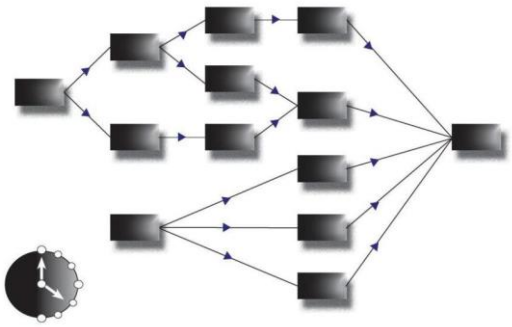
MultiBatch

White Paper



INSIDER TECHNOLOGIES

Table of Contents

Introduction

What Is MultiBatch

MultiBatch is a mainframe class batch scheduler product for the HPE NonStop server arena. It allows users to configure and automatically sequence any number of batch jobs to create an applications off-line processing flow.

This batch program execution can be monitored for start time, length of execution and successful and unsuccessful completion states. Based on this criteria, problems can be escalated to other management applications, or other batch jobs can be started.

The package comes equipped with a configuration GUI and/or Pathway interface, with a full set of reporting and management tools.

What This Document provides

This paper provides a technical overview of the MultiBatch product.

It describes the thinking behind the design of the product, using the products original installation to illustrate the principles behind this design.

The functionality of each of the MultiBatch software modules is described in detail.

Who Should Read This Document

The document is aimed at people with a technical background.

The document will provide an excellent introduction to new users at an existing installation, or to individuals who are considering a product evaluation and who are looking for a more detailed description outside of the information provided by Insider's product literature.

In the Beginning

In 1985, a major UK based organisation had committed itself to the Tandem (now called HPE NonStop) architecture and set about converting its existing OLTP computer systems to run on the Tandem platform.

At the end of each processing day there was a need to collect together updates made to the application database and to forward them to an external non-Tandem system for overnight processing. This Tandem application database was spread across multiple nodes. Later in the processing cycle, this external system would return updated information that needed to be redistributed around the Tandem network.

Timely and accurate batch processing was therefore crucial to the processing day and so the organisation compared its batch requirements against the software that was available at that time.

The requirements were:

- The product should make full use of the distributed architecture of the Tandem environment from the perspective of allowing parallel batch processing. This concurrent processing could be taking place on the same Tandem node or on other nodes within the network.
- Dependency should be allowed between jobs on the same node or between jobs on different nodes.
- There should no complex JCL to create and maintain.
- The batch schedule(s) should be capable of being monitored by an operator from a single console. This principle should hold true even if the schedule contained jobs executing on different systems - this concept is known as the “single system image”.

No product matched this selection criteria and the decision was made to create a solution in-house which would satisfy the requirements; therefore, MultiBatch was born.

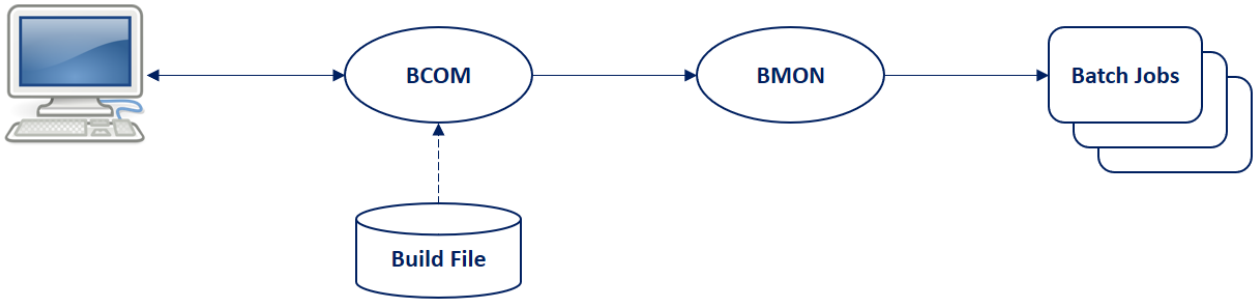
Since 1990, Insider Technologies Limited has owned and marketed this product. Although the product has received significant development investment to help extend and modernise the functionality on view, fundamentally, the core design of the product has not changed at all.

This original thinking has proved so flexible that the product has for 20+ years been deployed at many installations, including the UK’s leading and central banks, without the need for the core principles to be altered. This has led to a very stable and resilient software application.

The next section discusses the MultiBatch product design in more detail. If, however, you would just like to review the features of the product, then it is suggested that you move directly to the “MultiBatch Features” section.

The MultiBatch Benefits section of this White Paper describes how our customer base has utilised MultiBatch to facilitate significant improvements to their Disaster Recovery and System Coldload procedures.

MultiBatch Scheduler Process



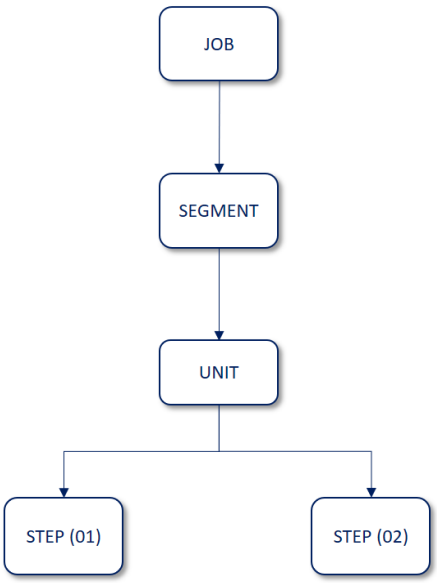
In this section we will discuss the design of the most crucial software module in the MultiBatch package, the **B**atch **MON**itor or BMON.

BMON is responsible for starting individual batch jobs in the correct order, monitoring that they complete successfully and informing you when they do not.

There can be any number of BMON processes executing on your system and each BMON will hold:

- A table of the jobs that it needs to execute
- The order that they should run in
- The names of other jobs in the schedule that they are dependent on

Let us now look at the way this scheduling and sequencing information is held in the BMON internal table. To do this we need to introduce some new concepts: Job, Segment, Unit and Step.



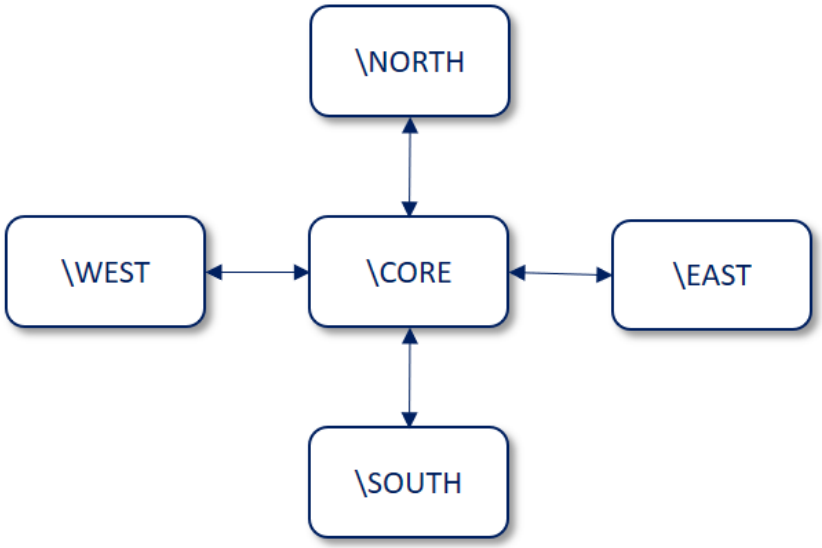
Jobs and Segments are used to define the batch flow sequence, dependency and parallelism.

Units and Steps are used to define a single application batch execution and all of the information that it would need to execute, such as start-up parameters, defines, file assignments.

To help explain the thinking behind the concept of Jobs and Segments, the original requirements of the first installation to implement the MultiBatch product need to be revisited.

The organisation in question had developed an application to provide telephone-banking facilities for their customers. The organisation had multiple regional offices, each with a local dedicated Tandem system that could be accessed by local clerks administering the accounts. To simplify things, let us call these regional Tandem systems, \NORTH, \SOUTH, \EAST and \WEST.

The datacentre also hosted a central Tandem system linking the regional systems together via Expand and it also provided a connection to the mainframe from the Tandem network. Let us call this central Tandem system, \CORE.



Operations staff managed all of the Tandem network from the \CORE system.

The main ledgering system was sited centrally, on a non-Tandem mainframe.

The banking application used distributed database techniques. A single file containing say “Customer Details”, would have four partitions, one resident on each of \NORTH, \SOUTH, \EAST and \WEST.

In the evening, the changes made to the distributed Tandem database during the day would be collected from the four remote systems to the central \CORE system and then forwarded to the non-Tandem mainframe as a single file; the changes would be applied to the database on that non-Tandem mainframe overnight.

The datacentre requirement for the batch scheduler was that all this activity could be configured, activated and monitored from a single point. Operations staff should not need to connect to the four systems to run the individual batch jobs, nor should they need to monitor four sets of logs for success/failure before starting the next job.

If we produce a simplified workflow for this proposed schedule it would look like this:

1. Build today’s parameter database on \CORE. This database would contain say, today’s date, address, customer name amongst other values.

2. Shutdown the on-line application on \NORTH, \SOUTH, \EAST and \WEST. As this was a PATHWAY system, freezing and stopping the PATHWAY servers was usually sufficient.

3. Run an application program to collect today's database changes (e.g. change of address) on \NORTH, \SOUTH, \EAST and \WEST and store the changes on \CORE.

4. Connect to the non-Tandem mainframe from \CORE and submit the file containing all of today's changes. To implement this schedule, a BMON was built to execute on \CORE. This meant that operations staff only needed to access this system in order to run the batch for the whole of the regional network.

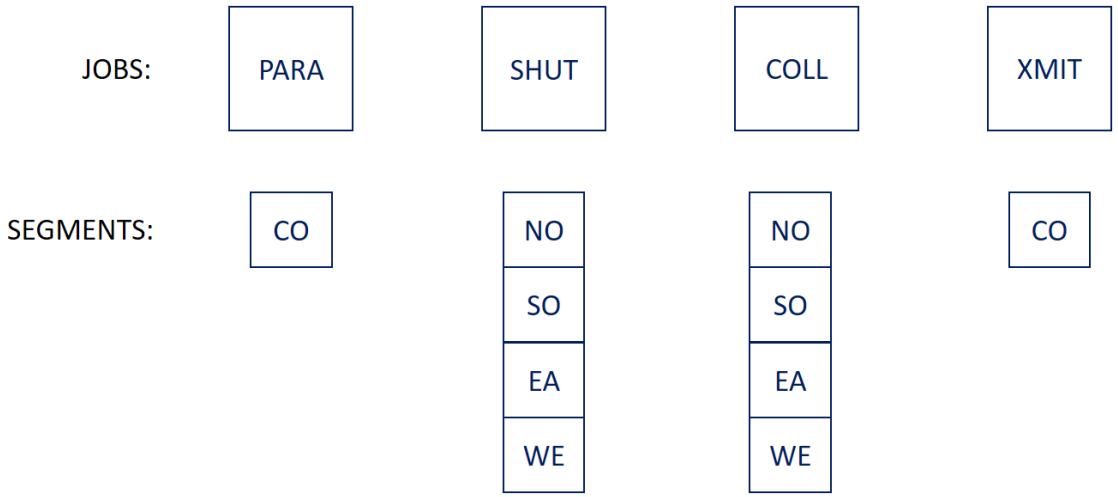
A MultiBatch JOB was created to map to each of the discrete functions described earlier. (A job name within MultiBatch is 4 characters long).

Job Number	Job Name	Previous Dependency	Description
1	PARA	None	Build today's parameter database
2	SHUT	PARA	Closedown a PATHWAY system
3	COLL	SHUT	Extract today's database changes and send to \CORE
4	XMIT	COLL	Send the changes from \CORE to the mainframe

The SEGMENT level of the hierarchy was used to signify which regional system that the program was executing on. (A segment name within MultiBatch is 2 characters long).

JOB	SEG	System	Description
PARA	CO	\CORE	Build today's parameter database
SHUT	NO	\NORTH	Closedown a PATHWAY system
	SO	\SOUTH	Closedown a PATHWAY system
	WE	\WEST	Closedown a PATHWAY system
	EA	\EAST	Closedown a PATHWAY system
COLL	NO	\NORTH	Extract today's database changes and send to \CORE
	SO	\SOUTH	Extract today's database changes and send to \CORE
	WE	\WEST	Extract today's database changes and send to \CORE
	EA	\EAST	Extract today's database changes and send to \CORE
XMIT	CO	\CORE	Send the changesfrom \CORE to the mainframe

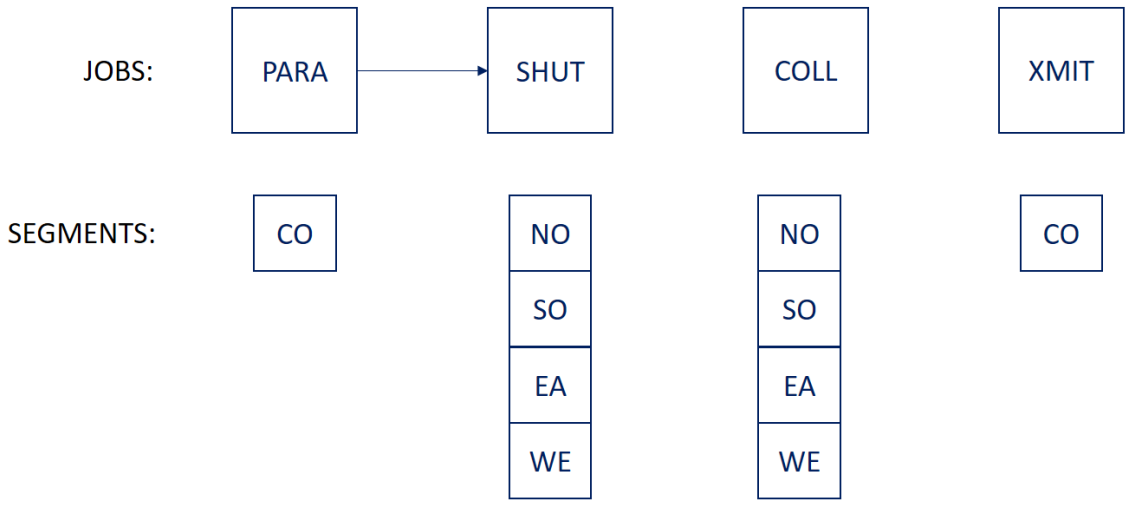
Once this structure is in place, then the flow can be built by choosing the right type of dependency.



The first job PARA will have no previous dependencies so this needs to be started manually, or through the automated scheduling techniques in the MultiBatch DS-EventTimer module.

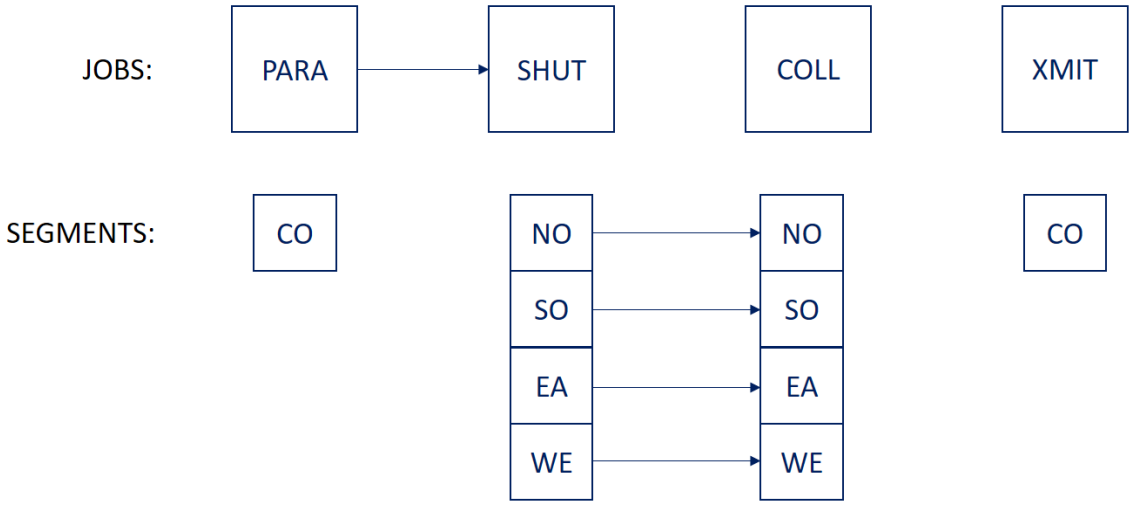
Once the PARA job is complete, then the objective is to close down the PATHWAY systems on the four Tandem nodes, \NORTH, \SOUTH, \EAST and \WEST. This activity can be done safely in parallel and it is achieved by starting the SHUT job rather than the individual segments within the job. All four segments then run simultaneously - one per Tandem system.

To allow this to occur, make job SHUT dependent on job PARA.

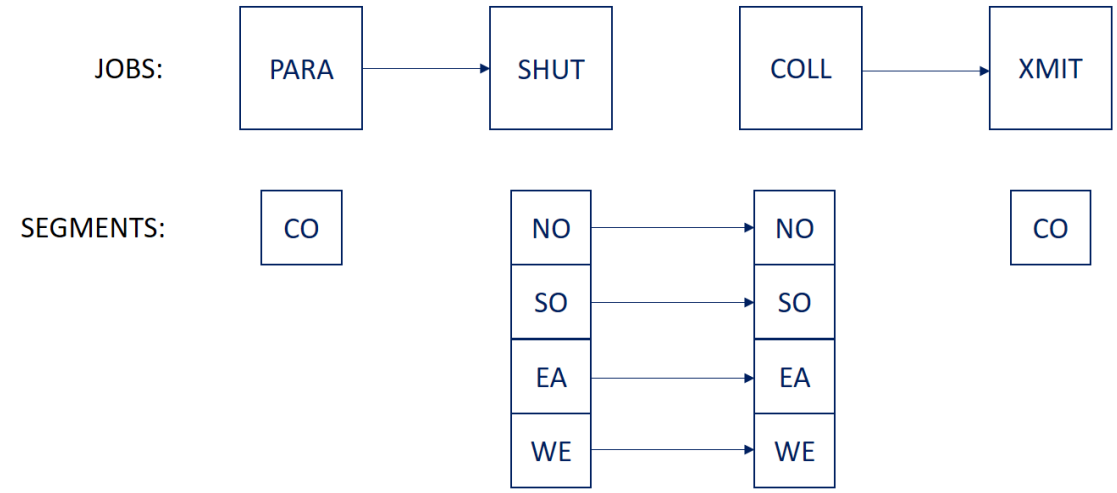


It would be possible to link the SHUT and COLL jobs together, but this makes an activity on \NORTH dependent on \SOUTH and vice-versa. The only requirement for the COLL collection program on \NORTH is that the local PATHWAY system has closed.

In this instance, it is advisable to make the dependency connection at segment level.



The last job, XMIT, can only be executed once all the collections have been completed. The best way to achieve this is to make the XMIT and COLL jobs dependent on each other.



So, looking back to the original operations requirements:

The operations staff have been able to configure this flow from just one Tandem system - \CORE.

They have been able to monitor this batch flow by looking in one place for log messages created by the BMON process on \CORE. The messages could be routed from the remote nodes to \CORE.$0 as EMS events.

Alternatively, the operator could connect to the BMON process on \CORE and issue STATUS JOB commands to discover the current status of batch jobs on systems other than \CORE. A “STATUS JOB COLL” command would give the status of all segments for the four programs on the four different systems.

In addition, the online configuration of all batch jobs could be altered from the one system. An “ALTER STEP COLL.EA.01 CPU 1,2” command from \CORE would alter the CPU specification for the \EAST collection mechanism. The job could then be restarted from \CORE.

The structure also provides operational control of the flow. If it is not possible to process batch on \WEST because of pre-arranged late working for example, then an operator would simply issue a “HOLD SEGMENT SHUT.WE” command. The remainder of the schedule for other regions would progress as normal.

In this example, segments have been mapped on to the same type of task running concurrently on four separate nodes. In another example, it could just as easily have been used to provide parallelism on the same node for different types of task, e.g. a FUP and a SORT. The MultiBatch product allows that flexibility.

Before starting to build a MultiBatch schedule it is advisable to draw out a schematic of the proposed batch flow. This diagram should identify all the proposed executables, their sequence and their dependency on other executables, the days, dates and/or times of execution, and most significantly the potential for parallel processing. The diagram can then be transferred easily into the MultiBatch configuration database.

Units and Steps

Now that we have addressed the sequencing, dependency and parallelism issues with Jobs and Segments, we can start to look at Units and Steps.

Units and Steps are concepts that deal with what the batch program is actually doing, rather than the order in which it runs.

Units and Steps include configuring the name of the object code, standard run options such as CPU, NAME, IN, OUT, PRI, TERM, HIGHPIN and file assignments, parameters and defines.

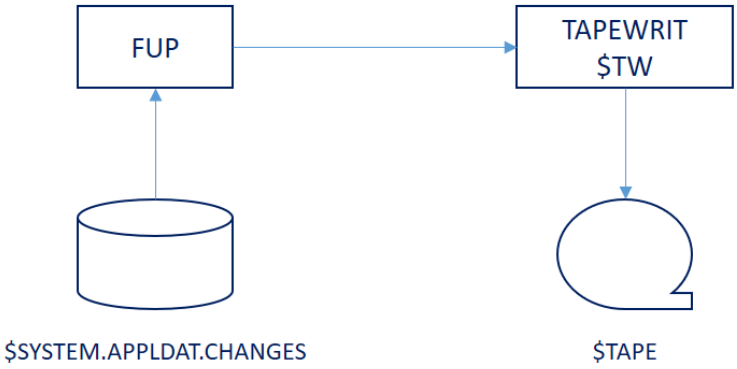
Users sometimes get confused when trying to understand the difference between a Step and a Unit, but in essence it is quite straightforward. In virtually all cases, the unit to step relationship is one to one. But there is a special case called the 2-step unit that we will discuss here.

Referring back to our example in the “Jobs and Segments” section, part of the requirement of that particular installation was to transfer information from the Tandem \CORE system to a non-Tandem mainframe. Initially this information exchange took place on tape and to achieve this the following batch sequence was required:

```
TAPEWRIT / NAME $TW, OUT $TAPE, CPU 0, NOWAIT /

FUP / CPU 1 / COPY $SYSTEM.APPLDAT.CHANGES, $TW
```

Although these are two programs, it is really one MultiBatch Job, as the two elements do not make any sense on their own. In addition, the FUP process needs to be started second as its output is $TW, which is the first program.



This set up is known as a 2-step Unit.

By configuring TAPEWRIT as Step XMIT.CO.01.1 and the FUP process as Step XMIT.CO.01.2, then the MultiBatch scheduler will start the processes in the correct sequence.

The MultiBatch product is also equipped with a command utility – BCOM - which operators can use to communicate with any active MultiBatch BMON scheduler process.

The BCOM commands that need to be applied to both programs at the same time, are Unit commands.

- START UNIT TAPW.AA.01
- RESTART UNIT TAPW.AA.01
- ABORT UNIT TAPW.AA.01
- HOLD UNIT TAPW.AA.01
- RELEASE UNIT TAPW.AA.01
- DELETE UNIT TAPW.AA.01

The BCOM commands that need to be applied to only one program at a time, are Step commands.

- ADD STEP TAPW.AA.01.1
- ADD STEP TAPW.AA.01.2
- ALTER STEP TAPW.AA.01.1
- ALTER STEP TAPW.AA.01.2

In the vast majority of cases, users configure 1-step Units. In these instances the above commands are still valid and although the user does not have to supply a step NAME (e.g. ALTER STEP TAPW.AA.01 CPU 1,2), implicitly it is “.1”.

So from this we can see that Units are another grouping mechanism and Steps are the configuration layer for the executing program.

To complete our case study we can configure the following options.

JOB	SEG	Unit/Step	System	
PARA	CO	01	\CORE	An application program to build the parameter database
SHUT	NO	01	\NORTH	A PATHCOM command
	SO	01	\SOUTH	A PATHCOM command
	WE	01	\WEST	A PATHCOM command
	EA	01	\EAST	A PATHCOM command
COLL	NO	01	\NORTH	An application program to extract database changes
	SO	01	\SOUTH	An application program to extract database changes
	WE	01	\WEST	An application program to extract database changes
	EA	01	\EAST	An application program to extract database changes
XMIT	CO	01.1	\CORE	The Tape production program
		01.2	\CORE	A FUP program

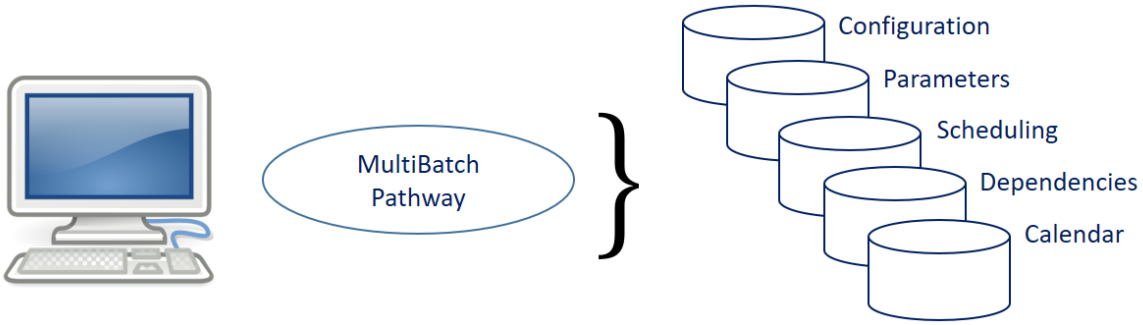
By including the Tandem system name in the object code specification, the BMON process will know that a remote process needs to be created.

MultiBatch Features

Now that we have described the concepts behind how a MultiBatch schedule is structured, we can take a step back and investigate the other components that allow us to create a fully primed BMON schedule process.

Configuration

A PATHWAY environment is provided with the product and this allows a user to build the MultiBatch configuration database. The database management system is entirely ENSCRIBE.



This set of PATHWAY servers can be accessed via either a Windows based GUI application or a set of “green screen” PATHWAY requesters.

Access to the PATHWAY system is only permissible if a valid Safeguard Tandem User id/Alias and appropriate password are provided. In addition, this user needs to have been registered within the MultiBatch security database by an administrator. Access to individual functions such as configuration, scheduling, and monitoring can be allocated at the user level. Security is discussed in more detail in a later section.

The configuration screens allow a user to build:

- The batch jobs that will execute.
- The sequence that they will execute in, together with dependencies.
- The attributes of the executing program, e.g. node, object code, CPU number, process name, file assignments, parameters and defines.
- The User Id that the batch job will runs as.
- The days of the week that the job, segment, or unit should be Included or Excluded from the schedule, e.g. include DAILY but exclude on THURSDAY.
- The calendar dates that a step should be included or excluded from the schedule, e.g. Include FIRST-DAY, where this is 02/01/2017 & 01/02/2017 & 01/03/2017.
- The maximum length of time that a step should execute and whether it should be aborted or reported when this time period is exceeded.
- The time that the step should have started. A log message will be generated if the start time passes before the job is invoked. This facility can be used to report schedules that are falling behind planned cut-off times or because of forgetfulness on behalf of an operator.

To ease the amount of configuration maintenance, there are a number of facilities available:

- Classes
- Conditional Parameters
- Migration

Classes

Common parameters, file assignments and defines can be placed within a class and the class can then be attached to more than one batch step. If a change is required then just a single update to the Class value can be made which will apply to multiple batch steps.

Conditional Parameters

For configuration attributes that might alter between environments then the use of conditional parameters can also guard against the requirement for multiple and manual updates.

For example, if the application disc on Tandem node \TEST is $TESTAP and on \LIVE it is called $LIVEAP, you could create the following file assignment in the MultiBatch configuration database.

ASSIGN REPORT-FILE <APPL-DISC>.APPLDATA.REPORT

Where <APPL-DISC> contains the requisite value.

Conditional parameters are held between “<” and “>” characters by default, although this can be changed.

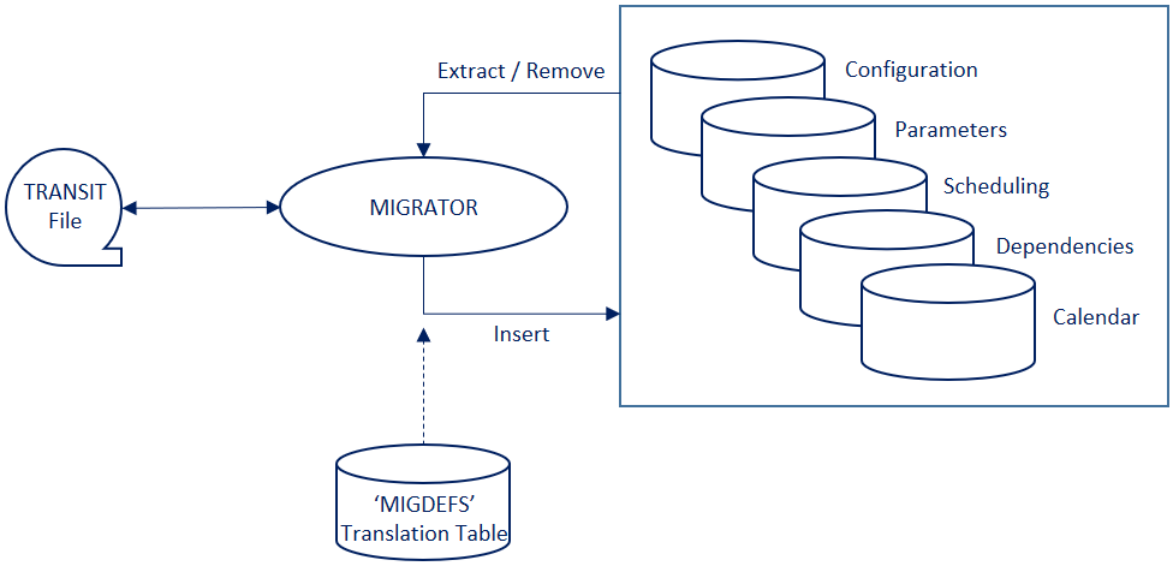
The conditional parameter database on \TEST should contain an APPL-DISC=”$TESTAP” entry and on \LIVE it should contain an entry for an APPL-DISC=”$LIVEAP”. The appropriate substitution will take place when the schedule is loaded.

Conditional parameters can be used for almost all configuration attributes.

More than one conditional parameter can be used in a configuration attribute, e.g. object code = <NODE>.<APPL-DISC>.<APPL-SVOL>.REPTOBJ.

Conditional parameters can also be used in assign, parameter and define classes, reducing the maintenance still further.

Migrator



After you have built and tested a batch schedule and you need to transfer it to another system, then you do not want to have to manually recreate either the entire schedule or the last set of updates.

The Migrator allows a single schedule, or part of a single schedule, to be “extracted” from a MultiBatch configuration database and written to a “transit” file. This solitary file can be released to a target environment where the “insert” functionality of the Migrator product can be used to update a target MultiBatch configuration database on the same or remote node.

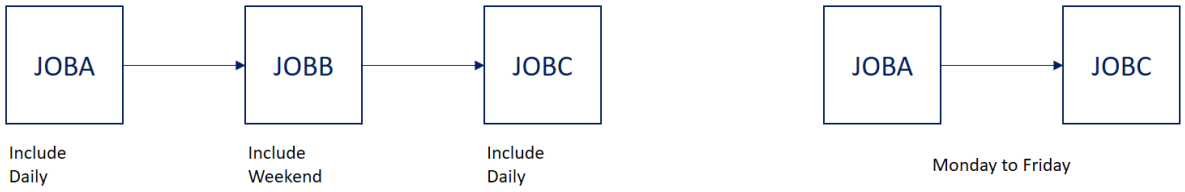
If the transit file contains references to the original node, volume or subvolume, then the “insert” processing on a remote node will alter these values based on a simple translation table.

A combination of classes, conditional parameters and the Migrator greatly reduces the amount of configuration required.

Prepare Processing

In putting together a MultiBatch configuration, we have described the sequence of a batch flow and how it would execute if every step executed every day.

An additional part of the configuration involves nominating the days and dates that steps will run. In this way, elements of the configured schedule are dropped on a given day and the remainder of the schedule is then linked together to execute in the correct sequence.



This processing is performed by the PREPARE program.

In this section, we will look at the PREPARE process in more detail but first we will examine the scheduling options.

By default, steps will run daily but this state can be overridden by using one or all of three types of parameter.

- **Absolute** - This is a hard coded parameter that includes DAILY, WEEKDAY, WEEKEND, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY and SUNDAY.
- **Calendar** – Associate one or more dates with a given name. An example could be BANKHOL = 01/01/2017, 25/12/2017, 26/12/2017.
- **Frequency** – This is not date driven. It is a Yes/No switch that can be used to force an ad-hoc job to execute.

Absolute and Calendar parameters support the use of “include” and “exclude”. For example, “Include DAILY” together with “Exclude THURSDAY” is a valid combination; likewise “Include MONDAY” with “Exclude BANKHOL” is also valid.

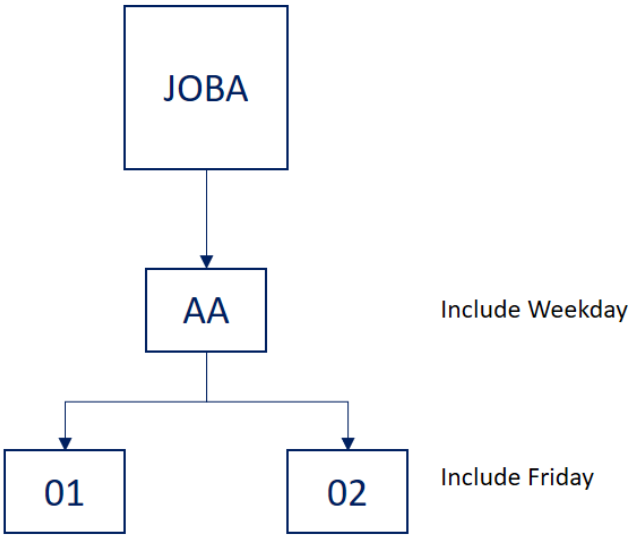
Some other features to note:

If no values are set, then the nominated Job, Segment and Step will execute every day, as the DAILY parameter is assumed as true.

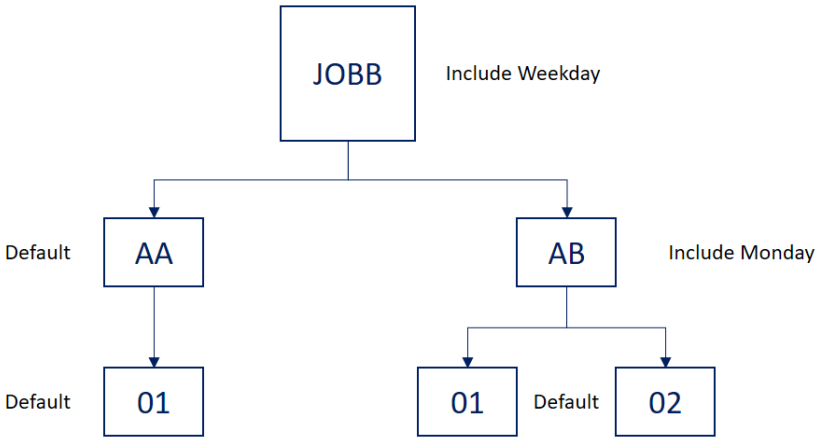
If a step has no schedule values, then it inherits its segment value.

The same can be said of a segments relationship with its job. Conversely if a step has a schedule value it overrides the segment value and likewise with the segment and job relationship.

See example schedules on next page.



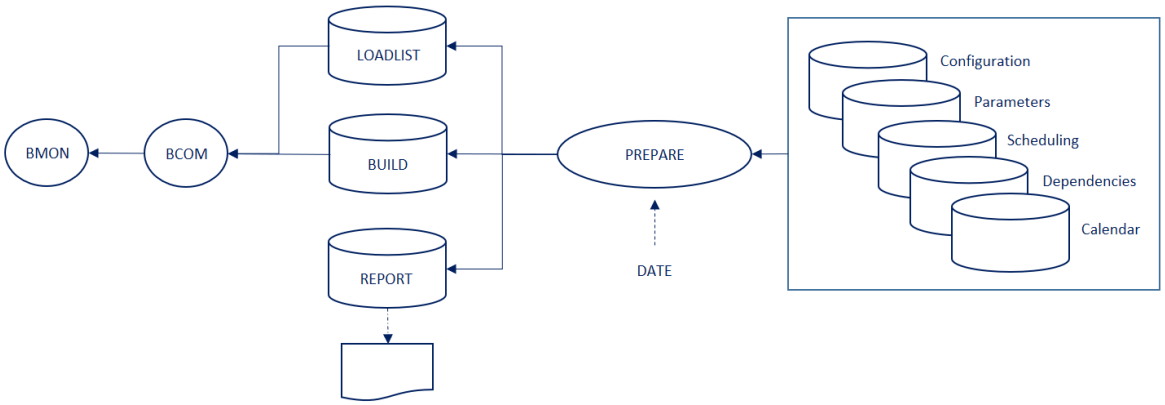
- JOBA.AA.01 – runs Weekdays
- JOBA.AA.02 – runs Fridays only



- JOBB.AA.01 – runs Weekdays
- JOBB.AB.01 – runs Mondays only
- JOBB.AB.02 – runs Mondays only

The concept behind the MultiBatch scheduling facilities is that a new job list for the day needs to be submitted every 24 hours.

The creation of that job list is the responsibility of the PREPARE process. PREPARE processing can be done in advance of the existing schedule completing and can be requested for days other than the current or next processing day and without affecting the existing schedule. This provides a very powerful scheduling tool, whereby operations staff can create a report showing the run order and perhaps more importantly, list the non-running jobs for an advance date such as a public holiday.



The PREPARE needs several parameters to function:

- The processing date.
- The name of the batch schedule to be “Prepared”.
- For this date, shall we calculate the Absolute parameters?
- For this date, shall we interrogate the Calendar files?
- The name of any Frequency parameters that are to be set to “true”.

For example, if we supplied a processing date of 02/01/2017 for the $BMON batch environment and selected Absolute=Y, Calendar=Y and Frequency=OLDQUART and NEWQUART, the following criteria will be set for batch job selection.

- DAILY – 02/01/2017
- WEEKDAY – 02/01/2017
- MONDAY – 02/01/2017
- FIRSTMON – Calendar
- OLDQUART - Frequency / ad hoc
- NEWQUART – Frequency / ad hoc

All the Jobs, Segments and Steps in the $BMON schedule will be examined and their scheduling criteria will be matched against this list. The include/exclude criteria are also taken into account

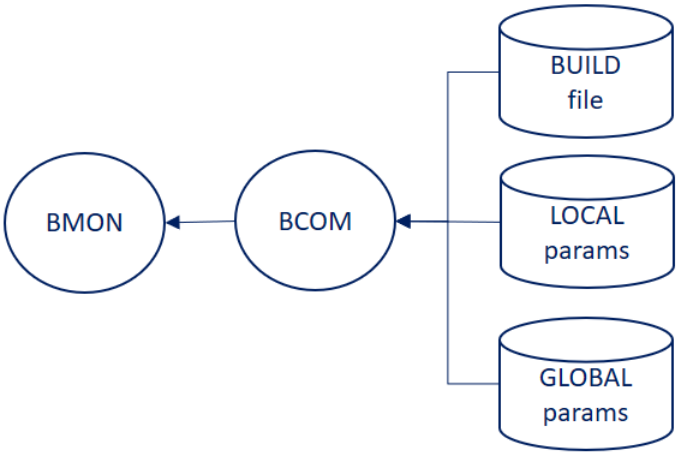
A report is generated that includes the following information.

- The parameters used.
- Batch elements that are to run and the reason why.
- Batch elements that will not run and the reason why.
- The sequence of the running jobs.
- Any re-linking that has taken place. JOBA ---> JOBC, because JOBB is not scheduled to run on that Prepare date.
- Jobs that have no previous dependencies and so will need either a manual start or an automatic start using the MultiBatch scheduling facility, DS-EventTimer.

The PREPARE process can create the Loadlist and Build files

The “Loadlist” file is a work file that contains the names of all the MultiBatch elements and their run status, together with any new links that have been created. The file can be viewed through the PATHWAY or GUI environments and it is used as part of the report and build processes.

The Build file contains the names of all the MultiBatch elements together with all the run time configuration information such as a steps object code, run-time information, file assignments, parameters and defines. This information will still have the conditional parameters embedded and the BUILD process will replace these values when the BMON is created.

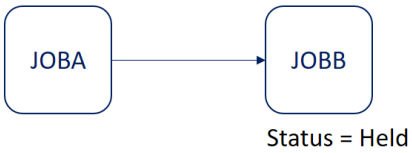


Batch Execution

Once the batch schedule has been provided with that processing day's list and order of processing, then the jobs can be loaded and then executed.

Jobs and Segments with previous dependencies configured will be started automatically once all the previous dependencies have successfully completed. There is one addition to this functionality. It is possible to start jobs through a combination of time and dependency. For example, start B once A completes or at 18:00 whichever is the later.

Where dependencies exist, they can be overridden by use of the HOLD and RELEASE commands within the command utility, BCOM



Jobs without previous dependencies can be started manually with BCOM.

BCOM $<name-of-the-scheduler>

» START JOB PARA

» EXIT

As we discussed in our earlier case study, if this job is structured with multiple segments and units then this single command could start many parallel batch processes distributed around your network.

If there is an issue with a network connection, the jobs could be started manually at segment level.

BCOM $<name-of-the-scheduler>

»START SEGMENT SHUT.NO

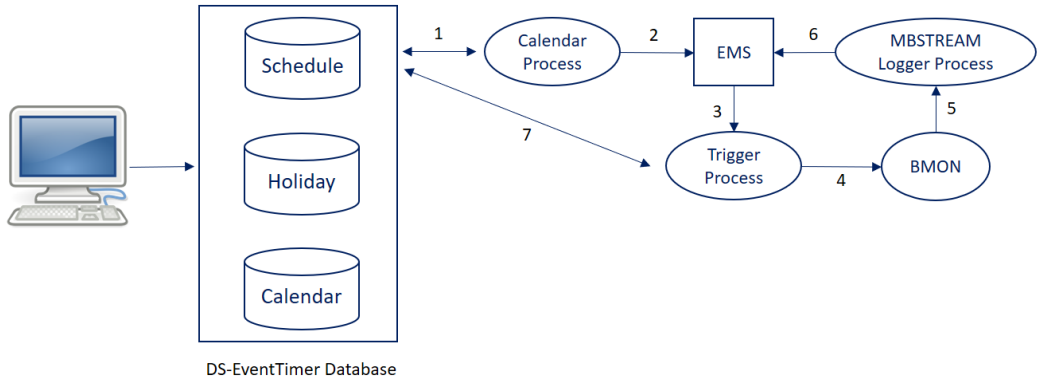
»START SEGMENT SHUT.SO

»START SEGMENT SHUT.WE

»EXIT

There are two methods are automating the execution of jobs: DS-EventTimer and LIBUTIL.

DS-EventTimer



Within DS-EventTimer, there are three types of scheduling available: AT, EVERY, CRONTAB

AT scheduling is used to start a job once per day, for example AT 21:00. You can select days and dates when the AT processing is performed.

EVERY scheduling is used to start a job at regular intervals, for example EVERY 15 MINUTES. You can select BETWEEN times such as BETWEEN 09:00 and 17:00. This type of schedule is normally used to execute homegrown system management macros such as a “file percentage full” check routine.

CRONTAB is a powerful, multi date/time based scheduling component of the MultiBatch DS-EventTimer. A user can configure a job, segment, or unit to run at any time of the day, several times a day, for 24 x 7 days of the year / years, e.g. start job at 08:45, 09:30, 11:12, only on a Wednesday, Friday, if date is 30th. Any combination is possible. CRONTAB is equivalent to the Unix Cron scheduler.

The DS-EventTimer facility comes with a reporting mechanism that can be viewed through the User interface or in print. For example, you can retrieve jobs in “next run” order or a list of jobs that failed to start.

If you are planning a system outage for major maintenance or an upgrade, all of the next run times can be recalculated as a single exercise by using the RESCHED utility.

A library routine is provided with the product that allows an application process to alter the schedule database. If the logic or an event within the application suggests, for example, that an associated batch job can start at 14:00 rather than 15:00, then this time can be altered using this library utility.

LIBUTIL

In those instances where the application needs to control the start of the batch jobs, the LIBUTIL library can be bound into your code to provide START, ABORT, RESTART, HOLD and RELEASE capability.

One existing installation uses this facility to allow an Operator to start a batch job from a PATHWAY function key, once the Operator has verified some application details on the same PATHWAY screen.

MultiBatch Batch Monitoring

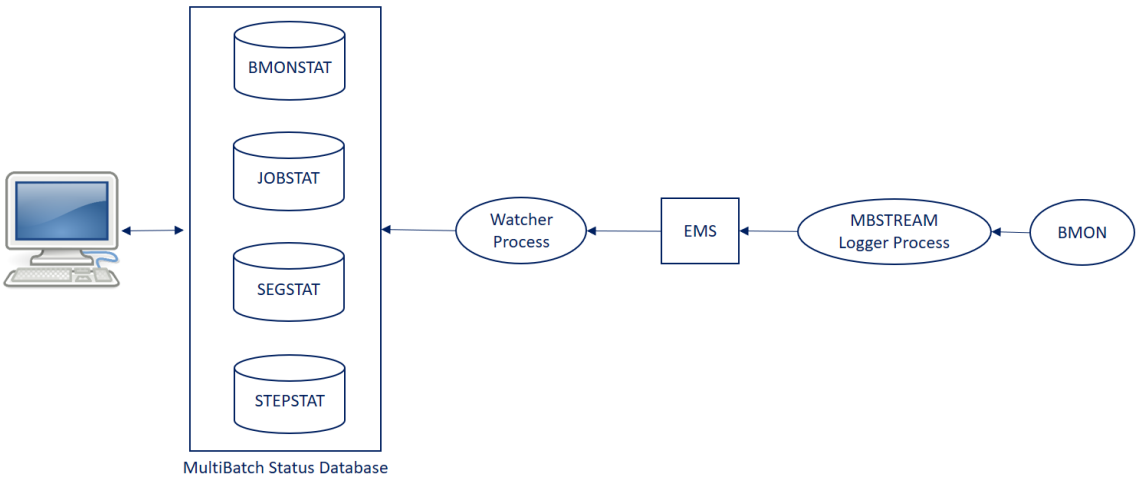
There are a number of monitoring facilities within the MultiBatch product.

Abend: Batch steps that abend or finish with a completion code other than 0, 1, or 8 are considered as incomplete. This fact is reported to the NonStop Event Management Service (EMS) and subsequent dependencies are not executed.

Check Start: When jobs do fail, then rather than wait for that issue to be fixed, a local Operational decision is often made to start the next job so that the schedule is not held up. In a complex schedule without up to date schemas available, errors can be made. The MultiBatch schedule will check, based on the configured dependencies, that jobs are not being run out of sequence or started for a second time and report this fact. An Operator can then choose to override this state.

Max Run Time: Each step can be allocated a “maximum” run time. If this run time is exceeded then that particular step can be aborted or a warning message written to the EMS log.

Monitored Start Time: The start time of a batch step can be monitored. If the nominated time passes without the step starting, then a warning message is written to EMS. This facility can be used to check that manual jobs have been started on time or that important cut-off times will be met. For example, to meet our 04:00 deadline we are normally at point X by 02:00 and this has not been reached yet.

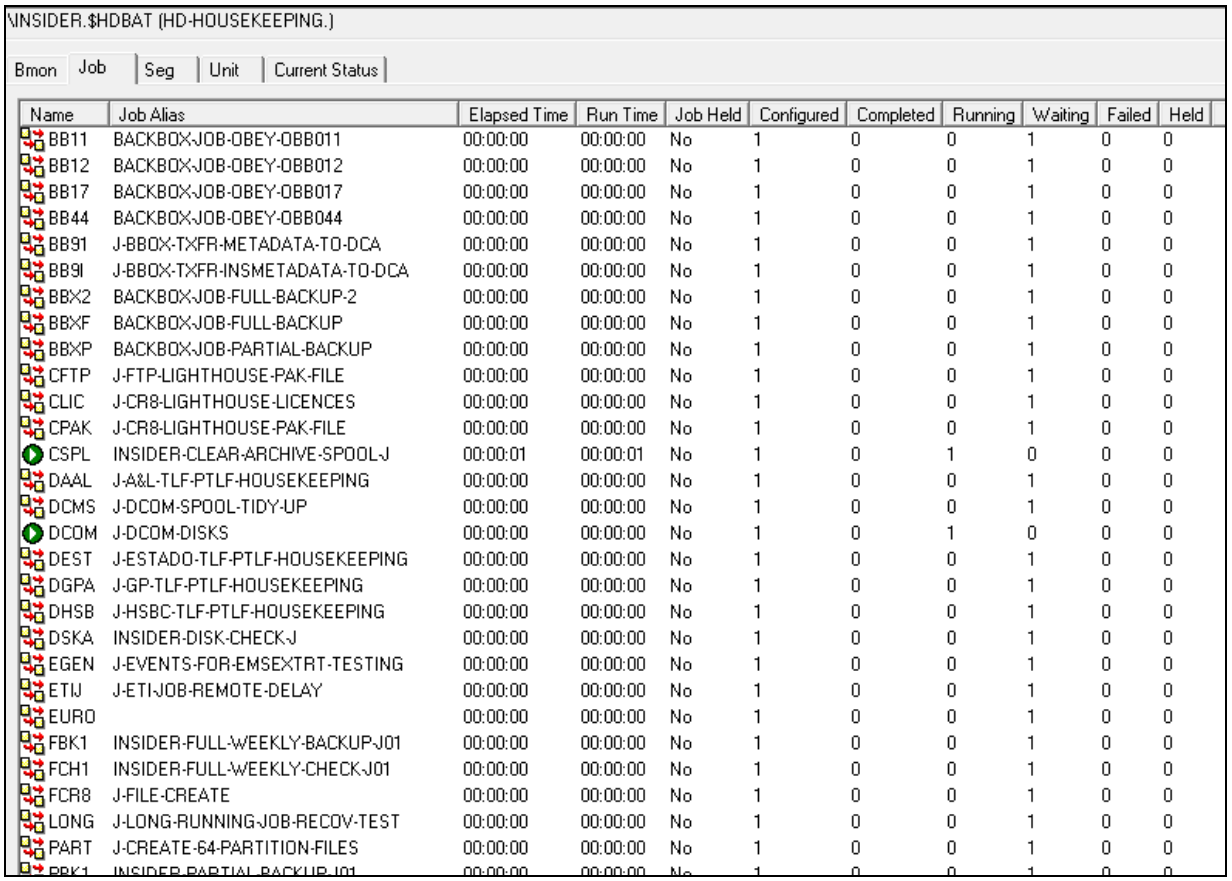


The MultiBatch scheduler is equipped with the ability to write to two nominated log files. For the MultiBatch modules DS-EventTimer and the Watcher process to function correctly, one of these log files should be an MBSTREAM process, a part of the MultiBatch product.

Real-Time Status Monitoring Facilities

The Watcher process is listening to EMS for MultiBatch events and subsequently updating the Status database, the results of which are reflected in Pathway and/or GUI status screens.

The number of MultiBatch Steps within each discrete batch environment are organised into Configured, Completed, Running, Waiting, Failed and Held columns and these statistics are updated in real-time within the MultiBatch Status screens as the events arrive.



\INSIDER.$HDBAT (HD-HOUSEKEEPING.)

Bmon | Job | Seg | Unit | Current Status

Name	Job Alias	Elapsed Time	Run Time	Job Held	Configured	Completed	Running	Waiting	Failed	Held
BB11	BACKBOX-JOB-OBEY-OBB011	00:00:00	00:00:00	No	1	0	0	1	0	0
BB12	BACKBOX-JOB-OBEY-OBB012	00:00:00	00:00:00	No	1	0	0	1	0	0
BB17	BACKBOX-JOB-OBEY-OBB017	00:00:00	00:00:00	No	1	0	0	1	0	0
BB44	BACKBOX-JOB-OBEY-OBB044	00:00:00	00:00:00	No	1	0	0	1	0	0
BB91	J-BBOX-TXFR-METADATA-TO-DCA	00:00:00	00:00:00	No	1	0	0	1	0	0
BB9I	J-BBOX-TXFR-INSMETADATA-TO-DCA	00:00:00	00:00:00	No	1	0	0	1	0	0
BBX2	BACKBOX-JOB-FULL-BACKUP-2	00:00:00	00:00:00	No	1	0	0	1	0	0
BBXF	BACKBOX-JOB-FULL-BACKUP	00:00:00	00:00:00	No	1	0	0	1	0	0
BBXP	BACKBOX-JOB-PARTIAL-BACKUP	00:00:00	00:00:00	No	1	0	0	1	0	0
CFTP	J-FTP-LIGHTHOUSE-PAK-FILE	00:00:00	00:00:00	No	1	0	0	1	0	0
CLIC	J-CR8-LIGHTHOUSE-LICENCES	00:00:00	00:00:00	No	1	0	0	1	0	0
CPAK	J-CR8-LIGHTHOUSE-PAK-FILE	00:00:00	00:00:00	No	1	0	0	1	0	0
CSPL	INSIDER-CLEAR-ARCHIVE-SPOOL-J	00:00:01	00:00:01	No	1	0	1	0	0	0
DAAL	J-A&L-TLF-PTLF-HOUSEKEEPING	00:00:00	00:00:00	No	1	0	0	1	0	0
DCMS	J-DCOM-SPOOL-TIDY-UP	00:00:00	00:00:00	No	1	0	0	1	0	0
DCOM	J-DCOM-DISKS	00:00:00	00:00:00	No	1	0	1	0	0	0
DEST	J-ESTADO-TLF-PTLF-HOUSEKEEPING	00:00:00	00:00:00	No	1	0	0	1	0	0
DGPA	J-GP-TLF-PTLF-HOUSEKEEPING	00:00:00	00:00:00	No	1	0	0	1	0	0
DHSB	J-HSBC-TLF-PTLF-HOUSEKEEPING	00:00:00	00:00:00	No	1	0	0	1	0	0
DSKA	INSIDER-DISK-CHECK-J	00:00:00	00:00:00	No	1	0	0	1	0	0
EGEN	J-EVENTS-FOR-EMSEXTRT-TESTING	00:00:00	00:00:00	No	1	0	0	1	0	0
ETIJ	J-ETI-JOB-REMOTE-DELAY	00:00:00	00:00:00	No	1	0	0	1	0	0
EURO		00:00:00	00:00:00	No	1	0	0	1	0	0
FBK1	INSIDER-FULL-WEEKLY-BACKUP-J01	00:00:00	00:00:00	No	1	0	0	1	0	0
FCH1	INSIDER-FULL-WEEKLY-CHECK-J01	00:00:00	00:00:00	No	1	0	0	1	0	0
FCR8	J-FILE-CREATE	00:00:00	00:00:00	No	1	0	0	1	0	0
LONG	J-LONG-RUNNING-JOB-RECOV-TEST	00:00:00	00:00:00	No	1	0	0	1	0	0
PART	J-CREATE-64-PARTITION-FILES	00:00:00	00:00:00	No	1	0	0	1	0	0
PBK1	INSIDER-PARTIAL-BACKUP-J01	00:00:00	00:00:00	No	1	0	0	1	0	0

The Watcher process also updates an error log which can be viewed by an Operator. This provides a list of failed jobs and the reasons why. This database is linked to an Operators diary that can be used as part of the hand over process between different support personnel. These files can be viewed through a set of status screens.

Batch Schedule Prediction

When MultiBatch jobs run and complete, records are stored in the status database. A component of this status database provides a mechanism where a user can view the average, maximum and minimum elapsed run times of selected jobs.

Utilising these metrics, MultiBatch can predict to the user how long a schedule will take. For example, JOBA has been running for 30 minutes longer than normal and based on this, later dependencies in the schedule may not complete on time, e.g. account records may not get updated before end of day processing starts.

Security

In this section we will look more closely at the security and audit functionality that is available within MultiBatch. There are security systems for both the configuration database and the executing batch schedule. Both are based on the NonStop Safeguard security model.

To access the configuration database through the user interface, a Guardian User name must be registered within the MultiBatch database in the SECPROFL file. The system is flexible enough to allow or deny access to each individual screen at the individual User level. In some instances even the functions on a screen can be divided, an example being the amend and delete keys on the Operators diary screen.

Access to screens can be grouped together, for example:

CONFIG = maintain BMON + maintain job + maintain segment + maintain step.

CONFIG is known as a Security Class and the contents of a class are held in the SECCLASS Enscribe file. Users, or the corresponding Safeguard alias, are attached to Classes and this is known as the Users profile. This information is held in the SECPROFL Enscribe file.

The last layer of database security is known as the “Owner”. Users can be made co-owners of a particular MultiBatch schedule. This allows the permissions defined in the CONFIG class, for example, to be applied to one batch schedule but not to another.

Once the MultiBatch BMON process is running, another security layer is in place to restrict alteration of the execution or configuration of the production batch environment. The BMON process has a nominated owner, which is not necessarily the User that the process is executing as. In this example, we will say that it is SUPER.OPERATE, or 255,200. The commands that can be executed through the command line utility, BCOM, are split into 4 sections:

- Information commands such as INFO and STATUS. There is no restriction on the access to these commands.
- Configuration commands such as ADD, DELETE and ALTER.
- Scheduling commands such as START, RESTART, ABORT, HOLD and RELEASE.
- Privileged commands such as changing a steps state from failed to complete or shutting the schedule down while jobs are still executing.

Both the configuration and scheduling groups of commands are allied to a file system type of setting. So if the specified owner = SUPER.OPERATE and security = G,O we are saying anybody in the SUPER group (G) can configure and only SUPER.OPERATE (O) can issue schedule commands.

The privileged category belongs to SUPER.SUPER and optionally another User nominated by the installation.

Both the database and BMON updates have an audit capability. For any database changes, a summary of the update is captured and can be viewed through the User interface. For example:

Time = HH:MM:SS,DD/MMM/YYYY, User=SUPER.OPERATE, Terminal=$T100, Screen= Maintain Step, Action=Amend. Optionally the changed record can also be captured and a print facility, AUDREPT, can be used to print out the alterations.

When any changes are made to a 'live' BMON configuration using the ALTER STEP command, an EMS event is generated with the salient details. An example of a log message is shown below:

```
PCH2.AA.01.1, this step's CPU altered from 01:00 to 00:01, \INSIDER.$HDBAT
DD/MM/YY, HH:MM; Requesting CAID = 255,255
```

Recovering a BMON

In addition to fully tokenised MultiBatch EMS events and subsequent updates to the status tables by Watcher, the status of a batch schedule including waiting, running, failed jobs is dynamically recorded in a TMF audited BMON Recovery file.

If a BMON process is shutdown whilst the schedule is still running, or perhaps due to a system crash, then recovery techniques are available to recover the BMON back to the point of failure.

Once recovered, this enables a user to determine the state of the batch schedule, e.g. jobs JOBA to JOBG are complete but JOBH is marked as incomplete as it was still running when the BMON process failed. A user can then determine the affect this has on their application.

Additionally, as the BMON Recovery file is TMF audited, data replication products can be used to maintain a copy of this file on a standby system. As the BMON Recovery file is regularly updated during the running of the batch schedule, the copy will be simultaneously updated on the standby system.

Therefore, if a system crash occurs, then following a site-swap, the BMON on the new live system can be rebuilt and users can check the status of the batch schedule up to the point of system failure.

MultiBatch Benefits

System Coldload

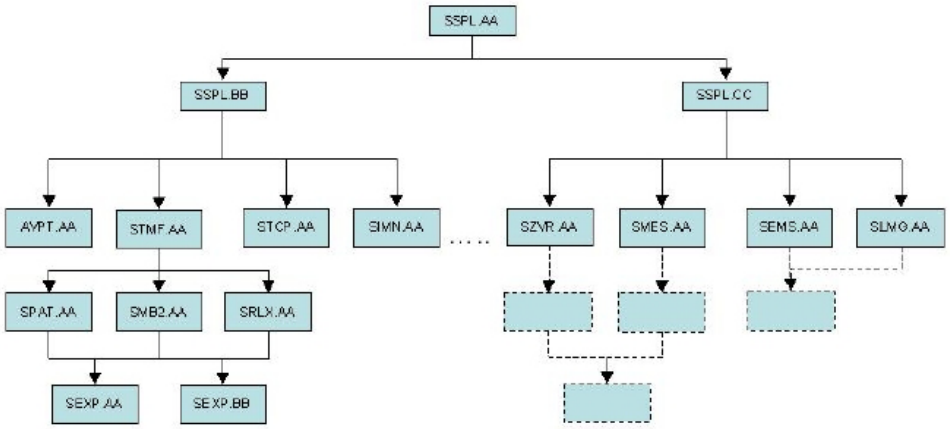
Traditionally, the coldload of a NonStop system invokes TACL based 'coldload' files. Contents of these files include commands to start TMF, DISKS, TCP/IP, LAN, EXPAND, RDF, Spoolers, etc. with each command line being invoked sequentially.

Although each command line in the startup files can be invoked sequentially and in turn load up the NonStop system, the parallelism of MultiBatch can speed up the coldload operation. This is especially beneficial when, following a site-swap, the 'new' Production node needs to be started up as soon as possible.

The MultiBatch units can execute the required site-swap commands using different user ids, e.g. a unit for STMF.AA (Start TMF) is run as 255,255, whereas the units for starting the various Pathways may be run under different 'Manager' ids.

As can be seen, the parallel approach of MultiBatch will facilitate the rapid coldload of a NonStop system, thereby reducing downtime and improving SLAs for customers.

SEGMENT ID	FUNCTION	EXECUTING USER ID
SSPL.AA	Start spooler $SPLS	255,255 (SUPER.SUPER)
SSPL.BB	Start spooler $BPLM	255,222 (SUPER.OPERATOR)
SSPL.CC	Start spooler $CPLM	255,222 (SUPER.OPERATOR)
STMF.AA	Start TMF	255,255
SVPT.AA	Start Viewpoint	100,255 (VWPT.MANAGER)
STCP.AA	Start TCP/IP	255,255
SIMN.AA	Start IMON	255,255
SZVR.AA	Start ZSERVER	255,255
SMES.AA	Start Measure	255,255
SEMS.AA	Configure EMSCTRL	255,255
SLMG.AA	Start LAN Managers	255,255
SPAT.AA	Start Application Pathway	125,255 (APPL.MANAGER)
SMB2.AA	Start MultiBatch Pathway	250,250 (MBAT.OWNER)
SRFX.AA	Start Reflex 80:20 Pathway	080,020 (REFLEX.OWNER)
SEXP.AA	Start Expand Line $EXPA	255,255
SEXP.BB	Start Expand Line $EXPB	255,255
Other jobs.segments	----	----

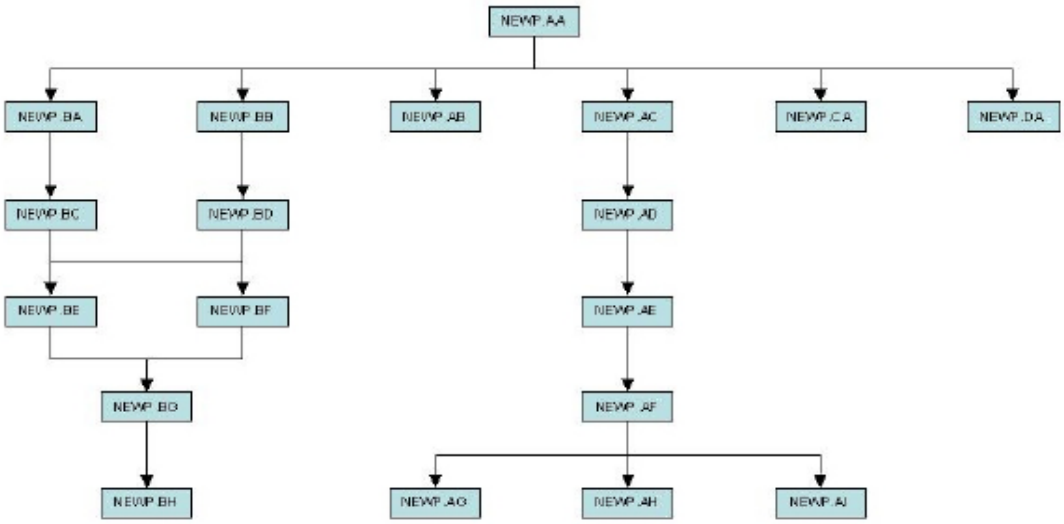


Disaster Reovery

A Disaster Recovery BMON can be created on a standby system in the event a site-swap scenario needs to be actioned. This site-swap BMON is pre-configured to reload the system, applications, networks, processes, Pathways, TMF and all essential subsystems in readiness for any site-swap.

MultiBatch allows jobs to be placed on hold, or in a ‘runoff’ situation, whereby any units that are configured with a ‘runoff’ value will be skipped. So, if a site-swap is required, operations can amend the units on the standby system from ‘runoff’ to ‘runon’ and the schedule can be started.

A flowchart example of how MultiBatch can be used in a site-swap situation is provided below, where the batch schedule executes on the new Production node. As with a Coldload MultiBatch schedule, the units can run under different user ids.



SEGMENT ID	FUNCTION
NEWP.AA	Set units in BMON $NEWP 'ready to run'
NEWP.AB	Down Expand Lines
NEWP.AC	Check RDF Takeover has completed successfully
NEWP.AD	Stop, Start TMF and set Next Audittrail MAT
NEWP.AE	Set Audit flags on database. This segment may not be required, depending on the version of RDF used (or other remote database duplication software used, e.g. 'Goldengate')
NEWP.AF	Check Audit Flags
NEWP.AG, AH, AI...	Run Online Dumps for the various databases
NEWP.BA	Start EMS Alternate Collectors and Distributors
NEWP.BB	Start 'Operations' Viewpoint
NEWP.BC	Start Application Pathway
NEWP.BD	Start Application Pathway
NEWP.BE	Start Application processes
NEWP.BF	Start Application Pathway Servers
NEWP.BG	Start Networks (TCP/IP, X.25, etc.)
NEWP.BH	Start External Networks
NEWP.CA	Down Testing Environments
NEWP.DA	Down Training Environments

Conclusions

MultiBatch provides mainframe batch scheduling capability for the NonStop platform. This functionality can be applied to both GUARDIAN and OSS executables.

Users can exploit the parallel nature of this architecture by building a parallel batch schedule that can execute concurrently across one or more networked nodes.

Configuration, execution and monitoring can be performed from a single point in a secure audited environment.

All the existing program execution options, such as CPU selection, high pin nomination and defines are supported.

Sophisticated scheduling and calendaring facilities can be used to provide a flexible selection of jobs for any given day.

Job execution can be performed through TACL processes, but the product also utilises a low-level procedure call interface to invoke new processes. This approach provides a more efficient solution and negates the need to build and maintain TACL macros.

The system will monitor itself by scanning for “not executed” or “over executed” steps. Alternatively, failures can be escalated to EMS so that they can be displayed on management consoles.

Insider Technologies provides sister products, Reflex and Sentra, that can be used as MultiBatch management consoles. Further details are available on request.

System Limits

LIMT	NUMBER
Number of BMON processes	No imit
Number of batch jobs per BMON process	999
Number of Job to Job dependencies	10 per job
Number of Segment to Segment dependencies	10 per segment
Number of assigns per batch job	60
Number of parameters per batch job	100, or less if the total length of the parameter names + total length of the values + (2 * number of parameters) > 1024
Number of defines per batch job	10
Length of startup parameter string	80 characters
Number of batch jobs monitored by Watcher	20,000
Number of Global parameters	2,000
Number of Local parameters	No Limited
Number of jobs in DS-EventTimer database	No limit

Insider Technologies is a UK-based software and services company, operating in the Financial and Messaging markets.

It provides Service Management, Tracking, Bespoke Software and Information Mediation solutions.

A cross section of our customers include Banking and Financial Services, Telecommunications Providers and Government and Military Institutions.

For details about the full range of products and services available from Insider Technologies Limited, please contact our Product Development Centre at:

Insider Technologies Limited
2 City Approach
Albert Street
Eccles
Manchester
M30 0BL
United Kingdom

Tel: +44 (0)161 876 6606
Fax: +44 (0)161 868 6666

e-mail: support@insidertech.co.uk
Website: http://www.insidertech.co.uk

Microsoft Partner
Gold Application Development