An In-Depth Look at the Autodesk® AutoCAD® App Autoloader Module

Fenton Webb
ADN

Agenda

The Requirement
The Problem
The Solution
How it works?
Demos

The Requirement

New Autodesk AppStore!!

The Store must be…

1) Easy for the user to buy, install and use Apps
   - User experience is absolutely key for success
2) Easy for developers to utilize while observing (1)
   - Without apps, there is no store!!
   - The need to make it easy to deploy apps is paramount.

And, must also work for

- Mac, Win32, Win64
- All languages
- All Autodesk Products

The Problem

Long History of Deployment Headaches

ARX/DBX/CRX

- Version specific registry keys and folder paths, installer nightmare
- Dependent DLL loading issues, OS paths not setup or conflicting, DLL Hell
- Load order issues, SDK versioning issues, dependency issues

.NET

- Same issues as ObjectARX
- LoadFrom() context issues, must be installed into the AutoCAD exe folder – installer nightmare

The Problem

Long History of Deployment Headaches (cont.d)

LISP

- Automatic startup extremely difficult, must modify user editable acad.lsp/acaddoc.lsp – installer nightmare
- Support Path setup – installer nightmare
- LISP is an uninstall nightmare, full stop

CUI/Mnu Files

- Automatic startup is not nice, must queue a LISP CuiLoad expression to the command line
- No way to uninstall cleanly, menus stay even when deleted!!
  - Must write a program that runs once the program is uninstalled in order to remove menu items by hand!

The Problem

Long History of Deployment Headaches (cont.d)

Win32, Win64 and Mac runtime are all different

But in most cases, the actual ‘Product’ is the same to the user, why does a user need to care what platform the installer is for?

Building an installer for each platform is a lot of work

These are plugins, one deployment method should be enough

The Problem

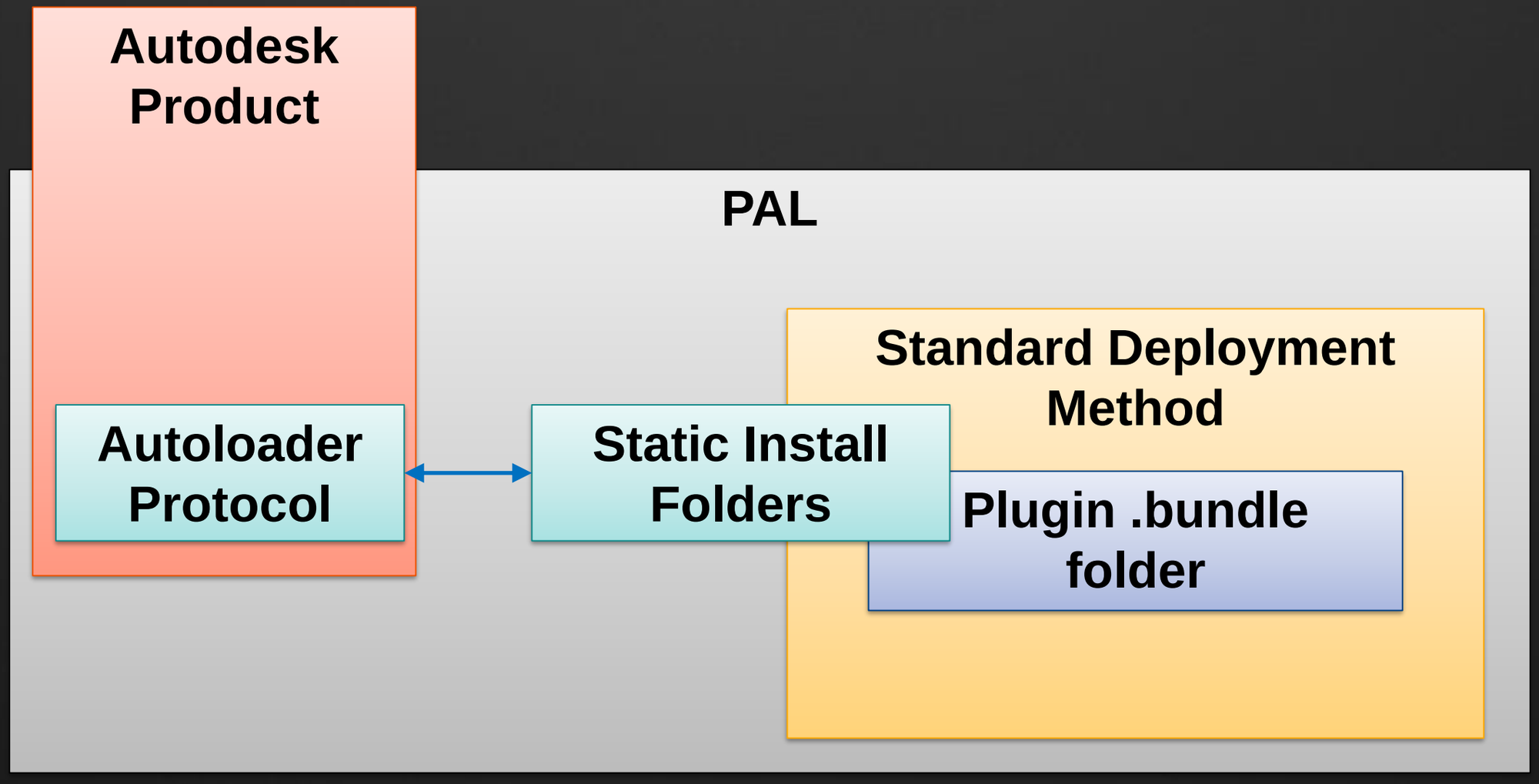
Long History of Deployment Headaches (cont.d)

Installing to different User accounts requires 2nd stage installer

- Version, Language, Vertical installer nightmare
- Need to piggy back onto AutoCAD's 2nd stage installer
  - It's tricky to implement and understand
  - Developers simply avoid it
    - Avoiding standard mechanism's does not give a good User experience

The Solution

Professional Application Loader



The Solution

Two Static Plugin Folders **(static = cannot be changed)**

<UserRoaming>/Autodesk/ApplicationPlugins

- Installing your App here provides single user install model

<ProgramData>/Autodesk/ApplicationPlugins

- Installing your App here provides All User installation model
  - no 2nd stage installer required!!!

The Solution

Why use Static folders?

Because the folders do not change…

- Installation uses a simple CopyTo protocol
  - Non-configurable, static folders make life much easier
    - Installer creation is super simple
    - Although, Installer is not needed

Cross Platform - Works for Mac and Windows

- ‘CopyTo’ deployments work on all systems
  - Copying folders works on all platforms

The Solution

Static Folders contain .bundle plug-in folders

.bundle folder structure separated into components

- Win32, Win64 and/or Mac specific runtime
  - Combined runtime, in one bundle package
  - User doesn’t care about what platform he’s running on!
    - Having it all in one folder structure key

- Version specific runtime can be easily included
  - Features enabled for 2010 but not 2009
  - Crash in sp1, or working in sp2

- Bundle includes your entire application runtime
  - Just like your Program Files folder
    - Support files, images, help, etc etc
    - Cui files, DCL, DLL dependencies, etc etc

The Solution

Static Folders Contain.bundle plug-in folders

.bundle folder structure separated into components

- Multiple Autodesk product support combined into one
  - Combine your Revit, Inventor, AutoCAD product into one bundle!!
  - AutoCAD WS, Vault support to come

The Solution

PAL – The Autoloader Protocol

Product implemented ‘Autoloader’ component

Universal loading mechanism for all Autodesk products

- Implemented through the ‘Autoloader’ protocol
- Processes all bundles in the ApplicationPlugins folders
  - For loading, registering, etc
    - On Startup
    - On Appearance (more later)
- Deals with all common plug-in loader issues
  - Demand loading automatically taken care of
  - Document specific loading automatically taken care of
  - UI placement automatically taken care of
  - Program paths automatically taken care of
  - .NET Load context issues automatically taken care of

How it works?

AutoCAD Autoloader Component – Mixed Mode ARX, DBX

Static folders are processed for Use cases:

- On Startup
- On Appearance
  - **When an application is installed while the host application is running**
  - **App Store is web based.**

Each .bundle has its PackageContents.xml processed

- Depending on the settings the Components are loaded
  - **Detects DLL dependencies and adds the module path to the system PATH to avoid DLL hell.**

On Shutdown and next Startup, application entries are cleaned

- Allows for simple delete of the program files.

The Solution

AutoLoader Component – PackageContents.xml

The root of the .bundle contains an XML file

- PackageContents.xml
- Describes the application
  - How it loads and runs
  - Which OS, Language and Version it was designed to run on
  - How it's dependencies are loaded and ordered
  - How the UI loads
  - How Support Paths should be setup
  - Version information
  - And more…

The Solution

AutoLoader Component – PackageContents.xml

PackageContents.xml contains

- ApplicationPackage element - only one per package
  - **CompanyDetails element – only one per package**
  - **Components element – 1-n processed top to bottom**
    - **ComponentEntry element – 1-n processed bottom to top**
      - **Commands element – only one per package**
        - **Command element – 1-n**

The Solution

AutoLoader Component – PackageContents.xml

PackageContents.xml contains

- ApplicationPackage element - only one per package
  - **Base element, attributes describe the product as a whole**
  - ***CompanyDetails element – only one per package***
  - ***Components element – 1-n processed top to bottom***
    - ***ComponentEntry element – 1-n processed bottom to top***
      - ***Commands element – only one per package***
        - ***Command element – 1-n***

The Solution

AutoIoader Component – PackageContents.xml

PackageContents.xml contains

- *ApplicationPackage element - only one per package*
  - ***CompanyDetails element – only one per package***

- **Components element – 1-n processed top to bottom**
  - **Logical Componment of your product**
    - **A set of DLL's or files that are related**
    - **A related product entry, maybe the Vault part of your product**

  - ***ComponentEntry element – 1-n processed bottom to top***
    - ***Commands element – only one per package***
      - ***Command element – 1-n***

The Solution

AutoLoader Component – PackageContents.xml

PackageContents.xml contains

- *ApplicationPackage element - only one per package*
  - ***CompanyDetails element – only one per package***
  - ***Components element – 1-n processed top to bottom***

- **ComponentEntry element – 1-n processed bottom to top**
  - **A file entry for your product**

  - ***Commands element – only one per package***
    - ***Command element – 1-n***

The Solution

PackageContents.xml RuntimeRequirements XML Element

Special element for describing the requirements of the section

Can be the child of the ApplicationPackage

- Defines the .bundle overall RuntimeRequirements
  - Used for performance reasons

Can be the child of a Components element

- Defines the Components overall RuntimeRequirements
  - Used to separate parts of the product to specific platforms

Can be the child of a ComponentEntry element

- Allows finer control over a ComponentEntry inside of a Components section
  - Feature in Sp1 crashes, but in sp2 it works fine

Questions?

…before Demos