ActiveBatch®

Customer IT Automation Success Story

Indiana University Foundation
Smooths Data Transfers with IT Automation

INDIANA UNIVERSITY FOUNDATION

Company: Indiana University Foundation
Industry: Not-for-Profit
Customer Site: Bloomington, Indiana, United States

Brief Company Overview:
Indiana University Foundation (IUF) is a not-for-profit corporation dedicated to maximizing private sector funding for IU. In fiscal year 2009-2010, Indiana University received more than $100 million in gifts from more than 100,000 individuals, corporations, and foundations. IUF manages an endowment of approximately $1.3 billion, administers about 6,000 gift accounts, and provides related fundraising services to IU and its donors.

INDIANA UNIVERSITY
FOUNDATION

SUCCESS STORY HIGHLIGHTS

- Integrating disparate applications
- Breaking free from manual monitoring
- Run jobs based on set conditions
- Get staff notified of errors immediately via email alerts
- The importance of reporting services

Managing $1.5 Billion in University Funds

Scheduling when various software applications on your campus should exchange data among themselves might not seem like the most glamorous of needs, but it's essential —and can waste large amounts of IT time and resources when it has to be done by hand.

Just ask Jay Sissom, manager of systems administration and customer support at Indiana University Foundation, a 250-person subset of Indiana University that manages more than $1.5 billion in funds for the university. The foundation's responsibilities include tasks like soliciting contributions from donors and investing and monitoring funds so that the university gets the best returns possible.

To keep everything in sync, Sissom's department runs 30 or 40 software jobs nightly to send information from one system to another. To do so, Sissom's 18-member IT staff must make sure that communication between various applications —the donor information system and the accounting system, for example— runs smoothly. Some of the applications, such as the general ledger and the donor information system, are from Datatel. The foundation also runs UNIX and Linux and has a number of custom applications written in-house, such as its investing system, which connects with both the GL and the donor information system and runs on Microsoft Windows and uses Microsoft SQL Server for its database.

That mix of operating systems and applications rules out a number of tools available for job automation because they are specific to a particular operating system. Instead, the foundation needed a system to handle job scheduling across several operating systems and disparate applications.

Also, although Sissom could schedule operating-system-specific jobs in advance through tools in the various operating systems, such as Windows Scheduler or Unix cron, it wasn't an efficient approach. "Those worked," he said, "but we had to build a lot of code to monitor the jobs, [since] those built-in tools don't provide any monitoring."

The Problem with Manual Monitoring

Given the mix of applications and operating systems, scheduling and tracking jobs at the foundation used to mean manually monitoring the data exchanges between applications —exchanges that typically take place at night, when computer resources are most available and users are least impacted. Often, one job could not start until another job has finished running correctly.

When jobs were scheduled manually, if problems were encountered overnight, the staff often had no way of knowing that until a user complained the next day. That could mean running jobs during the day, consuming considerable resources and makes users wait for correct data.

When jobs were scheduled manually, if problems were encountered overnight, the staff often had no way of knowing that until a user complained the next day.

The Solution

To solve the problem, Sissom turned to a tool from Advanced Systems Concepts called ActiveBatch Job Scheduler, which he and his staff have used for more than a year to schedule jobs in advance.

Instead of writing code and monitoring jobs manually, ActiveBatch provides "a nice GUI to schedule the jobs," Sissom said, one that allows jobs to run based on set conditions. If job A succeeds, for example, ActiveBatch can be set to run the next job. If a job fails, someone on Sissom's staff can be notified via email immediately. Eventually, he said, he hopes to add software that will skip the email notification for key jobs and phone a staff member immediately instead.

Reports are another plus for the system, since the software tracks what jobs ran when —or why a particular job failed to run. Partly because of ActiveBatch's reporting capabilities, Sissom said, he hopes to eventually run every scheduled job in ActiveBatch. With the reporting functions available, he will then be able to look in one place to see exactly which jobs succeeded and failed each night. "Whenever there are problems, we don't have to spend as much time researching it. ActiveBatch tells us the exact place the problem occurred," Sissom explained. "Using the GUI, we can go in and look at the log to see what the error messages are and correct the underlying program."

He'll also be able to schedule jobs right on top of one another, so that one can begin right after the preceding one finishes; that will allow more batch jobs to be run closer together over a single night.

"It's really improved our response to jobs that failed," Sissom said. "It's improved our uptime." Instead of waiting for users to complain that data hasn't been updated, he said, "In a lot of cases, we can fix problems and users won't even know there was a problem."

"Whenever there are problems, we don't have to spend as much time researching it. ActiveBatch tells us the exact place the problem occurred. Using the GUI, we can go in and look at the log to see what the error messages are and correct the underlying program."

Learn more at ActiveBatch.com

ASCI Advanced Systems Concepts, Inc.