

Smart/COMMIT

*Introduce COMMIT logic into programs
without changes to source code or JCL*



- **Dramatically improve data concurrency and availability**
- **Convert programs originally developed without commit logic into well behaved applications**
- **Enable isolated programs and jobs to evolve as integrated system components**
- **Dynamically vary commit frequency**
- **Fully support applications which update DB2, IMS, MQ and/or other RRS compliant resources**

Overview

Applications and batch jobs no longer run in isolation. The demands of a disappearing batch window are forcing concurrency and availability requirements on legacy applications that were originally developed without commit logic (on the assumption they ran in isolation). But small applications inevitably grow, batch windows shrink and requirements for concurrency increase. As such, organizations can no longer tolerate applications that run for 3 hours, abend and then spend 6 hours in back out processing — all in standalone mode.

Smart/COMMIT is intended for batch applications originally developed without commit logic or that issue too few commits to satisfy present concurrency requirements. Smart/COMMIT lets you define the unit-of-work external to the program -- without the need to change source code or JCL or to recompile or relink the application. The logic which detects the completion of a unit-of-work is located within Smart/COMMIT rather than the application source code. Smart/COMMIT triggers a commit (whose scope includes all RRS compliant resources) when processing reaches a point of consistency you define — so data integrity is preserved.

Retrofitting existing applications with COMMIT logic has been a time consuming and problematic exercise — until now. Smart/COMMIT is suitable for a wide range of legacy applications, including many that are difficult or impossible to modify.

Benefits

- Enables batch applications to run concurrently with other batch and online processes
- Reduces rollback time
- Shortens recovery time following an abnormal termination
- Reduces lock contention and lock escalation
- Reduces the need for a discrete 'batch window' where batch and online run separately
- Lets isolated programs and jobs evolve as parts of growing, integrated systems
- Releases locks to limit the length of time resources are held exclusively by a unit-of-work
- Eliminates 'endless' rollbacks from tape

Features

- Dynamic commit pacing lets you control commit frequency during execution to adapt to a changing mix of concurrent programs and users — all without program changes.
- Runtime monitoring, control and diagnostic facilities
- A prediction facility provides completion time estimates for applications and jobs
- Authorized users can explicitly request COMMITs, from the MVS console or through the Smart/COMMIT monitoring dialog.

Easy Implementation

Since the unit-of-work is specified external to the application, it is not necessary to modify source code or to even have access to the source. This makes it easy to retrofit existing applications as well as third party software supplied in object code only (OCO) format. Smart/COMMIT lets you define the unit-of-work without changes to source code or JCL.

The following example directs Smart/COMMIT to issue a commit whenever a control break field changes in the input driver file.

```
//STEP1 EXEC PGM=EXAMPLE
//TRANFILE DD DSN=input.driver.file,DISP=OLD <=== (1)
//OUTFILE DD DSN=output.file,DISP=OLD
//SYSOUT DD SYSOUT=*
//SRSPRINT DD SYSOUT=*
//RAINPUT DD *
COMMITFILE(CONTENT,TRANFILE,(1,8)) <=== (2)
//
```

where

- (1) The TRANFILE DD statement defines the input 'driver' file
- (2) Smart/COMMIT issues a commit when the 'control break' field changes value. This field starts at position 1 within the input record for a length of 8 bytes.

External COMMIT Criteria

COMMIT processing can be triggered based upon

- The count or contents of records read from a specific file
- The number of rows or the content of columns fetched from a specific DB2 cursor
- Elapsed time— as a function of time-of-day, day-of-week and/or shift
- The number of SQL statements issued since the last commits
- The number of SQL statements that change data (via DELETE, INSERT, and/or UPDATE requests) issued since the last commit

The above criteria can be further governed by commit frequency controls which can themselves be changed dynamically — during program execution.

Prerequisites

Smart/COMMIT is easily installed without an IPL and runs with all supported releases of z/OS, OS/390, DB2, IMS, WebSphere MQ and other RRS compliant resource managers.

Smart/COMMIT supports all DB2 attachment facilities when the external commit criteria are based on file I/O and/or elapsed time. In contrast, unit-of-work definitions based upon SQL activity require the Smart/RRSAF product from Relational Architects.

Smart/COMMIT — don't run batch without it!

Contact the z/OS batch experts at 800 776-0771