* – Debian format
 - *DisplayVersion* – Text
 - *CompatibilityVersion* – min NIPM required
- *Visibility*
 - *StoreProduct* – ni.com
 - *UserVisible* – in NIPM
 - *VisibleForRuntimeDeployment* – for “builders”
- *Relationships between packages:*
 - *Depends, Recommends, Suggests, Conflicts, Provides, Enhances, Supplements, Replaces*

Example for File package:

```
CompatibilityVersion: 230300
Depends: acme-part-small (>=1.0.0.0), acme-part-large (>=1.0.0.0)
Description: Update registry entries for our application.
DisplayName: ACME Update Registry Entries
DisplayVersion: 2023
Eula: eula-acme
Maintainer: Acme Corporation <email@acmecorp.com>
Package: acme-update-registry-entries
Plugin: file
Section: Infrastructure
StoreProduct: no
UserVisible: yes
Version: 1.0.0.0
VisibleForRuntimeDeployment: no
```

Instructions File

Contains:

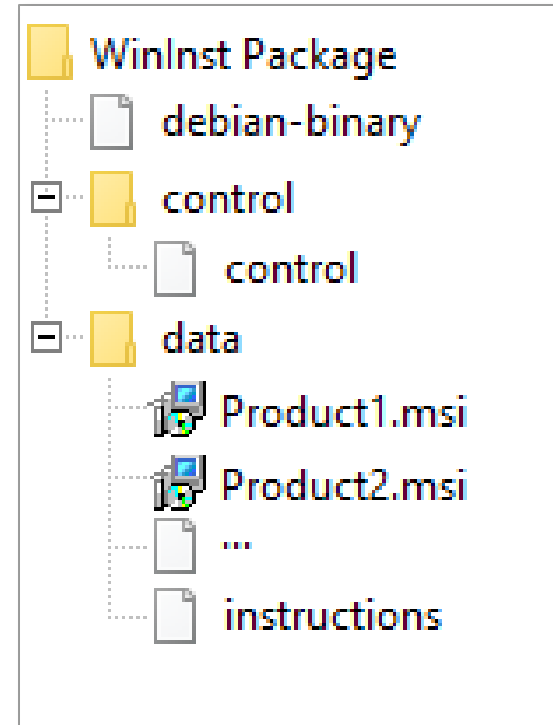
- Shortcuts –
 - Where to create shortcut
 - What the shortcut calls, including arguments
- Custom executes – runs exe or opens file
 - Calls during *install*, *repair*, or *uninstall*, and *pre*, *post* or *postall* to package operations
 - Pause script to debug console output:
 - */K* with CMD
 - *-NoExit* with PowerShell
 - Use *IgnoreError* for pre-uninstall
 - Otherwise, returned error prevents uninstalling package!

Example for File package:

```
<instructions>
  <shortcuts>
    <shortcut>
      <destination root="ProgramMenu"
        path="Test\Shortcut.lnk"/>
      <target root="Program Files"
        path="Shortcuts\executable.exe"
        arguments="-open %desktop%\file.txt"/>
    </shortcut>
  </shortcuts>
  <customExecutes>
    <customExecute root="ProgramFiles_32"
      exeName="ACME\Product\myexe.exe"
      arguments="%desktop%\file.txt"
      step="install"
      schedule="post"
      wait="yes"
      ignoreErrors="yes"
      hideConsoleWindow="yes"/>
  </customExecutes>
</instructions>
```

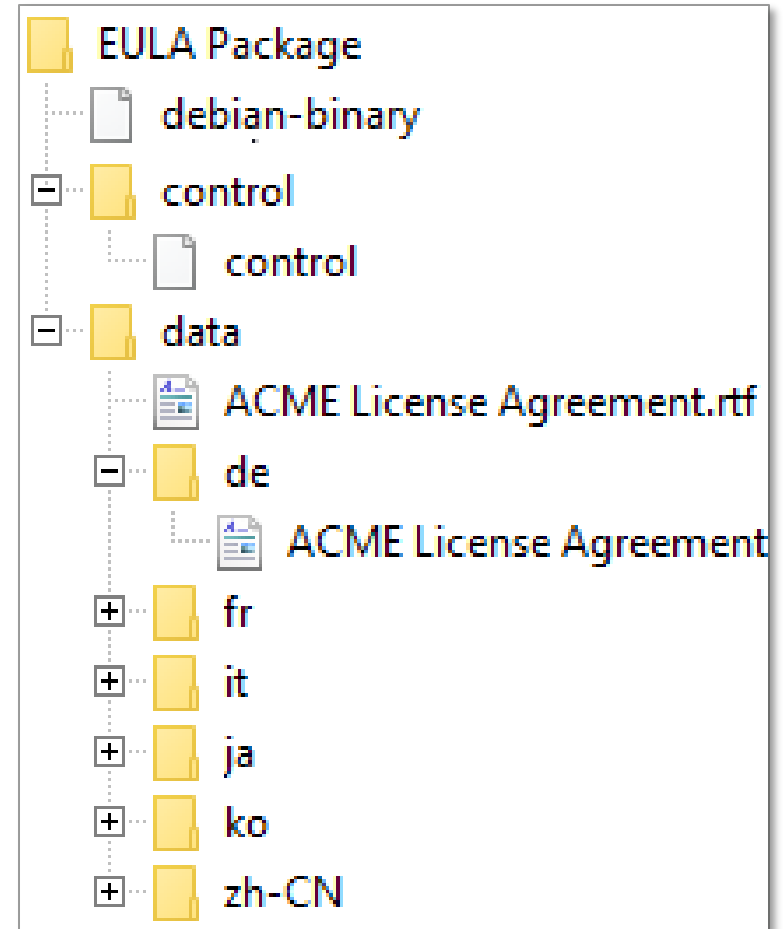
WinInst packages

- Contains
 - Control file
 - One or more MSI files
 - Instructions file (optional)
- Most NI software uses this package type
 - NI MSIs can also be deployed using MSI-based installer builder, e.g., LabVIEW



EULA packages

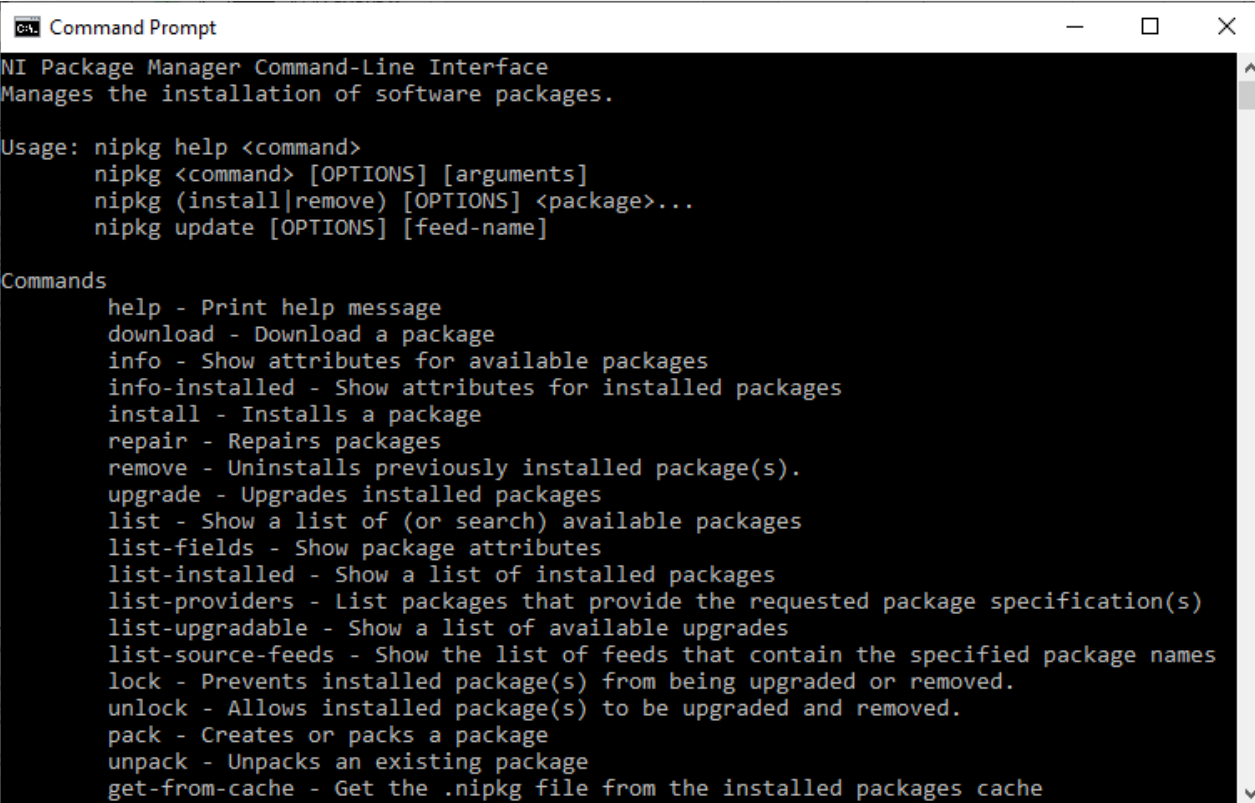
- Contains
 - Control file
 - Default license file (.txt or .rtf)
 - Language specific files (optional)
- EULA not installed like File and WinInst
 - Must be included in feed for installation
 - Installation displays when “EULA” attribute set in File/WinInst package



Demo...

Inspecting and modifying packages

What can a package contain?



```
Command Prompt
NI Package Manager Command-Line Interface
Manages the installation of software packages.

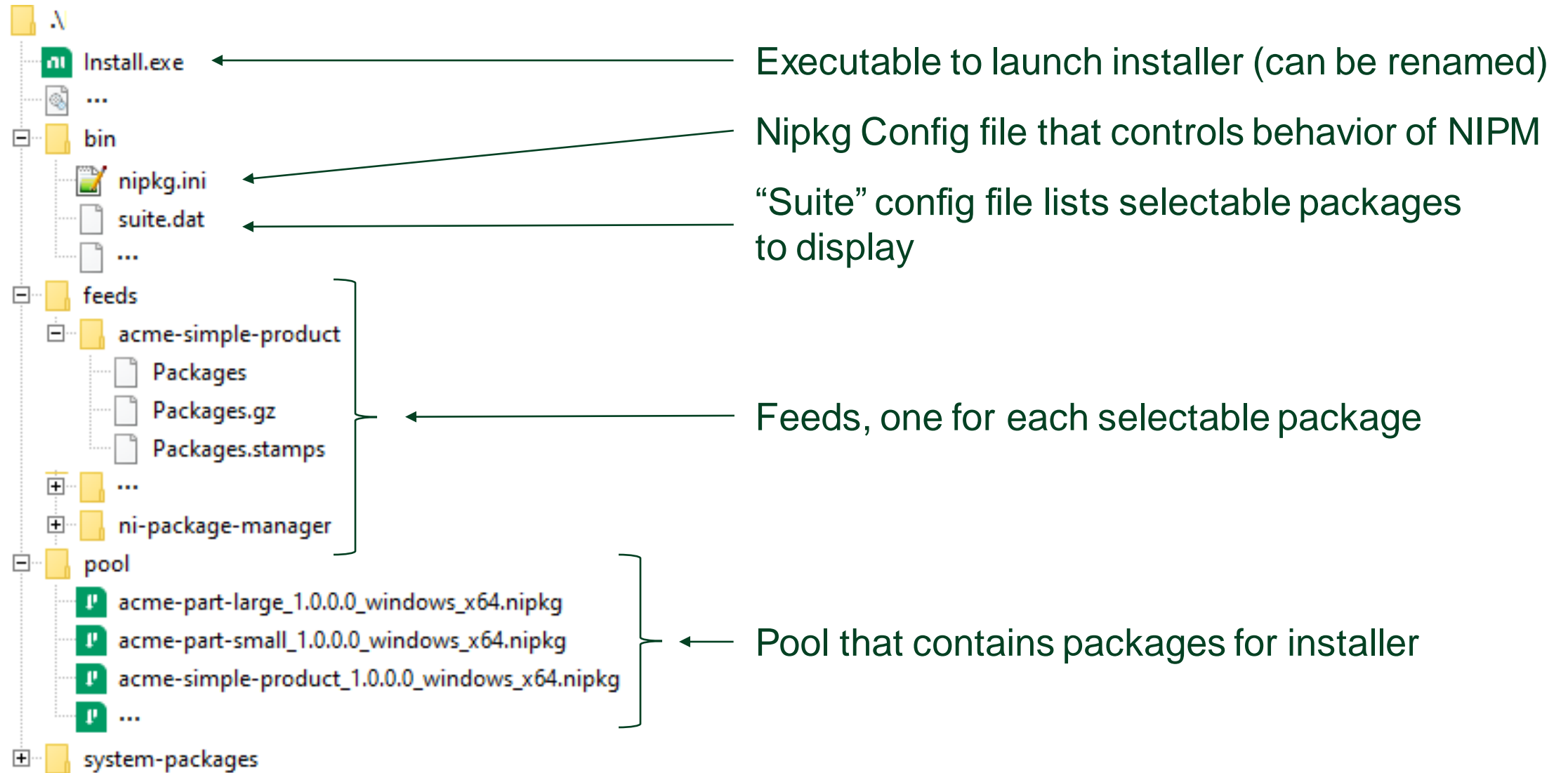
Usage: nipkg help <command>
       nipkg <command> [OPTIONS] [arguments]
       nipkg (install|remove) [OPTIONS] <package>...
       nipkg update [OPTIONS] [feed-name]

Commands
help - Print help message
download - Download a package
info - Show attributes for available packages
info-installed - Show attributes for installed packages
install - Installs a package
repair - Repairs packages
remove - Uninstalls previously installed package(s).
upgrade - Upgrades installed packages
list - Show a list of (or search) available packages
list-fields - Show package attributes
list-installed - Show a list of installed packages
list-providers - List packages that provide the requested package specification(s)
list-upgradable - Show a list of available upgrades
list-source-feeds - Show the list of feeds that contain the specified package names
lock - Prevents installed package(s) from being upgraded or removed.
unlock - Allows installed package(s) to be upgraded and removed.
pack - Creates or packs a package
unpack - Unpacks an existing package
get-from-cache - Get the .nipkg file from the installed packages cache
```



What really is a package installer?

Structure of an offline installer



Offline installer's nipkg.ini file

- Setting `applysystemsettings=true`
 - Honor system and user configuration file settings, like “show hidden package” and “enabling MSI logging”
- Setting `applysystemrepositories=false`
 - Ignores all feeds other than from installer
- Initially registers 2 feeds as shown

```
[nipkg]
applysystemsettings=true
applysystemrepositories=false
```

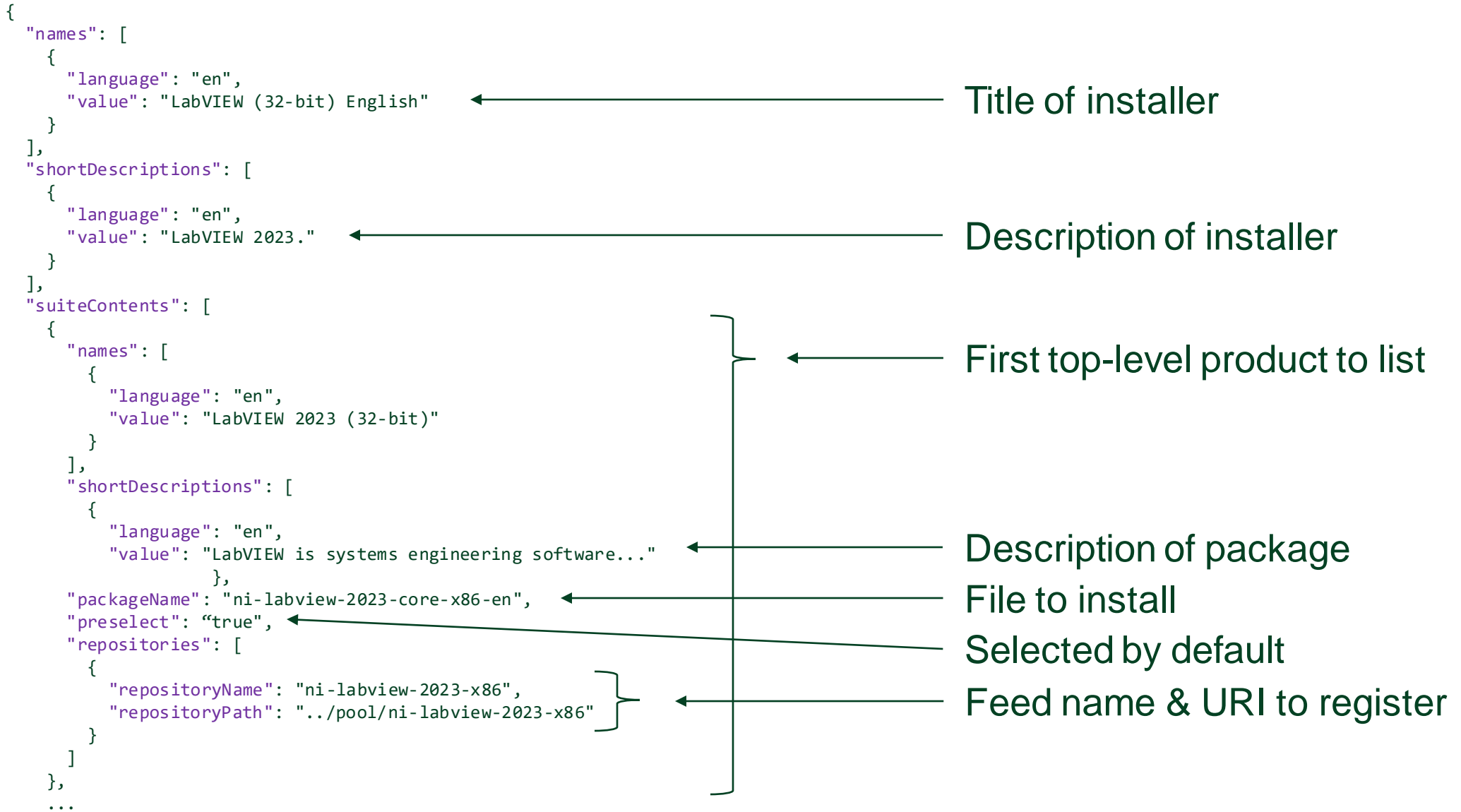
```
[nipkg.repo.system-packages]
name=system-packages
uri=../system-packages/
```

```
[nipkg.repo.ni-package-manager-feed-1]
name=ni-package-manager-feed-1
uri=..\feeds\ni-package-manager
```



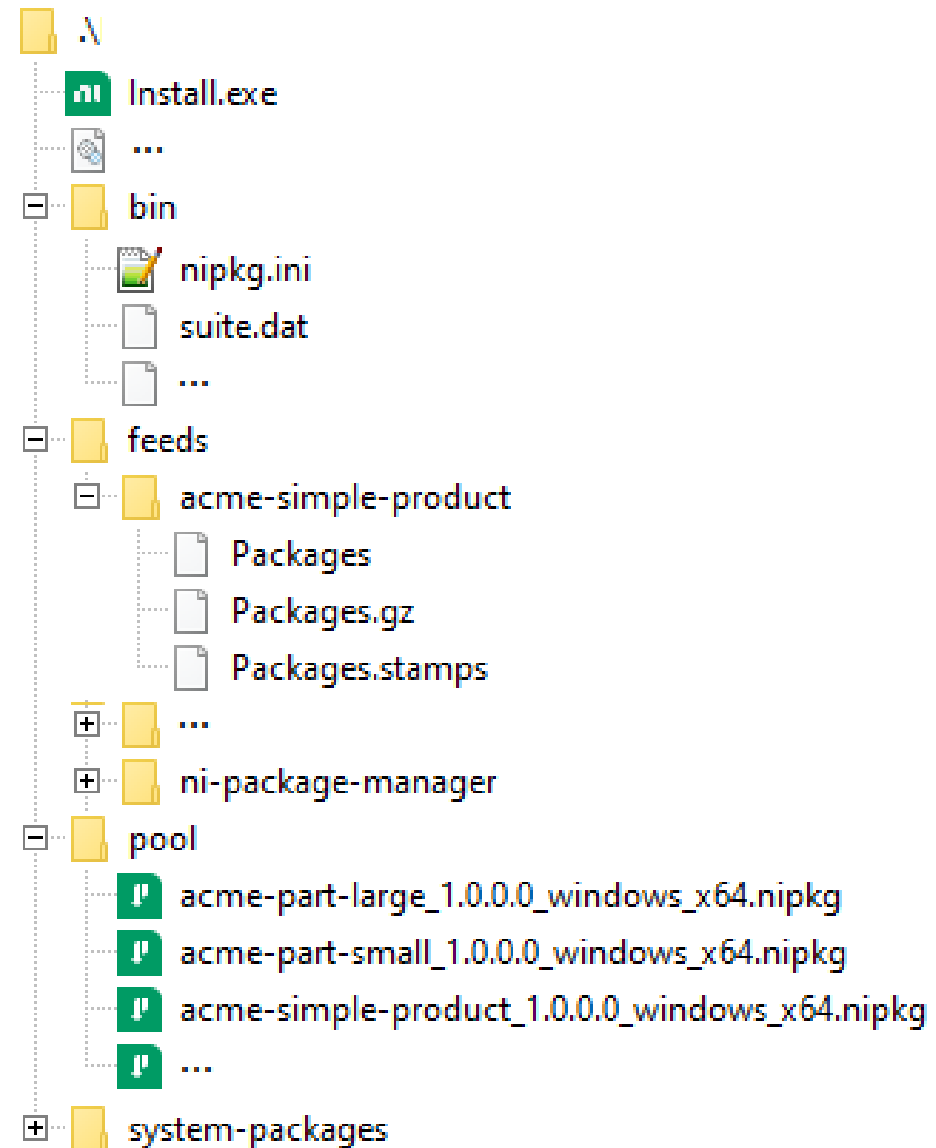

What really is a package-based installer?

Structure of installer's suite.dat file



Demo...

Creating a custom
online installer





How can I automate
installations?

Call “install.exe /?” to get installation options

- `--accept-eulas` Allows the installation of packages to indicate acceptance of all applicable license agreements.
- `--autoinstall-nipkg` Passively install NI Package Manager, but the rest of the installation might require user interaction.
- `--config=...` Uses a local configuration for this transaction.
- `--display-all-packages` Overrides the user settings for the session and display all packages on the summary.
- `--feeds=...` Specifies which feeds to use for installing or updating a package.
- `--force-essential` Forces the installation of essential packages.
- `--force-locked` Forces the installation of locked packages.
- `--hide-completion` Prevent showing end of the installation confirmation.
- `--hide-splashscreen` Skips displaying the NI Package Manager splash screen on launch.
- `--language=...` Specifies the user interface language by a given language tag.
- `--no-winmif-mutex` Prevents waiting for an already started installation to complete.
- `--output-transactions=...` Writes transaction details to output file.
- `--passive` Runs in a non-interactive mode where the user only sees graphical progress.
- `--prevent-activation` Prevents license manager from launching after installation.
- `--prevent-reboot` Prevents NI Package Manager from rebooting.
- `--quiet` Runs in a fully silent mode so that the user does not see any windows or graphical progress.
- `--temp-feeds=...` Specifies which feeds to temporarily register for installing or updating a package.
- `--update-feeds` Updates feeds before launching any windows.
- `--window-handle=...` Sets the parent window handle and style of NI Package Manager.

Controlling NI software installation directories

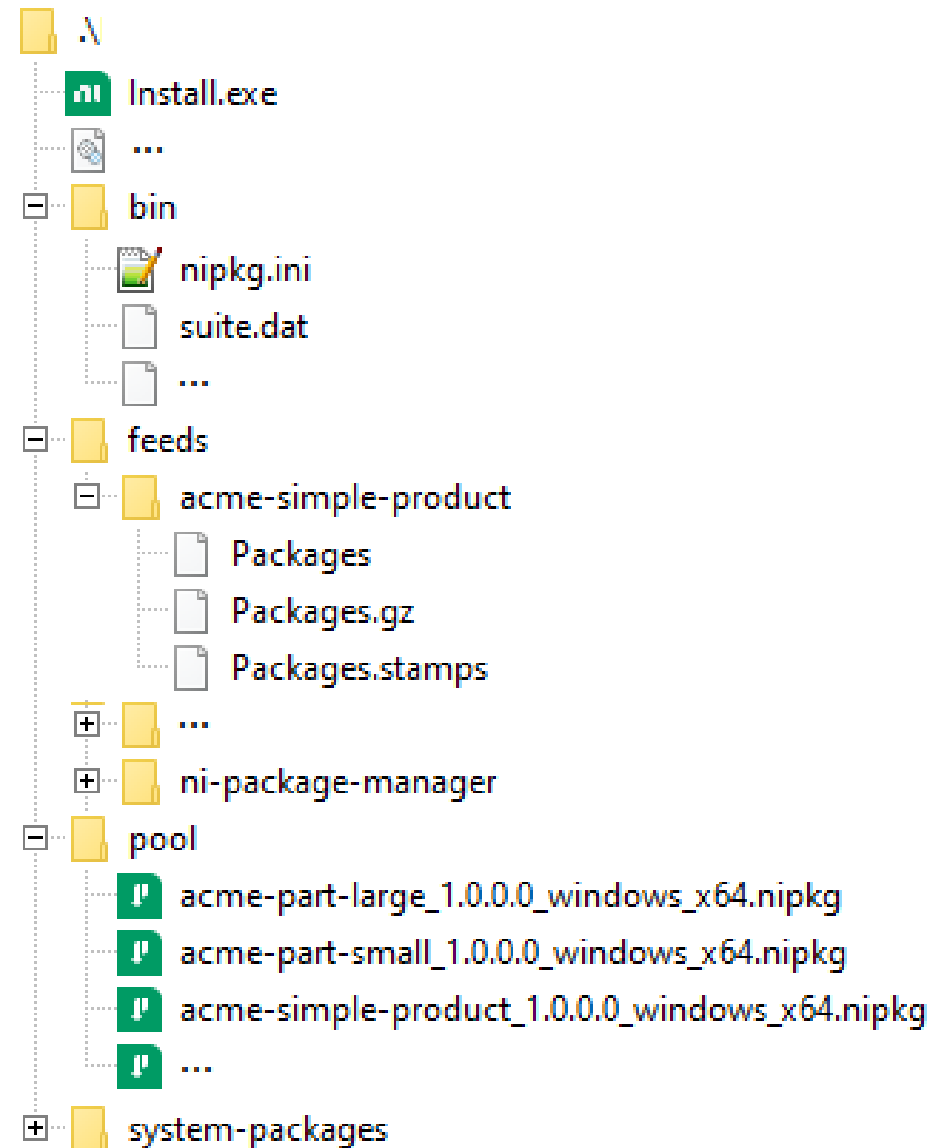
- NI software installers do NOT prompt users for destination directories
- Almost all software uses registry for their directories
- Before installing, pre-create directory and set registry key
- Examples:
 - HKEY_LOCAL_MACHINE\SOFTWARE\National Instruments\Common\Installer
 - NIDIR64 NI Program Files directory
 - NIPUBAPPDATADIR NI ProgramData directory
 - NIPUBDOCSDIR NI Users Public Documents director
 - HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\National Instruments\Common\Installer
 - NIDIR NI Program Files (x86) directory
 - NIPUBAPPDATADIR NI ProgramData directory
 - HKEY_LOCAL_MACHINE\SOFTWARE\National Instruments\LabVIEW\22.3
 - PATH LabVIEW (64-bit) directory

[KB: How Do I Change the Installation Folder of My Software in NI Package Manager?](#)

How can I automate installations?

Demo...

Custom install.exe and
using 3rd party Installer





How does NIPM use feeds?

NIPkg configurations files

- INI file format
 - Contains plugin options, GUI options, nipkg options, and registered feeds
- NIPM combines settings on launch from:
 - Installed files in Program Files NIPM directory
 - ni-package-manager-defaults.ini
 - ni-package-manager-released.ini
 - System-level file
 - %programdata%\National Instruments\NI Package Manager\Settings\nipkg.ini
 - Contains NIPM ni.com managed feeds
 - User-level file
 - %localappdata%\National Instruments\NI Package Manager\nipkg.ini
 - Contains NIPM GUI user settings and added feeds
 - Custom file when --config option used:
Custom config can ignore other files by setting **applysystemsettings** and **applysystemrepositories** to false

```
[nipkg.plugin.wininst]
name=wininst
override-windows-fast-startup=False
msilogs-enabled=true
```

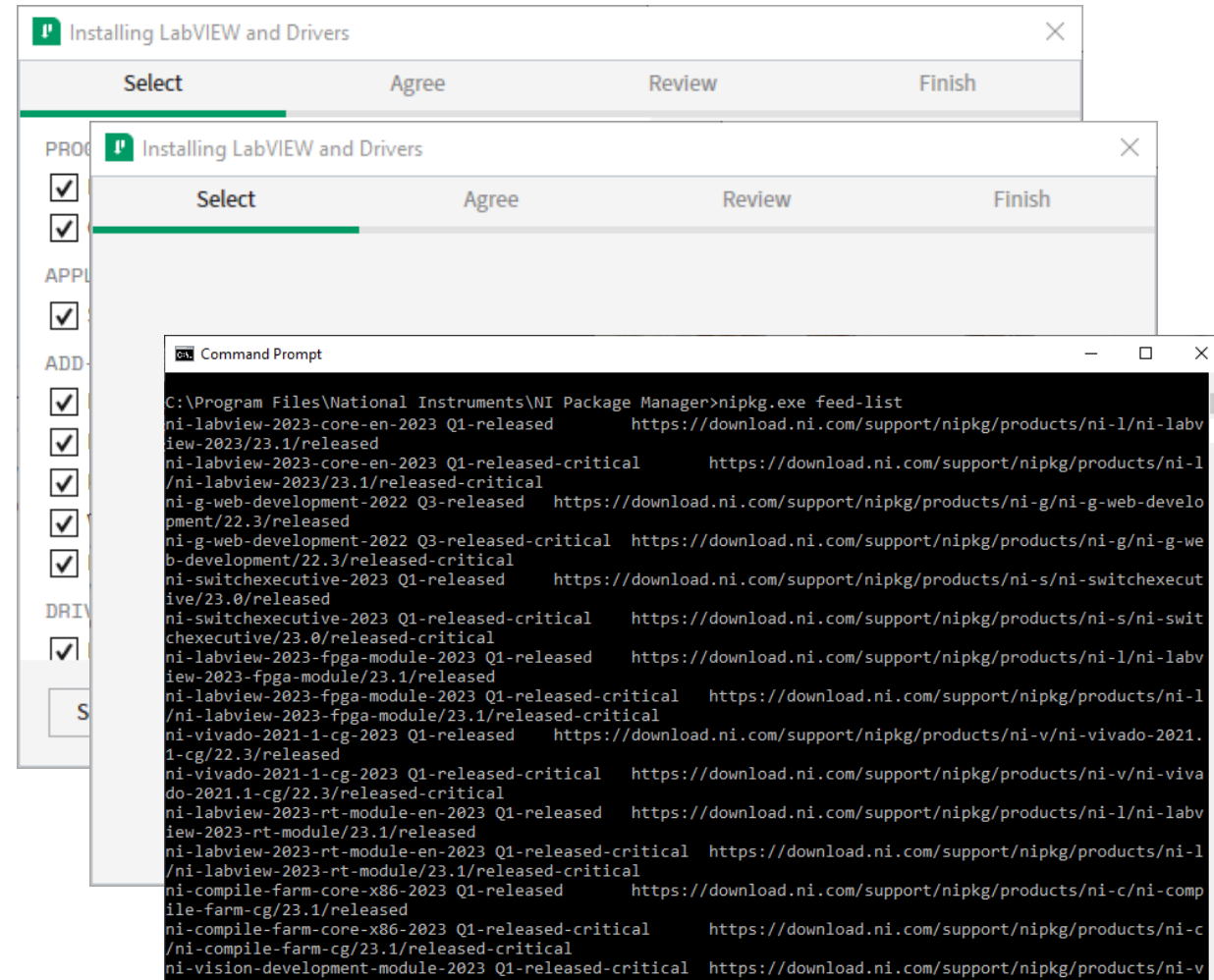
```
[nipkg.nipkgui]
automatically-install-missing-
dependencies=true
service-locator-preview=false
enable-browse-products=true
```

```
[nipkg.repo.acme.product1]
name=acme-product1
uri=https://download.acme.com/nipkg/product1
```

...

Discovering ni.com feed URIs

- Today, you cannot browse download.ni.com structure to discover feed URIs
- However, you can get URIs from online installer without installing
 1. Start install of software
 2. Select products
 3. Click next to Add Feeds
 4. Run `nipkg.exe feed-list`
 5. Cancel installation



Cloning a feed and its packages

- Use
 - Register remote feed
 - Download the packages from remote feed
 - Create local feed and add packages

- For example:

```
nipkg.exe feed-add --name=temp  
https://download.ni.com/support/nipkg/products/ni-p/ni-package-builder/22.5/released
```

```
nipkg.exe feed-download "ni-package-builder-2022 Q3-released" "c:\temp\myfeed"  
>> %temp%\output.txt
```

```
nipkg.exe feed-create "c:\temp\myfeed"  
"c:\temp\myfeed"
```

```
nipkg.exe feed-remove temp
```



What is in NIPM's future?

(What we are considering...)

Security

- Signing MSIs
 - Some system software is starting to prevent installation of unsigned MSIs
 - Microsoft Smart App Control in Windows 11
 - Beyond Trust
 - Currently NI software does not sign MSIs
 - We are researching now
- Credentialed feeds
 - NIPM is overprotective and blocks redirected URI paths from cloud servers
 - Cannot use feeds hosted on GitHub, OverDrive, S3, etc.
 - NIPM 23.5 will add support to allow redirection
 - NIPM does not support using credentials to connect to a remote feed URI
 - We should start researching this in late 2023

Short list of items we are considering next...

- Support absolute paths, i.e., non-OS based paths, such as D:\
- Allow a package to upgrade an older MSI-based installer
- Allow a package to require reboot after installation
- Allow package to create ARP (Add/Remove Programs) item
- More customization of installer behavior via config file
- Signing packages and/or feeds
- Setting registry keys



Questions and Discussion



SystemLink™

Use to manage distributed systems and much more...



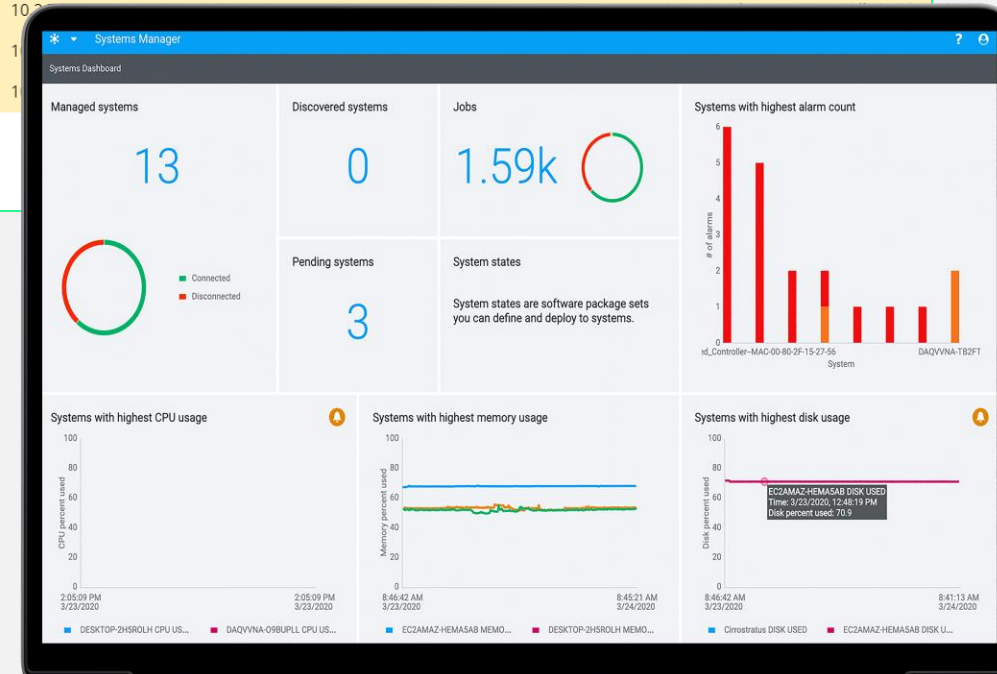
SystemLink

Managing software and data for test & measurement

Capabilities include:

- Remote Systems Management
- Test Asset Management
- Test Monitoring & Results Management
- Measurement Data Management

	Name	IP Address	Model Name	Operating System	Serial Number	Connection	Comments
<input type="checkbox"/> Automated Test Systems (2)							
<input type="checkbox"/>	PXIe-8840Quad-1	10.2.74.79	NI PXIe-8840 Quad-Core	Windows 7	030E1626	Connected	Test Station 1
<input type="checkbox"/>	PXIe-8840Quad-2	10.2.74.80	NI PXIe-8840 Quad-Core	Windows 7	030DDB85	Connected	Test Station 2
<input checked="" type="checkbox"/> Control Systems (4)							
<input checked="" type="checkbox"/>	NI-cRIO-9068-190CB7B	10.2.74.64	cRIO-9068	NI Linux RT 4.1	190CB7B	Connected	Test Cell 1
<input checked="" type="checkbox"/>	NI-cRIO-9068-190D5D5	10.2.74.65	cRIO-9068	NI Linux RT 4.1	190D5D5	Connected	Test Cell 2
<input checked="" type="checkbox"/>	NI-cRIO-9068-190D673	10.2.74.66	cRIO-9068	NI Linux RT 4.1	190D673	Connected	Test Cell 3
<input checked="" type="checkbox"/>	NI-cRIO-9068-190FDF5	10.2.74.67	cRIO-9068	NI Linux RT 4.1	190FDF5	Connected	Test Cell 4





Remote System Management

Discover, and efficiently manage and configure the software on your distributed systems using the following capabilities

- Manage which client systems can connect to server
- Upload packages and configure feeds that client systems can use
- Execute remote software installs, upgrades, and downgrades on systems using NI Package Manager as a service
- Capture, compare, and edit system “states” that represent software collections, and deploy to systems
- Deploy changes to multiple systems simultaneously using “jobs”
- Manage client system availability for updates using a lock/unlock feature

States	Category	Maintainer	View	Clear	Filter
Available					
Upgrades (5)					
Installed	LabVIEW Runtime (32-bit)	National Instruments	2016 f6	2016 f7	Upgrade
	LabVIEW Runtime (64-bit)	National Instruments	2018 SP1 f3	2018 SP1 f4	Upgrade
Feeds	NI Certificates Installer	National Instruments	2.0.1	2.0.2	Upgrade
	NI Package Builder	National Instruments	19.0.0	19.0.1	Upgrade
	NI Package Manager	National Instruments	19.6.0	20.0.0	Upgrade
Application Software (7)					
	Common Code Modules	Acme	3.0.0-0		Change version
	High Temp Test	Acme	2.0.0-0		Change version
	InstrumentStudio 2019	National Instruments	19.0.2		Change version
	Motherboard Test	Acme	2.0.0-0		Change version
	NI SystemLink Client	National Instruments	19.6.1		Change version
	TestStand (32-bit)	National Instruments		2019 f2	Install
	TestStand (64-bit)	National Instruments	2019 f2		Install

