Realtime publishers

Best Practices for Workload Automation in the Bimodal Era

sponsored by

ADVANCED SYSTEMS CONCEPTS, INC.

Dan Sullivan

Copyright Statement

Chapter 2: The Anatomy of a Workload Automation Solution

Workload automation has emerged as a solution to a long-standing but constantly evolving need in IT: the ability to perform tasks repeatedly, consistently, and reliably. Batch processing, the simplest workload processing model, is a staple of IT operations and has existed almost as long as IT itself. The applications and tools used to create and run batch jobs in the past are insufficient for today's workloads, which are driven by dynamically changing business requirements.

This chapter considers the composition of a modern workload automation solution, focusing on:

- A brief history of automation from batch scheduling to workload automation
- Key requirements of workload automation tools
- Advantages of workload automation
- Best practices of workload automation

Throughout the discussion, business requirements will keep appearing as a key influencer of workload automation.

A Brief History: From Batch Scheduling to Workload Automation

Businesses and other organizations execute their functions through a series of processes. Banks manage the movement of funds across accounts. Manufacturers fill orders from customers. Retail businesses stock and sell products. In many cases, there is a physical component to operations, such as manufacturing a part or packaging a product for sale. In other cases, the processes are more abstract, such as moving funds between accounts in the case of financial institutions. In all cases, however, information processing is an essential part of the business process.

Workload automation is the coordination and execution of a potentially complex array of business operations focused on information processing. The earliest form of workload automation was batch processing.

In the Beginning: Batch Processing

Early information processing operations tended to focus on bulk operations needed in large organizations. In business, early computers were used for accounting, payroll, and other data-intensive (by standards of the time) operations. Similarly, governments would apply computers to data-crunching tasks, such as calculating census statistics. These types of operations shared several characteristics:

- Time-based processing
- Relatively stable set of jobs
- Reliable custom scripts

Closing the books and running payroll are the types of operations that run on a regular schedule. Essentially, there is no unpredictability in these tasks. Fixed, time-based processing was typical with early batch jobs. This setup was a significant advantage to application developers and system operators. The tasks that were run last week or last month would be run in the same order this week or this month. There was no need to account for complex logic that would trigger the execution of some processes and not others.

The time-based processing that characterized early batch jobs meant that businesses could depend on a fairly stable set of jobs. Only so many business operations were automated, and they tended to address core business operations, such as managing revenues and payments. These types of business operations are stable and do not change much. In fact, they do not change much today thanks to well-defined and agreed-upon accounting procedures.

With a relatively stable set of jobs comes a stable set of scripts to run those jobs. After programmers write and debug the code behind a batch job, it can continue to run for years without change—as long as business requirements do not change. There may be some minor changes in the business logic of an operation—for example, a change in payroll law that requires a change in the way the process performs a calculation—but these do not necessitate large restructuring of code or development of new batch jobs.

This description of batch processing sounds almost mythical in the way it describes a time when simplicity was a defining characteristic of business operations. Of course, developers had to deal with complex issues, but they centered more around working with scarce computational resources and the lack of modern development tools. Managing workloads, once developed and debugged, however, did not entail the challenges IT professionals face today.

Business Changes Drive Changes in Automation

Business processes today change more frequently than in the early days of information processing. Twenty to fifty years ago, businesses could plan months, quarters, and years ahead with reasonable confidence that they could foresee the major characteristics of near future markets, competitive landscapes, and customer expectations.

Changes in Business Require Changes in IT

The pace of business change has accelerated significantly. In the era before online banking and e-commerce, product offerings and competitive landscapes changed more slowly than they do today. Businesses had more time to react and adapt business operations to new opportunities. For example, bank deregulation in the 1990s created business opportunities for financial institutions, but those institutions had long periods of time to prepare for such changes.

Businesses now operate in a well-connected, online, and interactive environment. Customers have become accustomed to around-the-clock access to business services online. Do you feel like paying bills late at night? No problem, your bank's bill pay service is available online. Do you need a last-minute gift for a friend's birthday? No problem, you can place an order and have it delivered the next day.

Consumers are not the only ones growing accustomed to instant access to services and information. Self-service portals are becoming very common in the B2B world. If you need to track orders, review account statuses, or get reports of past activity, you can access this information yourself through a customer-facing portal.

If we think about what must be going on behind the scenes of these operations, we can pretty quickly see that the characteristics of batch processing no longer hold (see Figure 2.1). In particular, these new operations are not:

- Time-based processing; they are triggered in response to external events
- A relatively stable set of jobs; they are changing according to new services needed to perform new business operations or support customer needs
- Supported by stable, custom scripts that change infrequently; they require new programming as well as continuous maintenance and bug-fixing

The collective impact of these differences requires a change in the way IT operates. Processes that worked well in the fairly stable world of batch processing are insufficient to meet the dynamic needs of today's business environment.

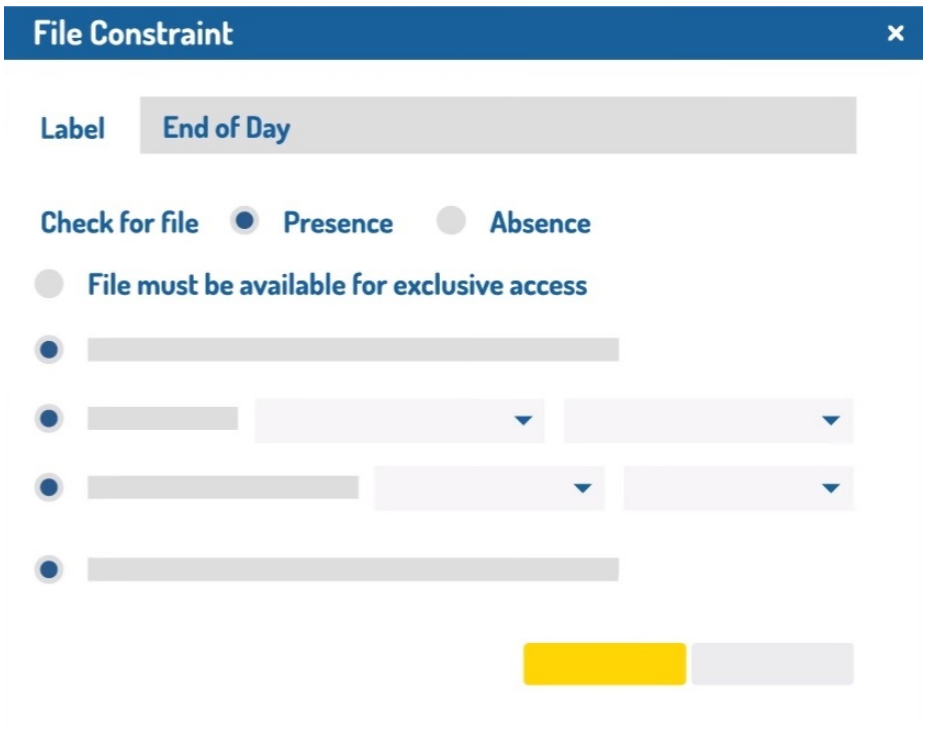

Figure 2.1: Time-based batch processing is too limited for today's workload management requirements, which must be capable of responding to a wide array of event triggers and file constraints.

The Limitations of Custom Scripts for Workload Management

In particular, custom scripts fail to keep pace with change. It is a relatively straightforward process to write scripts. Developers do it all the time. Problems arise for three reasons:

- Dependency on limited set of skills
- Custom scripts are difficult to modify
- Custom scripts are prone to errors leading to job failures and the need for debugging

Each of these reasons is difficult enough to address on its own, but these problems often emerge at the same time, compounding their adverse impact.

In addition to this, one of the biggest challenges IT managers face is finding and retaining skilled staff. IT positions go unfilled and business requirements go unfulfilled. Existing staff members are asked to continually do more with less resources to meet demands. One of the needs that can get overlooked is the development of custom scripts to perform a new set of business tasks in a controlled, repeatable, and dependable way.

In those cases where an individual develops custom scripts, those individuals are typically also the ones who have to maintain them. When developers are expected to write scripts on short schedules to meet a very specific requirement, it is no surprise to find the resulting script is highly tailored to the existing requirements. As new requirements emerge, the scripts must be modified. If the developer did not have time to anticipate the types of changes that might be needed in the future, their scripts may be difficult to adapt to the new requirements. In fact, it is not uncommon in such cases to simply start from scratch and write a new script to meet the new requirement. This cycle creates a proliferation of scripts that IT developers must continually maintain.

In the past, when businesses ran a limited set of batch jobs, there was time to develop and debug code that one could reasonably expect to run for long periods of time. After programmers wrote a batch script, the primary focus turned to making sure it ran as expected and addressing problems as they arose (e.g., running out of storage space for an unusually large set of data). Today, a developer might write a script and then turn her attention to writing another script or to debugging one that a colleague wrote several months ago that no longer works because of changes in an upstream process that generates input for the now problematic script.

At this point, there is an unsustainable set of IT practices that can no longer meet the needs of businesses. This argument is not a criticism of past practices but a recognition that business demands on IT are changing. Custom scripting and batch processing is the best practice when an organization needs to implement a small set of relatively stable, repeatable processes. As the need for the number of custom scripts grows to meet emerging business needs, the demands on IT grow in unsustainable ways. IT departments must change the way they meet workload needs to leverage their staff skills, minimize maintenance, and ensure reliability of operations. This reality creates the need for workload automation.

The Emerging Demand for Workload Automation

Workload automation is an IT practice that leverages programs and processing frameworks to execute custom workloads without requiring custom solutions for each new business task. In particular, workload automation:

- Reduces the need for custom scripts
- Leverages a library of code to execute parameterized jobs
- Reduces the need for custom error handling and alerting logic

Workloads have common requirements and workload automation tools take advantage of those similarities to provide tools that reduce the time-consuming and error-prone parts of custom script development.

Reducing the Need for Custom Scripts

If you were to look at tens or hundreds of custom scripts, you would see common patterns. These patterns include:

- Definitions of constant values, such as names of working directories and control parameters
- Specifications of input and output files
- Checkpoints to ensure input files exist and checkpoints to warn if an output file already exists and may be overwritten
- Commands to invoke programs to process data
- Commands to write information about the state of the job to a log file
- Post-processing operations to finalize output so that it is ready to be used by the next workload script

It would seem that custom scripts are not really needed after all if we can provide another mechanism to implement common features, such as specifying inputs and writing log data. Workload automation tools are designed to provide a framework that allows developers to rapidly configure workload automation processes without having to implement operations at the low levels of abstraction needed in custom scripts.

Leverage Library of Code to Execute Parameterized Jobs

One of the ways workload automation tools can help developers is by providing code for performing common tasks (see Figure 2.2). Libraries of code offer several advantages, in particular, the code is typically:

- Designed for reuse
- Debugged
- Optimized for performance
- Easily integrated with other components of the workload automation platform

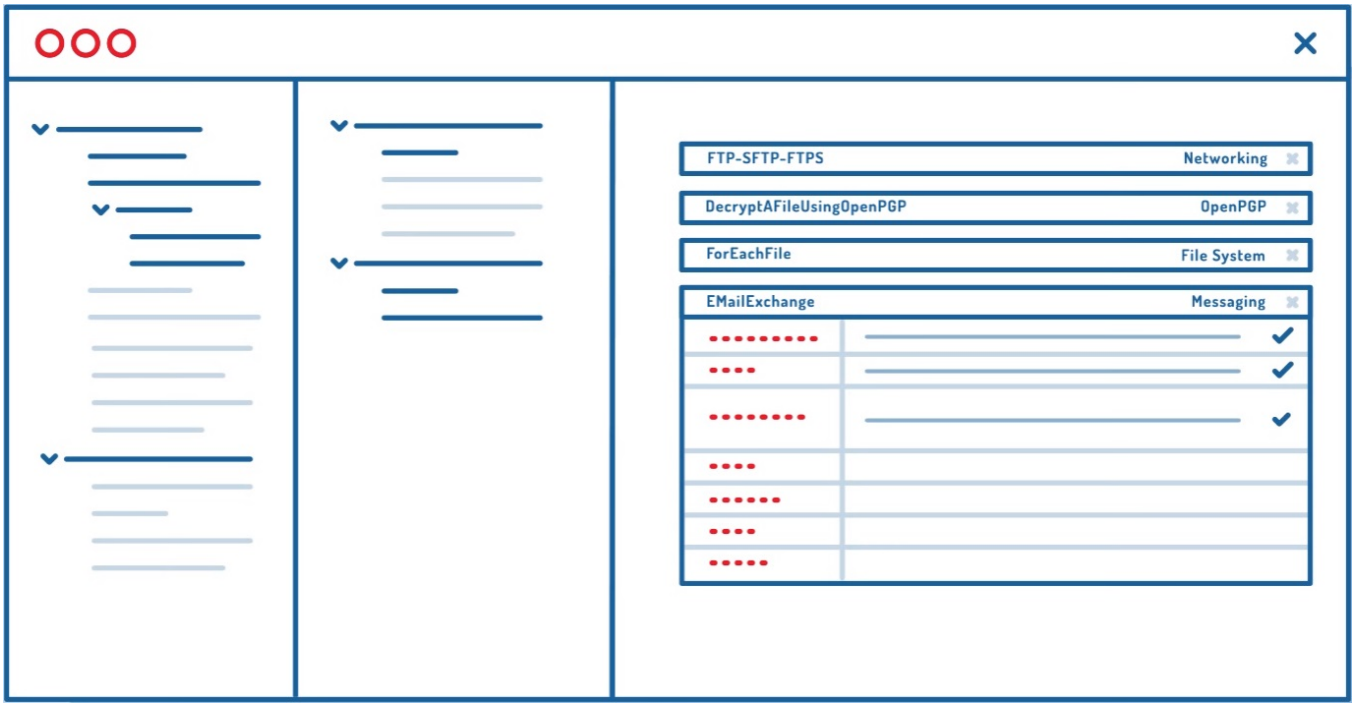

Figure 2.2: Example library component for performing secure file transfer.

For example, a component for specifying an input file could accept a filename as a parameter and perform a consistent set of operations, such as:

- Verifying the file exists
- Ensuring the file is of the correct format
- Locking the file at the operating system (OS) level to prevent other processes from modifying it
- Writing metadata about the file (e.g., filename, size, file type) to the workload log file

Libraries with a rich set of components can help developers create processes faster by eliminating, or at least reducing, the overhead of implementing commonly needed operations and logic. For example, Figure 2.3 shows a drag-and-drop workflow designer where users can simply pull from a content library of job steps on the left to easily create complex workflows.

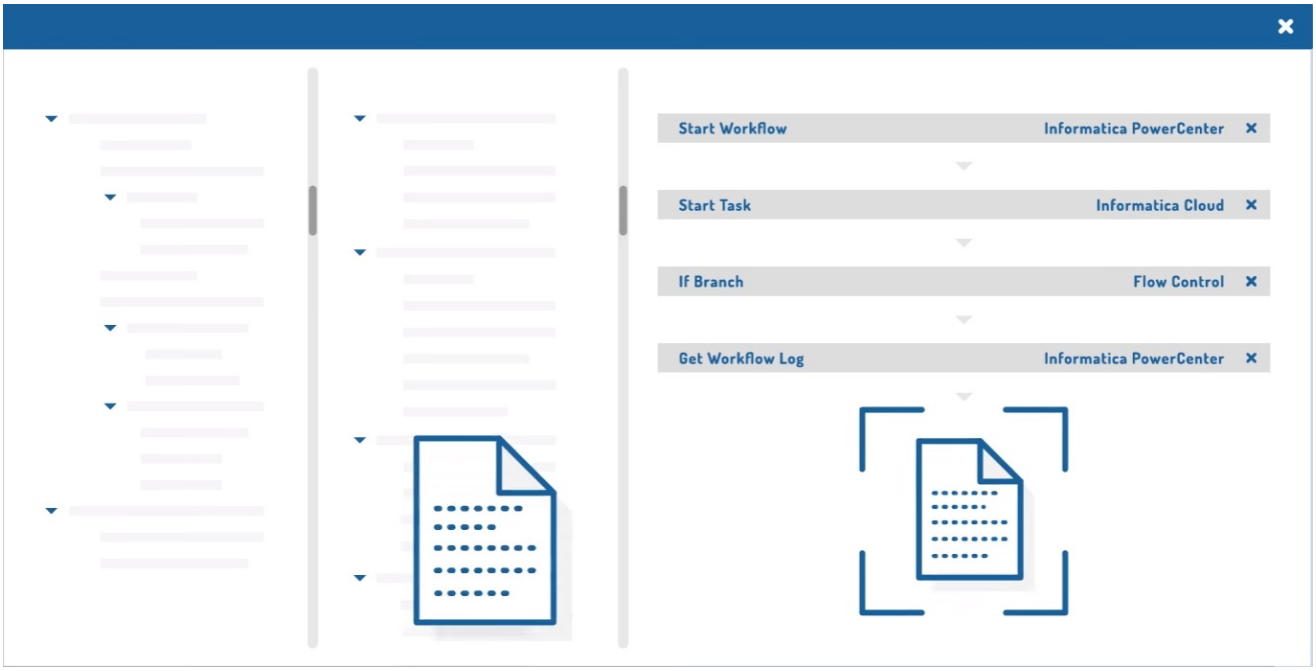

Figure 2.3: Example of a content-rich workflow designer.

Reduces the Need for Custom Error Handling and Alerting Logic

In addition to streamlining the development of individual processes, workload automation tools help businesses manage collections of these processes (see Figure 2.4). In particular, by using a common set of error-handling and alerting logic, system operators can use a single, centralized reporting and alerting tool to monitor the state of multiple workloads. Businesses that depend on custom scripts can find themselves developing individualized procedures to differentiate the error logging and alerting of those scripts.

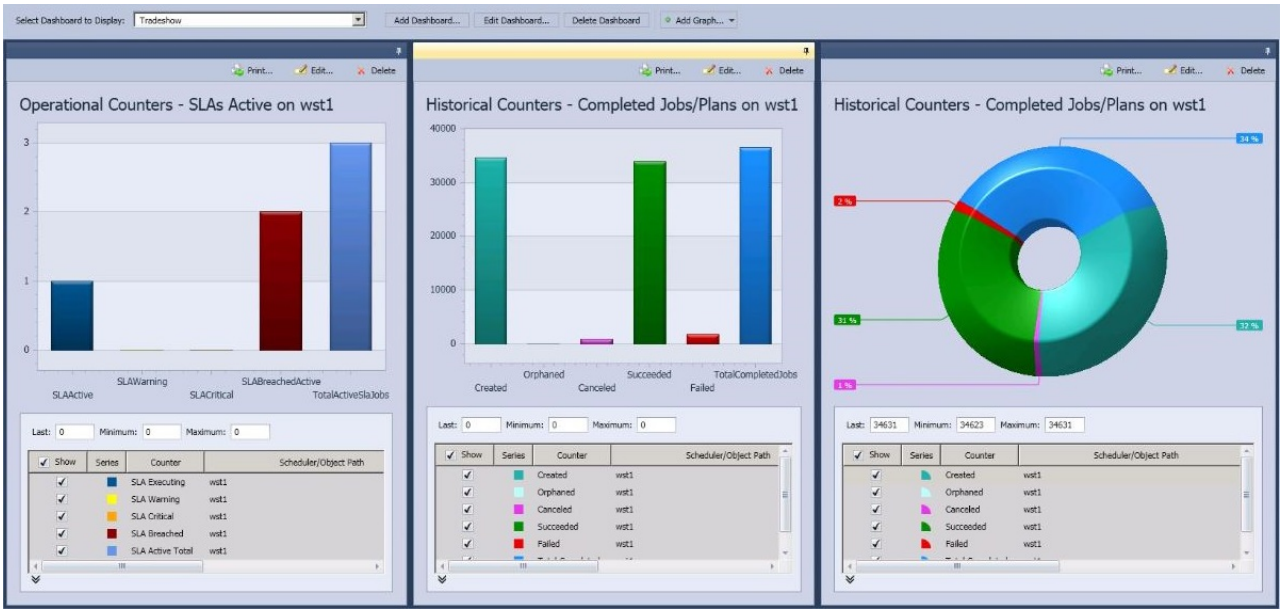

Figure 2.4: Dashboards in workload automation can provide a consolidated view of the state of multiple workloads.

A multitude of factors contribute to the value of workload automation solutions. No one feature or small set of features realizes the full potential of workload automation, but a solution that supports a full array of key requirements can go a long way to radically improving workload management operations in a business.

This section highlights some of the most important features to look for when selecting a workload automation tool:

- IT organizations depend on a heterogeneous mix of server platforms and operating systems. Workload automation tools should support the most popular platforms, including:
    - Windows
    - Linux
    - UNIX
    - OpenVMS
    - iSeries
    - z/OS

  Not every organization will have all of these platforms, but you should look for an automation tool that supports all of your operating systems. Leaving even one OS out of workload automation can diminish the overall value of the effort:

- Workload automation tools should support multitenant, secure environments. For example, automation tools should ensure plans and jobs are protected and isolated from editing and viewing by specific users or groups depending on access control permissions.
- Documenting code is a best practice that is sometimes left unfinished. This situation is especially problematic with custom scripts and is one of the reasons it is so difficult to maintain and debug custom scripts. Workload automation tools can help by generating self-documentation that is readily understood by developers.
- Business operations are subject to a range of regulations and policies. It is important that a workload automation tool support compliance and governance requirements. This is especially the case when businesses must demonstrate to auditors or other departments they have implemented procedures and have controls in place to protect the integrity of those procedures.
- A self-service portal is a necessary feature for any organization with multiple workloads administered by different groups. Just as there are limited staff resources to write custom scripts, there are limited system support personnel to perform routine tasks that can be done by end users.

- As noted earlier, time-based batch processing can address the needs of a small set of workload automation requirements. Event-driven automation is an essential feature of modern workload automation. Common events can trigger workload execution:
    - Receipt of an email
    - A message delivered in a message queue
    - System startup or shutdown
    - Web service execution
    - Database trigger

  Robust workload automation tools have mechanisms to detect and respond to events across the IT ecosystem. Also look for automation tools that can balance workloads. You might find, for example, that some servers are running at capacity while others are underutilized. The ability to move jobs among resources can help optimize workload execution (Figure 2.5).

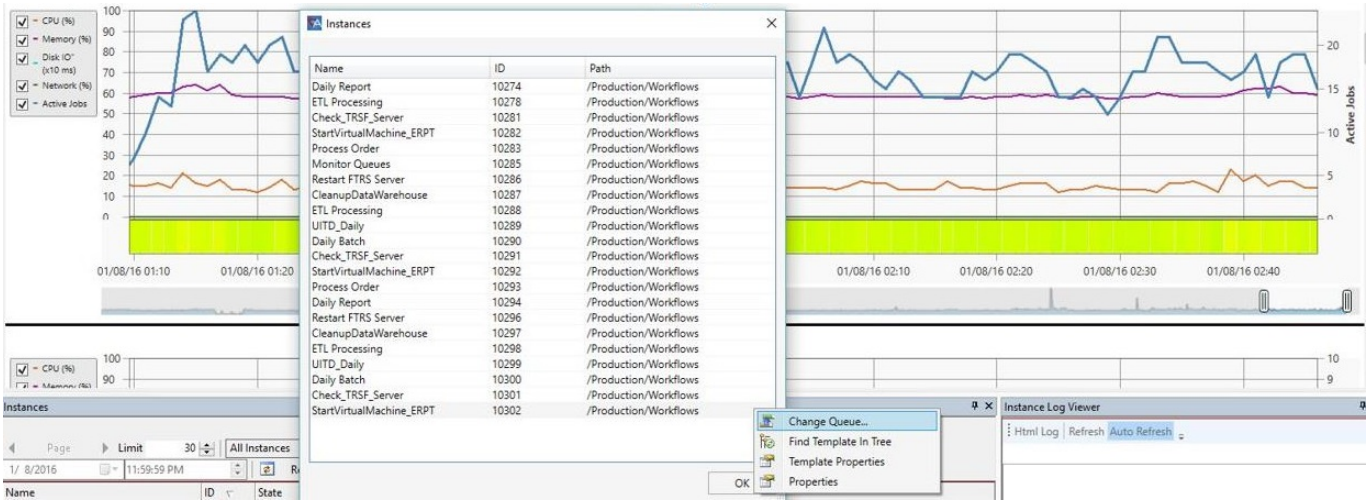

Figure 2.5: Workload automation tools should allow administrators to move jobs across server resources to execute workloads efficiently.

- The contents of the workload automation library strongly influence the overall value of the platform. Libraries should include tools for common functions:
    - Managing files and directories
    - Transferring files between servers
    - Logging workload information
    - Integrating with databases, enterprise resource planning (ERP), customer relationship management (CRM) and other enterprise applications
    - Working with legacy scripts
    - Invoking API functions

- Workload management processes are defined in software and should be managed like other software assets. Look for change management support tools, including versioning and the ability to roll back to previous versions of scripts.
- Consider the way system support personnel and end users will interact with the system. Mobile access is especially important for receiving alerts about unexpected events as well as monitoring the status of workloads.
- Workload automation tools should provide a holistic view of jobs executed in the platform. Reporting and analytics should be available to allow users and workload administrators to understand overall usage patterns and demands on resources as well as verify compliance with policies and procedures.

This collection of key features is not arbitrary but constitutes the set of features that provides the collective advantages of workload automation.

Advantages of Workload Automation

Workload automation addresses several common challenges faced by businesses responding to the dynamic demands of customers, business partners, and the market in general. The benefits of workload automation stem from four core advantages:

- More reliable job execution
- Reduced demand on limited skills (improved use of limited skill sets)
- Improved reporting, alerting, and consolidated management
- Faster deployment of new business processes

Workload automation vendors invest design and development resources to produce platforms that provide a robust execution environment for workload processing. Developers writing custom scripts typically do not have the time to invest in high-reliability features that may be used only in a single script. Vendors developing specialized workload automation tools, however, have a different set of economic constraints and incentives, and it is in their interest to invest in high-reliability features.

Another advantage is the reduced need for specialized skills. You do not need to find an expert PowerShell developer or Bash script programmer each time you need to deploy a new workload or modify an existing process. This idea is also important from an opportunity-cost perspective: Even if you have the skilled staff on hand to write custom scripts, is that the best use of their time?

IT managers will appreciate the consolidated reporting, alerting, and management that comes with a workload automation tool. This functionality is especially useful when self-service portals and mobile device access are provided. Perhaps the biggest advantage of using workload automation platforms is that they provide the ability to deploy new business processes faster. As with any software tool, it is the combination of quality software and best practices that help businesses realize the most benefit from their investment.

Best Practices in Workload Automation

Over the long history of batch processing and more recently workload automation, IT professionals have developed a set of best practices, including:

- **Code Reuse.** Code reuse is a resource multiplier. If one developer writes a parameterized module for a common task, that module can be reused so that others do not have to reinvent the proverbial wheel. Libraries also enable referencing of code instead of copying. The advantage here is that if the library is updated with a bug fix or performance enhancement, all users of the library code will benefit. If a developer simply copied and pasted a piece of code, there are two versions of the code that can drift out of sync.
- **Streamlined Installation.** You can leverage workload automation tools to ease installation and upgrades of software throughout the enterprise. What might have taken days in the past can be done in minutes using workload automation.
- **A Focus on Architecture.** Think in terms of integrated architectures and information ecosystems. Begin with an architecture that supports workload requirements across the enterprise and avoids silos of automation. Silos isolate code and lead to duplicated efforts.
- **Workload Automation Deployed Early and Often.** Implement automation when first starting a new business process. Do not treat automation as an add-on after the fact. It is difficult to overestimate the benefits of starting a new business process with workload automation in mind. This approach avoids, or at least reduces, the need for code that will be thrown away when the process is properly automated. It is likely that working with a workload automation tool will lead to a faster deployment than using a custom set of scripts. Remember, even simple scripts can have bugs, and the last thing any developer wants is to be debugging a script after it has gone into production.

- **Using Version Control and Other Software Management Practices.** Employ version control and lifecycle management procedures with workload automation. Workload automation scripts are valuable assets that need to be managed like other software assets.
- **Workflows Executed Based on Events.** Time-based automation is a small subset of the ways to trigger workload execution. Events are constantly occurring across an IT infrastructure. Many of these events warrant some kind of action, such as starting a workflow or reporting the event. Automating based on events, and not simply time, enables businesses to respond more rapidly and efficiently than if processes are queued to run only at predefined intervals.

To maximize the benefits of workload automation, be sure to consider best practices such as these as you map your strategy.

Summary

This chapter has examined the composition of a workload automation solution with a focus on framing workload automation in terms of the history of batch processing. Businesses will realize the optimal benefits of automation by choosing a workload automation tool that supports key requirements and combines that platform with a series of best practices.