

# *Windows PowerShell Modules: Patterns & Practices*

Dan Harman

Senior Program Manager, Microsoft

# Design Goals

Ability to organize PowerShell code into reusable, self-contained units

Module packaging

Provide an isolated execution environment with separate state

Session state

Control import of named members to avoid polluting the global session state

Module cmdlets

Attach metadata to describe module functionality and control loading behavior

Module manifests

Ability to repackage and abstract other modules to create custom solutions optimized for a specific task

Nested modules

# Scenarios

## ⦿ Libraries

- Export a cohesive set of functions for doing common tasks
- Similar to .NET classes with public functions and private variables
- Typically share a small number of nouns
- Example: BITS module for performing file transfers

## ⦿ Configuration

- Customize caller's environment with variables and functions
- Similar to profiles, dot sourcing, and customized environments
- Typically export multiple nouns and configuration variables
- Example: PowerTab customized tab completion and utilities

## ⦿ Repackaging

- Repackage and/or improve upon existing functionality
- Present a customized interface optimized for specific tasks
- Typically abstract other modules and re-export a subset of members
- Example: AD replication manager that wraps LDAP & AD core modules

Perl  
Python  
Ruby

PowerShell

# Defining a Module

Wide dynamic range philosophy

Script  
(.psm1)

Binary  
(.dll)

Dynamic  
(in-memory)

No file

Single file

Collection of files

Without module folder

With module folder

Without manifest

With manifest

# The Experts Conference

*Explore the Outer Limits.*

**DEMO**

Presented By:  **QUEST  
SOFTWARE**

Sponsored By: **Microsoft**

# Appendix

# Tips & Tricks for Script Modules

- ⦿ Variables can be shared between functions
- ⦿ `$PSScriptRoot` is the location of the module
- ⦿ Non-terminating errors work in a module
- ⦿ Modules imported into the root script module become nested modules
  - Hidden from the user by default
- ⦿ Can hook module unload
  - `$MyInvocation.MyCommand.ScriptBlock.Module.OnRemove`
- ⦿ Can XCopy themselves to avoid trust issues locations

# Module Manifest Processing Order

1. Validate Module Manifest
2. Check RequiredModules
3. Process RequiredAssemblies
4. Load Types & Formats
5. Load NestedModules
6. Load ModuleToProcess
7. Add to Module Table
8. Process Exports