Smart/RESTART and Smart/RRSAF – Summary of Functionality

Smart/RESTART

Smart/RESTART provides batch applications with robust and symmetric COMMIT and ROLLBACK services whose scope encompasses changes to *all* resources a batch application may access. In its role as resource manager and syncpoint coordinator, Smart/RESTART guarantees that updates to RRS compliant resources (such as DB2 and WebSphere MQ) stay 'in sync' with the application's sequential file and cursor position, program storage and random VSAM updates. This 'all or nothing' guarantee applies even if the application program, the system or a participating resource manager fails. This enables the application to restart from near the point of failure – with *all* resources in a consistent state. Restart capability can be added transparently, with little if any change to source code – provided the application requests checkpoints at unit-of-work boundaries and resumes execution at a unit-of-work boundary in the event of failure and subsequent restart.

Smart/RRSAF

Smart/RRSAF lets DB2 batch programs run with regular JCL, so they look and behave like standard OS jobs. Smart/RRSAF enables your applications to update DB2, MQ, IMS and/or any other RRS compliant resources within a single unit of recovery. Smart/RRSAF supports *dynamic* checkpoint frequency and lets you flexibly trace SQL, RRSAF, CAF and IFI activity. Applications can utilize either the RRSAF or CAF attachments to DB2 – without the need for user code to explicitly manage a DB2 connection and thread.

Repositionable Sequential Access Method (RSAM)

Smart/RESTART transparently uses its Repositionable Sequential Access Method (RSAM) to access sequential files that must be repositioned on restart. RSAM allows I/O to sequential files to be performed using verbs native to the programming language in which the application is coded. For example, COBOL programmers can code standard SELECT and FD statements and use native OPEN, CLOSE, READ and WRITE verbs.

Smart/LOGGER

Smart/LOGGER expands the scope of a Unit-of-Work to include both sequential and random changes to VSAM data.

Smart/RESTART and Smart/RRSAF – Summary of Functionality

Manual and Automatic Restart

Smart/RESTART supports both manual restart (by submitting the same JCL used for the initial run) as well as automatic retry (following the occurrence of site defined SQL codes, error conditions and abends). Smart/RESTART intercepts all abends and non-zero SQLCODEs. The list of failure conditions eligible for automatic restart (and the number of retries permitted) can be defined at installation time or specified on an application basis. Automatic restart occurs without terminating the current job step, so job scheduling packages are unaware of jobstep failure and restart. Automatic retry is particularly suitable, in the view of many Smart/RESTART customers, following "resource unavailable" conditions wherein the unit-of-work is backed out in its entirety and automatically retried after a user specified interval. Note that Smart/RRSAF is a prerequisite for automatic retry.

Checkpoint Pacing

The checkpoint pacing feature lets you control checkpoint frequency during execution by governing whether an application's checkpoint request is actually processed. Each checkpoint an application issues is in fact a logical request which, when honored, results in a physical checkpoint. You can 'pace' the checkpointing frequency by varying the number of logical checkpoint requests Smart/RESTART must receive before it physically issues a Smart checkpoint. The decision to issue a physical checkpoint can be based on factors such as number of input records processed, number of output records written, elapsed time, the number of logical checkpoint requests received, time of day, heuristic code and/or operator request. Smart/RESTART also lets you predefine different checkpoint frequencies for separate shifts.

Smart/RESTART also allows you to issue checkpoints in applications that don't request them by defining checkpoint criteria external to the application. For example, applications driven by a sequential input file can be directed to take a checkpoint after reading a specified number of records or when a 'control break' occurs.

Efficiency and Overhead

Smart/RESTART is extremely efficient with millions or tens of millions of updates. Smart/RESTART's Repositionable Sequential Access Method (RSAM) makes use of optimized, parallel I/O techniques that outperform sequential file access via QSAM. Sequential file repositioning is particularly efficient by going directly to a SAM TTR (denoting the proper volume and physical block) or VSAM RBA in one operation. As such, the overhead associated with Smart/RESTART is often negligible.

Restart Enablement and Removal

Smart/RESTART is easy to implement (and easy to remove should the need arise). Just the application's main driver module needs to be restart enabled while lower level routines can typically be ignored.

Smart/PRECOMPILER (included with Smart/RESTART) runs as a discrete job step during program preparation to automatically add the data structures and procedural logic necessary to restart enable the application. Smart/PRECOMPILER works in a manner analogous to the DB2 Precompiler and CICS Translator in that 'restart enabled' code is passed to a downstream compile or assembly step. As such, developers maintain the original source code rather than the precompiled source module created at program preparation time. To remove the product from an application, simply re-prepare its main driver module without Smart/PRECOMPILER.

Restart enablement can also be accomplished by embedding COBOL COPY members or PL/I INCLUDE text (two data and two procedural) within the main driver module. This technique obviates the need to create a special program preparation procedure for main modules and can be removed by first nulling out the COPY or INCLUDE text and then rebuilding the driver module.

A simple means is also provided to disable Smart/RESTART at execution time for selected runs of the application (as opposed to removing it permanently).

Smart/RESTART and Smart/RRSAF – Summary of Functionality

Smart/RESTART – *what it does for you*

- Provides a reliable means to *resume* an abended job from a checkpoint rather then *rerun* from the beginning
- Supports all RRS compliant resource managers (such as DB2, MQ and IMS) and serves as a sync point coordinator
- Smart/RESTART also functions as a resource manager for application storage, QSAM files and randomly accessed VSAM datasets
- Keeps all resources in sync even if the application program, the system or a participating resource manager fails. Smart/RESTART's checkpoint and rollback services are implemented *symmetrically* to keep all resources in a consistent state
- Implements restart functionality in system software rather than within your application programs
- Provides common and simplified operating procedures for restart that let you use the *same* JCL for both initial and restart runs
- Lets you commit after *every* Unit-of-Work without concern for performance (Checkpoint Pacing)
- Lets you take checkpoints in applications which do not currently issue them (to enable concurrency)
- Lets you change and recompile failing programs if necessary, and then restart them
- Supports automatic retry of failing SQL statements as well as automatic restart
- Provides an ISPF dialog to monitor restartable applications and predict their completion times
- Provides useful diagnostics such as compile date, compile options, record count and record contents at abend time
- Interoperates with Language Environment condition handling and abend percolation
- Supports all versions and releases of COBOL and PL/I (both with and without LE) and lets you mix compile and runtime library releases *without restriction*
- Interoperates with debugging aids like XPEDITER© and the IBM Debug Tool and fully supports diagnostic facilities like ABEND-AID©
- RSAM ensures you never lose committed records, write short blocks or write duplicate records. Programs perform repositionable I/O using native language constructs like OPEN, CLOSE, READ and WRITE

Smart/RRSAF lets you harness DB2 in batch without the hassle

- DB2 batch applications look and behave like standard OS jobs
- Prevents SNAFUs in DB2 batch production – job-step completion codes are set correctly
- Ensures proper dataset disposition and the correct execution of downstream jobsteps
- Provides granular SMF accounting by application program name rather than 'lumping' all SMF charge back statistics under the name of the TSO Terminal Monitor Program
- Transparently supports both RRSAF and CAF
- Expands the commit scope to include all RRS compliant resources
- Lets you vary COMMIT frequency dynamically
- Lets you flexibly trace SQL, RRSAF, CAF, IFI and DB2 command activity
- Provides built-in monitoring and reporting facilities to help identify and resolve problems with SQL performance and contention
- Lets you convert selected load modules or entire libraries from any attachment to any other DB2 attachment

Smart/RESTART and Smart/RRSAF – Summary of Functionality

Smart/RESTART and Smart/RRSAF – *Feature and Benefit Summary*

- Provides batch applications with restart capability that is as simple as possible to implement and deploy
- Many applications which implement a Unit of Work loop can run restartably without source changes
- Conserves precious 'batch window' time
- Retries automatically from 'resource unavailable' conditions like SQLCODE -911
- Extends the scope of a Unit of Work to include program storage, sequential files and VSAM datasets
- Smart/RESTART keeps ***all*** resources the application accesses consistent and 'in sync'
- Repositionable Sequential Access Method (RSAM) improves I/O performance by 10% over QSAM
- Checkpoint pacing lets you adjust the physical checkpoint frequency to dynamically optimize the tradeoff between performance and concurrency
- Prediction of jobstep completion times helps improve management of the z/OS batch workload
- Provides simplified operating procedures that allow the use of ***same*** JCL for both initial and restart runs