

BSTAR Backup and Recovery for Library Units (BRLU)
Version 6
Implementation for model application

Title: *BSTAR BRLU implementation for model application*
File: *BSTAR BRLU implementation for model application.pdf*
Build Date: *12/08/2009 21:53 Build Version: 6.4.2.0*

Table of Contents

1.	Introduction	3
2.	Model application	4
2.1	General description of the model application.....	4
2.2	Recovery procedures of the model application	4
2.3	Model application problems	5
3.	BRLU, help!	6
3.1	Checkpoint-based recovery policy.....	6
3.1.1	IBM journals	6
3.1.2	BRLU recovery mode	6
3.1.3	BRLU checkpoints	7
3.1.4	BRLU recovery procedure.....	8
3.2	Backup transformation	9
3.3	CL command substitution (DB copy acceleration).....	12
4.	Conclusion	16

1. Introduction

This document describes a model implementation of BSTAR Backup and Recovery for Library Units (BRLU) tool. The purpose of the document is to illustrate and showcase the key features of BRLU.

2. Model application

2.1 General description of the model application

The model application, or rather sub-application, is a daily batch executed during the night hours when, typically, there is no other activity in the system.

The model batch contains multiple processing steps (STEP1, STEP2, STEP3, etc.) executing in the natural order. Each step consists of multiple concurrent processes performing database updates. For each step of the batch, in order to start it all processes from the previous step must successfully complete.

It is possible to optionally suspend execution of the batch at the end of each given step (e.g. in order to perform HW maintenance) and then resume execution from the point of interruption.

A database backup (full or incremental) is typically taken before the daily batch process starts, but specific backups can be taken by multiple components of the daily batch.

Some of the data from the main application database can be copied and saved for later report generation.

Typically, the daily batch is a very time consuming process taking at least a few hours to complete.

The main application database is large and the daily backup takes between 30 minutes and 1 hour.

2.2 Recovery procedures of the model application

Each process of the daily batch makes use of commitment control and is designed not to crash when a logical data error is encountered. In certain cases, however, some of the daily batch processes may end abnormally in case of logical corruption of data. On average, this happens once or twice a year.

When a process in a certain step of the daily batch crashes, it is sometimes possible to fix the error and restart it. But there are cases when logical corruption of the data causes multiple processes of a step (and sometimes several steps) to generate incorrect data. If only failed components are restarted, the databases and printed documents may contain incorrect information. That's why in these rare cases the daily batch has to be recovered.

The only way to recover the whole batch is to restore the database from the most recent backup media, fix the data, and then re-submit the batch. Unfortunately, because the batch takes a long time to run as it is, database restore and the second run can spill into the next business day. This is a very unpleasant scenario meaning lost business and unhappy customers.

2.3 Model application problems

There are two major problems with the model application as it was described:

- long daily batch processing
- occasional inefficiency of the recovery procedure

Specifically, the daily batch includes the following long-running components:

- daily backup
- CPYLIB- and CRTDUPOBJ-based database copy process (creation of the “reporting” copy of the database)

3. **BSTAR, help!**

BSTAR BRLU can help dramatically improve the recovery situation with the model application without changing a single line of code in the latter. Specifically, what BSTAR BRLU can help with is

- introduction of the new checkpoint-based (rather than backup based) batch recovery policy
- automatic transformation of traditional backups to save-while-active backups (if the checkpoint-based recovery policy is in place, the level of system protection is no worse than without BSTAR BRLU)
- accelerate DB copy operations

The rest of this chapter contains detailed information on how each of the above objectives can be achieved.

3.1 **Checkpoint-based recovery policy**

3.1.1 **IBM journals**

Checkpoint-based recovery policy is actually IBM journal-based recovery policy. RMVJRNCH (Remove Journal Changes) system command provides the basic functionality, but, unfortunately, it suffers from several serious afflictions. The two most important of them are poor performance due to mandatory single-streamed recovery and the need to know journal entry numbers related to business-defined points in the application day cycle. So, RMVJRNCHG is actually an API that can be used to build a good recovery tool rather than the tool itself.

But there is yet another nasty catch, and that is, many processes, traditional (OPM/ILE) or Java-based, are not “recoverable” using IBM journals. Certain CL commands, such as, for example, CLRPFM, or data manipulation statements, such as SQL “DELETE * from Table” cause the previous contents of the database to be irrevocably lost.

BRLU addresses all of the above problems. Three major components of the solution are BRLU recovery mode, BRLU checkpoints, and BRLU journal recovery procedure.

3.1.2 **BSTAR recovery mode**

In order for CL commands executed in a job, such as CLRPFM, CPYF, CPYFRMQRYF, DSPFD and a few others, to become recoverable, jobs must execute in BSTAR recovery mode. To enter this mode interactive jobs can execute STRBSTMOD command. Batch jobs submitted from such interactive jobs usually inherit BSTAR mode from the jobs submitting them. More detailed information regarding BSTAR mode can be found in *BSTAR BRLU Product Manual*.

```

tn5250 - - cinimex
File Edit View Help
Start BSTAR mode (STRBSTMOD)

Type choices, press Enter.

Library unit . . . . . > TEST          Character value, *LIBL
Control library name . . . . . > TEST      Name, *UNIT
Journal name . . . . . > TEST          Name, or *NONE, or *UNIT
  Library name . . . . . > TEST          Name or *LIBL
List of protected libraries . . > TEST      Name, *UNIT, *NONE
  + for more values
Unit hot library name . . . . . > *NONE     Name, *UNIT, *NONE, *STD
Automatic journaling . . . . . > *NO       *YES, *NO, *UNIT
Asynchronous backup flag . . . . > *YES     *YES, *NO, *INDEP, *UNIT
Command replacement enabled . . > *YES     *YES, *NO, *SOURCE, *UNIT

Additional Parameters

Records in CLRPFM stream . . . . > 50000    50000-999999, *NONE, *UNIT

Bottom
F3=Exit  F4=Prompt  F5=Refresh  F10=Additional parameters  F12=Cancel
F13=How to use this display  F24=More keys

5250 MW 037/005
    
```

Library unit is a set of IBM i libraries containing application database objects. One of the libraries of the set has to be designated as the control library (that’s the library where BSTAR will create its configuration objects). It is assumed that the control library is always in the library list of every job started on behalf of the application. List of protected libraries contains those libraries that must be protected from undesirable effect of CLRPFM and similar commands.

Description of other STRBSTMOD command parameters can be found in *BSTAR BRLU Product Manual*.

When a job enters BSTAR mode, all subsequent calls to unrecoverable commands, such as CLRPFM, are intercepted and substituted with their recoverable clones. For instance, in case of CLRPFM a special BSTAR “Clear Physical File Member” process is submitted. This process deletes records in the target member rather than clearing the entire member. In order for this process to be efficient BSTAR BRLU can automatically submit multiple record deletion streams, whose number is configurable.

3.1.3 BSTAR BRLU checkpoints

At the appropriate logical point of the daily application cycle a BSTAR checkpoint can be taken:

```

tn5250 - - cinimex
File Edit View Help

Create checkpoint (CRTCKP)

Type choices, press Enter.

Library unit . . . . . > TST           Name, *LIBL
Checkpoint id . . . . . > CKPT1        Name
Checkpoint definition . . . . . *EXPLICIT  *EXPLICIT, *IMPLICIT
High availability mode . . . . . Y       Y, N
MIMIX primary library . . . . . MIMIX    Character value
MIMIX data group name . . . . . MIMIXDG  Character value
MIMIX System 1 name . . . . . SYSTEM1  Character value
MIMIX System 2 name . . . . . SYSTEM2  Character value

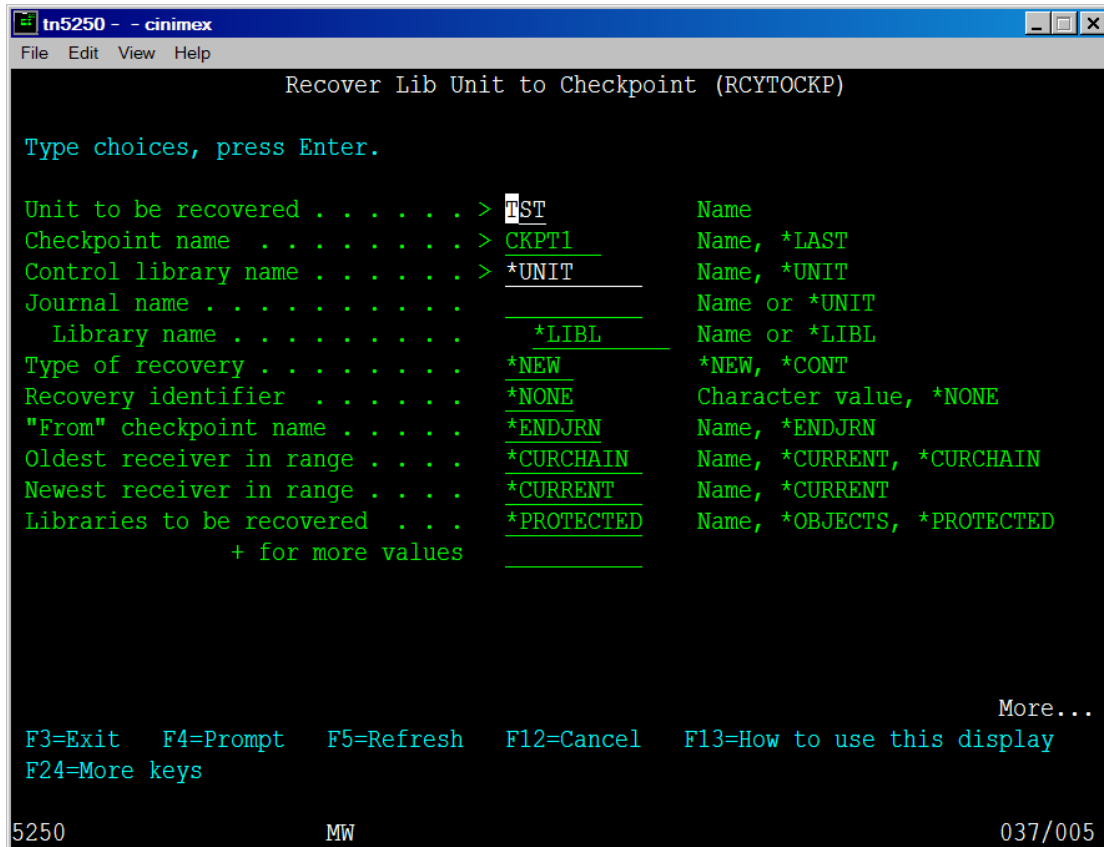
Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys

5250 MW 037/009
    
```

This can be done manually; alternatively, application code can be modified to execute CRTCKP commands at appropriate moments. As can be seen from the example above, MIMIX High Availability solution can optionally be supported.

3.1.4 BRLU recovery procedure

In case of a logical problem in either the daily batch or, really, any other part of the application, the application database can be recovered to any of the previously defined checkpoints:



Many additional parameters are supported, including those defining recovery object filters, optional multi-streaming, etc. The process of recovery can then be continued to one of the earlier checkpoints, if required.

Additional advanced recovery facilities can completely automate complex recovery processes involving a special recovery subsystem, multiple main storage pools, journal verification and early object locking warnings.

3.2 Backup transformation

When checkpoint-based recovery policy is in place, it is safe to change the mode of database backups to save-while-active. Unfortunately, sometimes it means changing the application code or writing new code. BSTAR BRLU does not require any of this. If “asynchronous backup” feature of BSTAR is installed, the following command can activate it for the entire application:

```

tn5250 - - cinimex
File Edit View Help
Start BSTAR mode (STRBSTMOD)

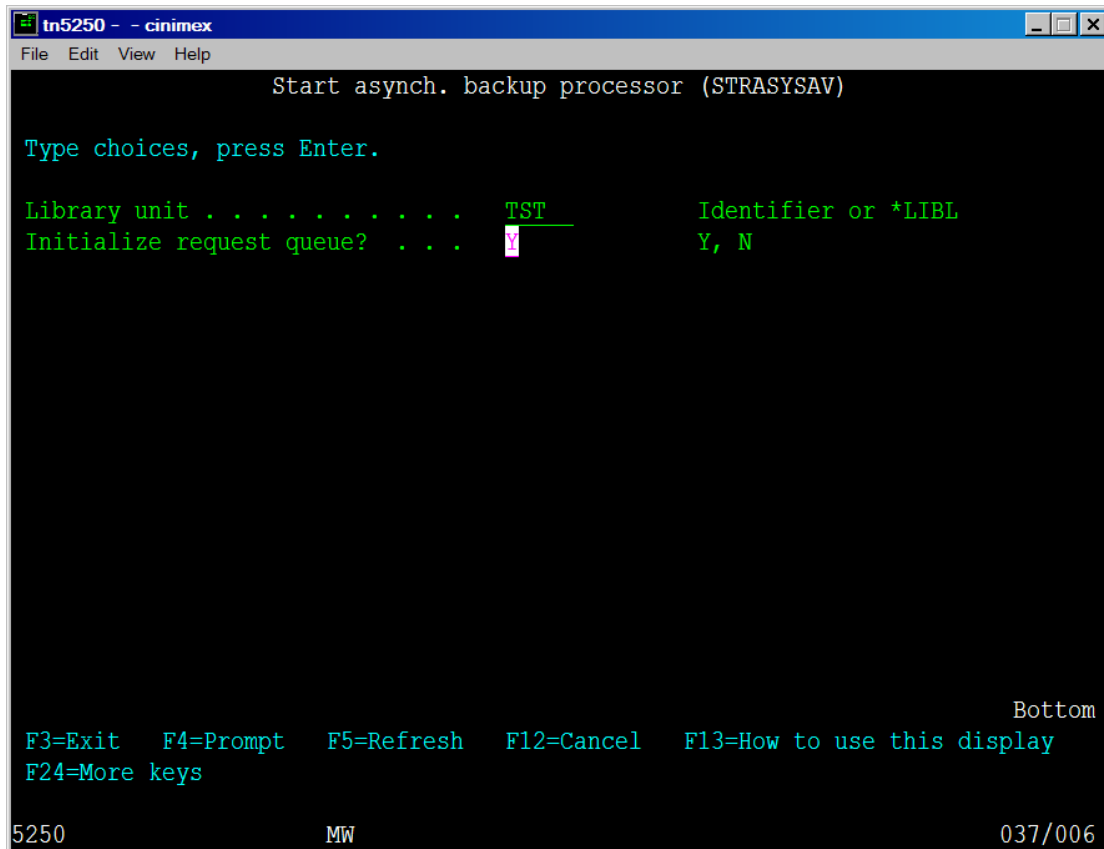
Type choices, press Enter.

Library unit . . . . . > TST          Character value, *LIBL
Control library name . . . . . TEST      Name, *UNIT
Journal name . . . . . TEST           Name, or *NONE, or *UNIT
  Library name . . . . . TEST         Name or *LIBL
List of protected libraries . . TEST     Name, *UNIT, *NONE
  + for more values
Unit hot library name . . . . . *NONE   Name, *UNIT, *NONE, *STD
Automatic journaling . . . . . *NO      *YES, *NO, *UNIT
Asynchronous backup flag . . . . . *YES *YES, *NO, *INDEP, *UNIT
Command replacement enabled . . *NO     *YES, *NO, *SOURCE, *UNIT

Bottom
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display   F24=More keys

5250                MW                042/011
    
```

The other requirement is to start the BRLU background backup process job for the application:



After that, the following request executed from a job running in BSTAR mode

```
SAVLIB LIB(EANDATAP) DEV(TAP02)
```

Will be automatically transformed to

```
SAVLIB LIB(EANDATAP) DEV(TAP02)
  SAVACT(*SYNCLIB) SAVACTMSGQ(CRCVQS/SAVRQMTST)
```

and sent to the background processor for execution. The job requesting the background is released and proceeds to the next application step. The status of the request can be displayed like this:

```

tn5250 - - cinimex
File Edit View Help
                                BSTAR asynchronous backup processor
Unit: TST                               14/08/09 11:59:22
Refresh in seconds . . . . . 015
Server job . . . . . 073817/CINOWNER/SAVJOBTST
Server job state . . . . . Active/LCKW/SYN |

Type options, press Enter.
1=Display request | 4=Remove request | 5=Work with job

                                Pending backup requests

Opt Date   Time   Job           User           Number Command
-----
1 140809 115822 QPADEV000B CINOWNER    073816 QSYS/SAVLIB LIB(EANDATAP) DEV(TA

Enter=refresh PF1=help PF2=indep.requests PF12=exit PF19=auto-refresh off

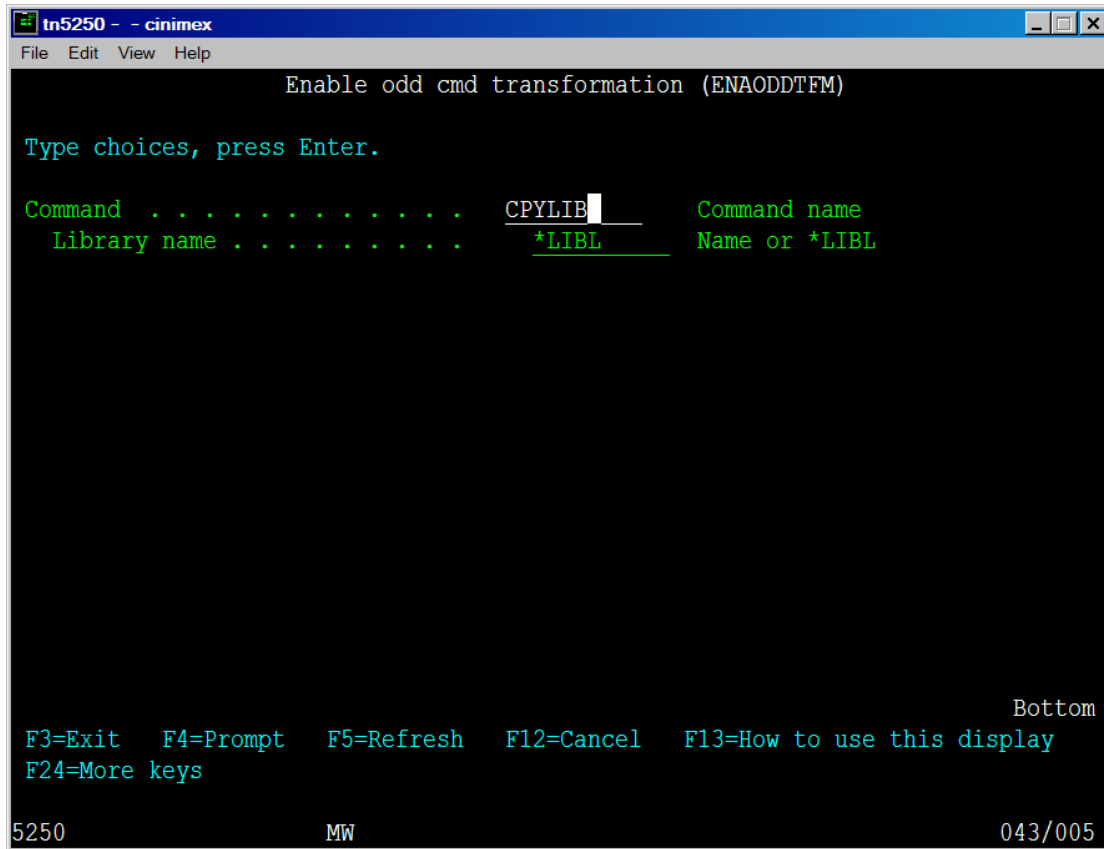
5250                               MW                               002/014
    
```

The backup processor can handle parallel, as well as serial, backups; it supports BRMS, backups to both tapes and saves files, etc. The content of the backup media is exactly the same as it would be if BSTAR BRLU were not installed.

3.3 CL command substitution (DB copy acceleration)

In the model application database copy is based on CPYLIB command – very inefficient for large databases. Since this command is embedded in the application, it is not easy to change it. Even if the application is changed and a more efficient way of copying the database is implemented (e.g. SAVRSTLIB), it will most probably still be a synchronous process, that is, the application will have to wait until the database has been copied.

BSTAR BRLU help comes in the form of its “CL command substitution” feature. Practically every CL command can be enabled for substitution/replacement:



In order to implement command substitution BSTAR BRLU registers QIBM_QCA_CHG_COMMAND exit program for the specified command. Then, command substitution has to be enabled for the application (the substitution parameter appears only if the appropriate BSTAR feature is installed):

```

tn5250 - - cinimex
File Edit View Help
Start BSTAR mode (STRBSTMOD)

Type choices, press Enter.

Library unit . . . . . > TST           Character value, *LIBL
Control library name . . . . . TEST       Name, *UNIT
Journal name . . . . . TEST           Name, or *NONE, or *UNIT
  Library name . . . . . TEST           Name or *LIBL
List of protected libraries . . TEST       Name, *UNIT, *NONE
  + for more values
Unit hot library name . . . . . TEST2     Name, *UNIT, *NONE, *STD
Automatic journaling . . . . . *NO       *YES, *NO, *UNIT
Asynchronous backup flag . . . . *YES     *YES, *NO, *INDEP, *UNIT
Command replacement enabled . . *YES     *YES, *NO, *SOURCE, *UNIT

Bottom
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display   F24=More keys

5250                               MW                               037/007
    
```

The only remaining thing is to define command substitution rules for CPYLIB. This can be done, for example, as follows:

```

tn5250 - - cinimex
File Edit View Help
Columns . . . : 1 71 Edit CRCVQS/CRCVCMdTST
SEU==> CPYLIB1
FMT ** ..... 1 ..... 2 ..... 3 ..... 4 ..... 5 ..... 6 ..... 7
***** Beginning of data *****
0001.00 CPYLIB FROMLIB(LIB@P1) TOLIB(LIB@P2) CRTLIB(*NO)
0002.00 /**/
0003.00 SAVRSTLIB LIB(LIB@P1) RMTLOCNAME(INTRA) OMITOBJ((LIB@P1/*ALL +
0004.00 *SQLPKG) ACCPTH(*YES) RSTLIB(LIB@P2)
***** End of data *****

F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle
F16=Repeat find F17=Repeat change F24=More keys

5250 MW 013/009
    
```

The substitution definition is provided as a source file member linked to a particular library unit (application). @P1 and @P2 in the picture above are substitution variables. Basically, this substitution definition would cause every CPYLIB command entered with no additional parameters to be replaced with SAVRSTLIB command. The library names, of course, will be preserved. All SQL packages will be omitted.

Furthermore, if BSTAR asynchronous backup feature is installed and configured, the target SAVRSTLIB command will be sent to the background processor for execution. Thus valuable time will thus be saved.

If SAVRSTLIB operation is unsuccessful, it will always be possible to use checkpoint recovery to return to the point immediately preceding CPYLIB. That is, if the appropriate checkpoint was defined.

Command substitution facility can be used for all sorts of purposes, including automatic definition of BSTAR BRLU checkpoints.

4. Conclusion

BSTAR BRLU is a easy-to-use tool that can help implement IBM journal-based recovery policies while significantly improving performance characteristics of almost any IBM i application.