Departures
TRAIN DESTINATION TIME PLATFORM REMARKS
AZ0101 ROME 07:30 8A DEPARTED
BA0553 LONDON 08:10 8C ON TIME
AF0904 PARIS CDG 08:22 8B ON TIME
KL3459 AMSTERDAM 08:30 9A ON
LH3857 COPENHAGEN 09:00 5C ON
SU0474 MOSCOW 10:20 7C ON
8 A A 9

Platform-Independent Task Scheduler

There are numerous reasons why an administrator or developer might want to schedule particular tasks to run automatically at a particular point in time or at a regular interval such as a nightly backup job, maintenance utility, or some kind of batch processing job. One common method for scheduling tasks is the UNIX or Linux cron job. However, certain types of tasks (especially those which require interaction with BBj® Services) would be more easily managed if BBj had a built-in scheduling feature and it would also remove operating system dependencies from an administrators deployment plan. Therefore, BBj 12 introduced a new scheduling feature to the already robust set of tools available in the BBj Enterprise Manager as well as the Admin API.

Two components make up the scheduling system: Task Groups and Tasks. A Task Group contains a list of one or more tasks to be executed synchronously in sequential order, when it should first execute, and how often the Task Group should repeat execution. A Task is a single item to be managed and executed by a Task Group. The best way to understand the scheduler is to simply walk through an example.

By Jeff Ash
Software Engineer

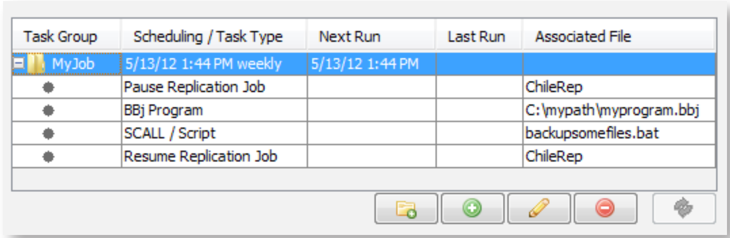
By Nick Decker
Engineering Supervisor

Scheduling Example

The example in this article illustrates what might occur when making a weekly backup of data files located on a replication target for zero downtime on the live system. Outside the BBj scheduler, this particular job would require additional Java or BBj coding to handle pausing and resuming the replication job but since the scheduler is designed with BBj in mind, it requires only a couple of mouse clicks to perform these tasks.

The Scheduling panel in the Enterprise Manager displays a list of all currently configured Task Groups and provides an interface for creating, modifying, and removing them. The panel shows the tasks within each group, when they will run next, and when last run. Access the Scheduling panel by selecting the “Scheduling” item from the Enterprise Manager navigation area.

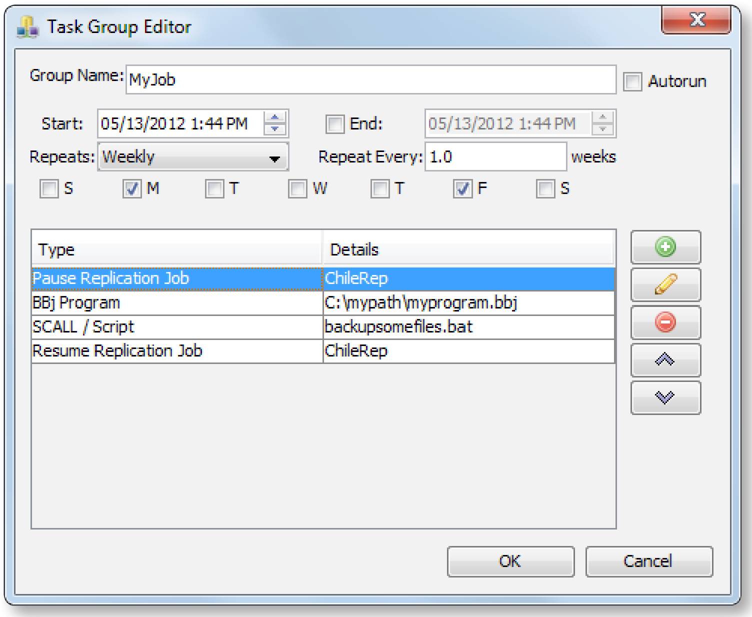
Figure 1 shows four Tasks that make up this Task Group. The first task pauses the replication job, which is not as easily done from something like a cron job. This ensures that the replication target is in a clean state and ready for backup. The second task executes a BBj program that could do anything needed to prepare the data for backup. Next, the scheduler executes a system call; in this case executes a Windows .bat batch file to perform the backup operation. Finally, the last task automatically resumes the replication job so that the target can catch back up with the source.



Task Group	Scheduling / Task Type	Next Run	Last Run	Associated File
MyJob	5/13/12 1:44 PM weekly	5/13/12 1:44 PM		
●	Pause Replication Job			ChileRep
●	BBj Program			C:\mypath\myprogram.bbj
●	SCALL / Script			backupsomefiles.bat
●	Resume Replication Job			ChileRep

Figure 2 shows the configuration of the task group which executes weekly, every Monday and Friday at 1:44 pm. Note that specifying an “End” date would cause the Task Group to cease executing at that particular point in time and a change to the “Repeat Every” field allows for skipping weeks.

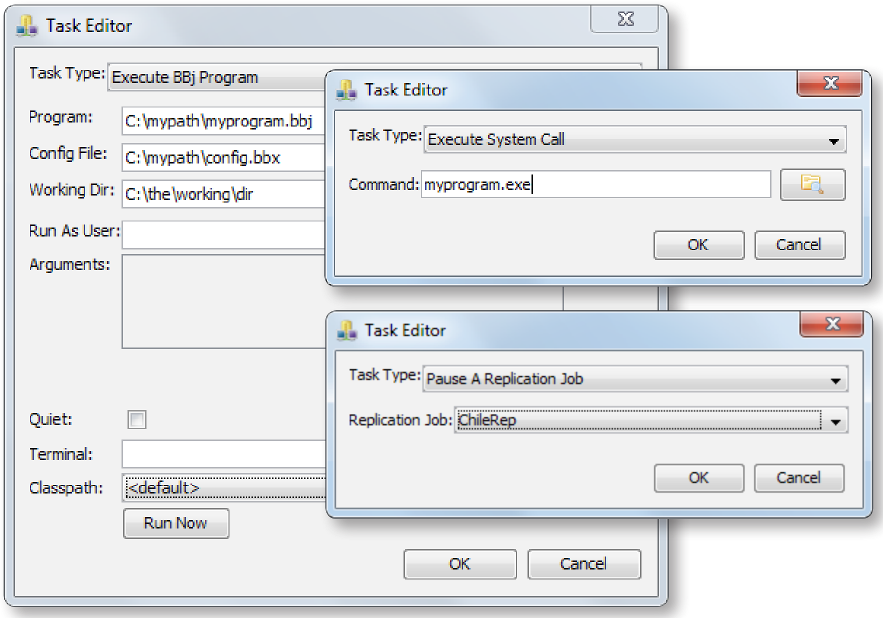
Four types of Tasks are available with the scheduler: Run BBj Program, Execute System Call, Pause Replication, and Resume Replication. Each Task type has a different set of configuration settings as shown in **Figure 3**.



Group Name: MyJob   Autorun: ☐
Start: 05/13/2012 1:44 PM   End: ☐ 05/13/2012 1:44 PM
Repeats: Weekly   Repeat Every: 1.0 weeks
☐ S ✓ M ☐ T ☐ W ☐ T ✓ F ☐ S

Type	Details
Pause Replication Job	ChileRep
BBj Program	C:\mypath\myprogram.bbj
SCALL / Script	backupsomefiles.bat
Resume Replication Job	ChileRep

Figure 2. Task Group Editor



Task Type: Execute BBj Program
Program: C:\mypath\myprogram.bbj
Config File: C:\mypath\config.bbx
Working Dir: C:\the\working\dir
Run As User:
Arguments:
Quiet: ☐
Terminal:
Classpath: <default>

Task Type: Execute System Call
Command: myprogram.exe

Task Type: Pause A Replication Job
Replication Job: ChileRep

Figure 3. Task Editor showing Run BBj Program (left), Execute System Call (top right), and Pause A Replication Job (bottom right)

Another Use Case

When we first designed the new BUI-based BASIS Product Suite Download page (links.basis.com/getbbj), one of our goals was to integrate it seamlessly with the automatic build system in the cloud. The BUI program accomplishes this by utilizing a custom BBj class that dynamically retrieves the available released and development builds from an Amazon S3 bucket (S3 is the nickname for Amazon's Simple Storage Service, which we think of as a large hard drive in the cloud). The end result is that the download page lists all available BBj downloads automatically, without any human intervention. The code worked perfectly, but subsequent performance analysis revealed that querying the cloud machine at runtime was occasionally slow. Amazon guarantees uptime for their servers and while access is typically very fast, various tests showed that querying the S3 bucket added anywhere from 0.5 to 5 seconds to the launch time of the BUI app.

Because we are obsessed with speed and making BUI as fast as possible, adding several seconds to the launch time was unacceptable, especially because new development builds typically occur just once per day. Since the files in the bucket did not change very frequently, constantly querying the bucket every time a user hits the download page would be overkill. If we could somehow query the bucket on a regular basis and save out the results to a local file on the server, then we would shave several seconds off the launch time.

BBj's new scheduling feature fit the bill perfectly! In just a couple of minutes, we set up a job to run a BBj program that retrieved the current list of available cloud builds and saved the information out to a file on the server. We configured the job to run every 15 minutes so the download page would never be out of date by more than a quarter hour. The job itself only takes a few seconds to run so the load on the server is negligible. We then modified the BUI download program to retrieve the list of available builds from the local file on the server, which occurs almost instantly.

The benefits don't stop there. Because we have replicated servers located around the world, the BUI app is served to customers from the server nearest to them. A user in Germany, for example, will be running the download page app from our server in Ireland, while we in Albuquerque get it from a server in California. All of this is transparent to the end user by way of geo-aware servers. Additionally, the replicated servers automatically get the latest set of available builds as the scheduler saves that information to a local file that is automatically pushed to all of the servers via BBj's replication ability. Lastly, Amazon's CloudFront content delivery network sends the desired BBj build to the user from the server closest to them, further streamlining the download process.

Summary

With the addition of a powerful platform-independent scheduler in the BBj product, administrators and developers have more power and flexibility at their fingertips. It is no longer necessary to use one or more third party schedulers to manage BBj backups or other business processes requiring execution at regular intervals. And finally, since the scheduler is built right into the BBj system, it makes certain interactions with BBj-related processes such as replication, much easier to configure and manage. ■

For further information regarding the refactor and optimization of the BUI-based BASIS Product Suite Download page, see *The Anatomy of a Web App Makeover* at links.basis.com/12webapp