
Improving
the Value of
PowerShell Scripts
Through Intelligent
Automation
PS C:\>_
ActiveBatch
ADVANCED SYSTEMS CONCEPTS INC.

For many in the world of IT, PowerShell is *the* language of automation. As the most popular scripting framework for Microsoft products, PowerShell is used to define jobs and tasks within not only Windows, but also scores of other Microsoft products.

PowerShell is designed to work with many of Microsoft's most important services and components, including Active Directory, Exchange, and Skype for Business. The recent open source debut of PowerShell adds even more versatility, enabling the crossover of PowerShell scripts—and those who specialize in scripting for the Microsoft ecosystem—between Linux, OS X and other platforms.

That kind of longevity and integration gives PowerShell a permanent place in IT's bag of automation tools. It also explains why most IT organizations retain old PowerShell scripts and continue creating new ones. The repository of existing scripts is one major incentive. Since PowerShell was introduced in 2006, it has quickly become a "go to" scripting language. Many enterprises have a huge investment in scripts that work well and are in daily use.

Another reason is that programmers are comfortable with PowerShell. People tend to stick with what they know, and change is always a challenge. The .NET framework upon which PowerShell is built is familiar territory for those steeped in Microsoft products, making the shift to other solutions, even versatile and efficient ones, an uphill climb.

Finally, IT departments often have a specific use case that requires a PowerShell script. Some applications accept PowerShell cmdlets easily, prompting the shell's use. In other cases, such as updating or revising configurations created primarily in PowerShell, it only makes sense to return to the framework.

Using PowerShell Doesn't Have to be an "All" or "None" Decision

Yet for confirmed PowerShell developers, it's possible to gain an immense amount of flexibility—not to mention productivity—by adding a workload automation application. A modern automation solution doesn't have to end the use of PowerShell; in fact, the capabilities of such solutions can actually enhance PowerShell scripts.

Simplification may be the best reason to consider an intelligent automation platform. **According to Gartner, the average IT organization uses between three and eight different automation tools.** Applications, operating systems, and web hosting services are all typically equipped with schedulers and resource managers. The exceptional interoperability of a top-notch intelligent automation platform gives IT the "single pane of glass" it needs to create, view, modify, and monitor all its scripts and executables—even those in PowerShell.

Another advantage is the ability to easily enhance the utility of a PowerShell script without adding significant lines of code. By using the pre-built, drag-and-drop job steps in an intelligent automation platform, application developers can quickly move through the development process. Job steps can be assembled in concert with PowerShell and/or other scripting languages as easily as erecting a Lego sculpture.

Using this hybrid approach, coding is reduced to only proprietary business items or functionalities. In many cases this allows development work to be handled by operations personnel, analysts, or more junior IT staff. The organization's most experienced developers are freed up to handle more demanding tasks.

How Intelligent Automation Adds Value to PowerShell

Intelligent automation platforms are designed to improve productivity not only for the developer, but also for the IT organization and even the entire enterprise. The power and interoperability of these tools are formidable. However, the ultimate goal of a centralized automation platform is not to force the elimination of PowerShell scripting; it is simply to provide a quick and easy path to pre-built, pre-tested functionality while minimizing additional lines of code.

For example, if time is an issue and a task requires a new or enhanced PowerShell executable, intelligent automation can dramatically ease workloads while actually improving effectiveness and efficiency. The library of pre-built job steps in an intelligent automation platform provides drag-and-drop simplicity, faster time-to-completion, and proven reliability. All the designing, writing, testing, and debugging is done; just move a PowerShell script into the automation platform, select the appropriate job steps that will add upstream and/or downstream functionality, and the task is complete.

Many online services now contain repositories of PowerShell scripts that are free to download. These off-the-shelf scripts can be loaded into an intelligent automation platform for enhancement. If an existing script requires dependencies or variables, the automation job can serve as a "wrapper" that contains triggers, constraints, and/or other dependencies, as well as alerts and monitoring instructions. With intelligent automation it's easy to pass variables from one step to another; data can also be shared between PowerShell and non-PowerShell objects as desired.

Automation solutions can work alongside PowerShell scripts with full interoperability. It's easy to execute pre- and post-actions for PowerShell scripts; say, for instance, a task involves running a report on a database backup before the job executes. Instead of searching for a PowerShell script for the task, simply pull "create report" or "database backup" from the automation platform's content library of pre-built functions and attach it to the PowerShell executable.

ActiveBatch and PowerShell: A Powerful Combination

ActiveBatch®, from Advanced Systems Concepts, goes beyond other advanced automation platforms to provide PowerShell, and PowerShell users, the best experience available. ActiveBatch supports PowerShell at no additional cost through its Integrated Jobs Library. The powerful Job Steps within ActiveBatch empowers PowerShell scripts with a wide range of new capabilities:

- **Object Passing.** Normally PowerShell objects are accessible only with the script itself. With ActiveBatch, you can pass data from one PowerShell script to another or even to a non-PowerShell job step. This type of data passing is uniquely provided by ActiveBatch and further enhances the capability of PowerShell.
- **Enhanced Security.** Only authorized users can edit or change a job, including those involving PowerShell scripts.
- **Audits and Change Tracking.** Once a PowerShell script is moved into ActiveBatch, the platform's audit capabilities allow IT to see who made changes to the code, as well as how it was modified. ActiveBatch allows reviewers to compare any two versions; they can also roll back to any previous version, no matter how many have been created.
- **Fast Script Editing.** ActiveBatch's script editor allows developers to edit scripts within the interface, then trigger them to see the results live. It also highlights syntax errors and is equipped with find/replace and variable auto-completion features.
- **Granular Scheduling.** Scripts can be triggered to run based on events like emails, file drops, network events, and WMI events. Complex date scheduling can be implemented using Gregorian and fiscal schedules as well as date arithmetic; a wide range of other events can also be used to trigger workloads.
- **Implicit/Explicit Remoting.** ActiveBatch supports agentless remote execution of jobs on machines not controlled by a PowerShell script. If PowerShell exists on one machine, ActiveBatch will execute instructions for a second machine even if it cannot be managed by PowerShell.

Flexibility for the Next Phase of IT

According to a recent Gartner study, IT is entering a new phase—one in which two levels of automation are required. The first uses workload automation in a more traditional sense for scheduled, mission-critical jobs; these jobs have led to the proliferation of application- or system-specific job schedulers most IT organizations see today. The second requires support for jobs or tasks that must be released continuously into production to follow the increasing rate of change in the modern enterprise.

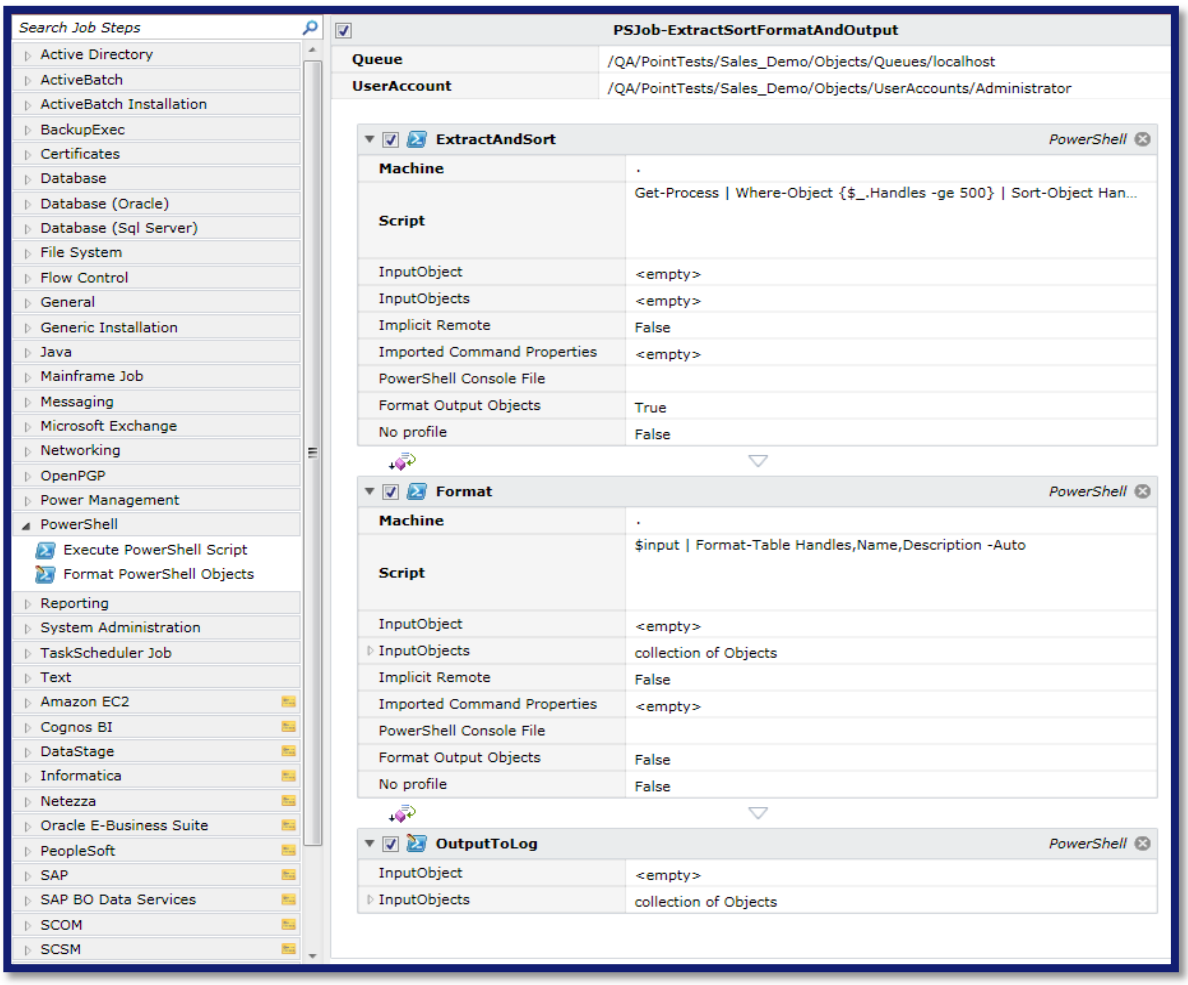
Dave Aron, Gartner Fellow and research lead for the annual Gartner CIO Agenda survey, reports that 66% of CIOs believe staffing will continue to be the number one IT challenge over the next five years. Budgets are flat in many IT departments and the shortage of skilled developers is acute. The situation calls for solutions that can cut workflow development times dramatically.

When used in conjunction with PowerShell, ActiveBatch can significantly reduce scripting time—in many cases by 50% or more.

In fact, many organizations have found that once they embrace intelligent automation they discover a new path to productivity; instead of creating processes manually and then automating them, the strategy is to design the process from the start through automation, thus bringing new processes to fruition faster and more reliably.

Regardless of whether developers are skilled in one language or many, ActiveBatch allows them to automate and manage multiple systems effectively. Moreover, the centralized "single pane of glass" nature of ActiveBatch allows IT to stretch its staff by directing all of its automation to one area, with the ability to easily pass data and instructions between machines—even those that are disparate or in remote locations.

PowerShell, Oracle, SAP, SQL Server, Informatica, Amazon—it doesn't matter. The multi-language nature of ActiveBatch allows PowerShell users to speak to all these technologies. ActiveBatch understands and executes jobs accordingly.



PowerShell Job Steps in ActiveBatch®

Never Make a Choice Between Expertise and Progress

Developers who are steeped in PowerShell, or who are simply comfortable with PowerShell scripting, should never have to make a choice between expertise and progress. With the right automation platform in place, these professionals can gain a new level of efficiency without abandoning skills that are useful and proven.

ActiveBatch, as a solution that is script-language independent, can add security, collaboration, audits, versioning, remoting, and much more. It speeds job creation through its huge library of pre-built, pre-tested, drag-and-drop job steps. It provides the versatility and connectivity modern enterprises require, and extends the knowledge and resources already in place. As a cornerstone of automation, PowerShell is certainly valuable. But with ActiveBatch, PowerShell can be an even better tool—one IT organizations will be able to depend on now, and far into the future.

Learn More Today!

Case Studies
Learn how global recruiting leader, SThree, drives business efficiency with end-to-end automation

Blog
To Script or Not To Script: That Isn't The Question

Videos
Demand More From Your IT Automation!

Visit us at ActiveBatch.com

Advanced Systems Concepts, Inc. is a leader in the development of Workload Automation and Enterprise Job Scheduling software. Our software simplifies the automation and integration of business and IT operational processes across single or compound workflows that can share data and manage dependencies. Automating as well as integrating workflows improves resource utilization, increases IT agility, and ensures higher service levels.

ActiveBatch® Job Scheduler and Workload Automation Solution provides IT organizations a solution that is designed to more easily accommodate change to your workloads based on business, technology, and regulatory requirements as well as decrease cost, mitigate risk, and reduce complexity.

Trusted by 2000 customers in 49 countries around the world, ActiveBatch addresses the automation challenges of cross platform computing environments that are expanding in size and complexity.

Visit **www.ActiveBatch.com** or call **+1 (973) 539-2660**
to get started with a Personalized ActiveBatch IT Consultation