Syncpoint Processing Options

This section describes the available Syncpoint processing options.

Overview of Syncpoint Processing Options

The normal rules of transactional programming under CICS and other TP systems are different from those that are familiar to Adabas programmers. Specifically, in a standard CICS application, a screen I/O normally means that pending changes are committed, and locks are freed, whereas Adabas allows a transaction to span screen I/Os. Indeed, Natural utilities such as SYSMAIN could behave differently when the ATM CICS Resource Manager Interface is in use, because they are written in such a way that they expect transactions to remain open across screen I/Os.

With the advent of ATM, this difference in programming styles led to two requirements for the ATM CICS RMI implementation:

- it should be possible for pending Adabas changes to remain uncommitted across pseudo-conversational task end;
- it should be possible to have ISNs released from held status, automatically, when a CICS syncpoint occurs.

The first requirement makes the introduction of ATM easier for sites that have taken advantage of the possibility of keeping Adabas transactions open across screen I/Os. It also allows Natural utilities to execute as before. The second brings Adabas behavior into line with standard behavior in CICS and other TP environments, and is ATM's normal mode of operation. The two are, of course, incompatible, and are therefore implemented as options through the `TRANMODE` job parameter.

TRANMODE=MESSAGE Option

This is the default option, and defines ATM's standard way of processing syncpoints. Processing of a message always terminates with a syncpoint. Normally this means that a screen I/O causes pending changes to be committed. If an external transaction coordinator is in control of a transaction, an `ET` or `CL` command will trigger a commit syncpoint by the external coordinator; a `BT` command, or an `OP` to a changed database, will trigger a rollback syncpoint by the external coordinator. Held ISNs will be released, or will remain held, according to the setting of the extended hold option, and any P or M command options. An unsolicited syncpoint from an external transaction coordinator will cause ATM to commit or back out pending changes to all databases and release all held ISNs. In any case, the commit process will be synchronized for all `DTP=RM` databases; changes to other databases will be committed or backed out after completion of the syncpoint processing.

TRANMODE=DYNAMIC Option

This option allows existing Adabas applications (including Natural utilities such as SYSMAIN) to execute under the ATM CICS RMI without being affected by the unsolicited CICS syncpoints that occur at pseudo-conversational task end. ATM will honor all rollback syncpoints, whether they originate from BT or OP commands, from a CICS command, or from CICS itself. ATM will also honor commit syncpoints triggered by ET or CL commands, but it will ignore other commit syncpoints.

This option is appropriate for the Natural utilities, and for applications which keep transactions open across screen I/O operations. However, it might not be suitable for applications that execute under the CICS RMI and change both Adabas and non-Adabas resources. If such an application encounters an unsolicited commit syncpoint (when a screen I/O occurs, for example), its non-Adabas changes will be committed, but the Adabas changes will remain uncommitted until an ET or CL command is executed. That is, all syncpoints will continue to affect other resource managers (such as DB2) exactly as they did before, regardless of the behavior of ATM.

Note:
If you use TRANMODE=MESSAGE under the CICS RMI, and execute Natural using an ADAMODE setting that causes Natural to execute two parallel Adabas sessions, you must make sure that Natural zaps NA44231 and NA45010 are applied, otherwise repeated response code 9, subcode 97 will occur, and other problems may follow. These zaps are both included in Natural version 3.1.6. You must also ensure that Natural's system session always begins with an OP command, otherwise the same problems will occur; you can do this by always supplying a non-blank ETID or by using the DBOPEN parameter.

The TRANMODE parameter has no effect for IMS/TM systems whose transactions are coordinated by RRMS, and whose local ATM runs with TMSYNCMGR=RRMS.

Overriding the Current TRANMODE Option

The TRANMODE option specified in the job parameters takes effect, by default, for all users executing in matching jobs. However, a user can dynamically change the TRANMODE setting for the user's own session by calling a utility program. A Natural CALLNAT program and a command-level CICS assembler program are provided for this purpose.

The supplied programs, PRMORIDE (Natural) and ATMORID (assembler) are functionally similar. They can be executed during the execution of a CICS task. They override the default TRANMODE setting for the user of the current CICS task. The new setting will remain in effect until it is overridden by another execution of either utility program, or until the user finally stops work or logs onto CICS again. The Natural programs can be used in a similar way in environments other than CICS.

Note:
In TP systems where Adabas users are identified by Terminal ID, and where a Terminal ID can be re-used by different users in succession, a new user can inherit the TRANMODE setting that was in effect for the previous user of the Terminal ID. Therefore, if the override facility is used in such a system, it should always be executed at the start of a new session, to ensure that the correct TRANMODE is set, rather than assuming that the default setting will be in effect.

The assembler program, ATMORID, is executed by CICS LINK. It expects a COMMAREA containing the following fields:

```
RETCODE DS F Output parameter - return code
ADANAME DS CL8 Input parameter - name of the Adabas link
*                               module through which the caller's commands
*                               are processed
PARMSTR DS CL16 Input parameter - new parameter string
DIAGS1  DS F Output parameter - error diagnostics
DIAGS2  DS F Output parameter - error diagnostics
DIAGS3  DS F Output parameter - error diagnostics
```

The parameter string must be one of the following:

```
TRANMODE=DYNAMIC
TRANMODE=MESSAGE
TRANMODE=?
```

The TRANMODE parameter replaces the SYNCMODE parameter of ATM version 1.2. For the sake of compatibility with programs that were written to operate with ATM version 1.2, SYNCMODE parameter overrides are also accepted as follows:

- SYNCMODE=ALL is equivalent to TRANMODE=MESSAGE
- SYNCMODE=ADABAS is equivalent to TRANMODE=DYNAMIC
- SYNCMODE=? will return either SYNCMODE=ALL or SYNCMODE=ADABAS, as appropriate.

After successful execution, the return code will be zero, and the PARMSTR field will contain the current TRANMODE setting. A return code of 4 indicates a CICS LINK error. Other values are ATM error codes, as described in the section Messages and Codes.

The subprogram PRMORIDE is the Natural equivalent of ATMORID. It expects two parameters:

- The parameter string (TRANMODE=MESSAGE,TRANMODE=DYNAMIC,or TRANMODE=?) (A16)
- A return code field (N4)

The following is a sample program that calls PRMORIDE:

```
DEFINE DATA
LOCAL
1 #PARM-ORIDE (A16)
1 #PARM-RET-CODE (N4)
END-DEFINE
INPUT #PARM-ORIDE(AD=I'_')
CALLNAT 'PRMORIDE' #PARM-ORIDE #PARM-RET-CODE
WRITE '=' #PARM-RET-CODE
END
```

If PRMORIDE detects an error it will set the return code (values and meanings are the same as for ATMORID) and display some diagnostic information on the screen.

Adabas Commands and External Syncpoints

ATM allows Adabas changes to be committed synchronously with non-Adabas changes, by interacting with external transaction coordinators. When an external transaction coordinator takes a syncpoint, ATM ensures that changed Adabas databases take part in the commit or rollback operation. By default, ATM also causes the external transaction coordinator to take a syncpoint whenever it detects that pending

changes are to be committed or backed out.

Usually, this means that every ET or CL command causes an external commit syncpoint, and any BT command causes an external rollback syncpoint to take place. However, there are cases in which this behavior might be different from what is required. For example, consider a CICS environment in which a Cobol program changes DB2, then starts a Natural session, expecting Natural to return control before a decision to commit or back out is taken. Natural can issue an ET command during LOGON processing, and CL commands at session end. By default, each of these commands (if issued under the same Communications ID as the Cobol program's commands) would cause a CICS SYNCPOINT to take place, and the first of these would cause the pending DB2 changes to be committed.

Job parameters are provided which can be used to change this behavior. See the descriptions of the SYNCONBT, SYNCONCL, and SYNCONET parameters. In the example described above, it would be appropriate to specify SYNCONCL=NO and SYNCONET=ET, so that the CL and ET commands generated by Natural would not cause a CICS SYNCPOINT.

By default, these job parameters take effect for all users executing in matching jobs. However, a user can dynamically change their values for the user's own session, by calling a utility program. This can be achieved using a supplied Natural CALLNAT program or (in a CICS system) command-level CICS program. The technique is the same as for changing the current TRANMODE setting, and the same Natural and CICS programs are used for the purpose. See Overriding the Current TRANMODE Option for details. The following parameter strings are accepted by the override programs:

```
SYNCONBT=YES
SYNCONBT=NO
SYNCONBT=?
SYNCONCL=YES
SYNCONCL=NO
SYNCONCL=?
SYNCONET=YES
SYNCONET=NO
SYNCONET=?
```

Note:
In TP systems where Adabas users are identified by Terminal ID, and where a Terminal ID can be re-used by different users in succession, a new user can inherit the parameter settings that were in effect for the previous user of the Terminal ID. Therefore, if the override facility is used in such a system, it is a good policy to ensure that it is always executed at the start of a new session, to ensure that the correct setting takes effect, rather than assuming that the default setting will be in effect.