

CICS Node Error Program Considerations for Natural

This section discusses CICS node error program considerations. The following topics are covered:

- Normal Situation
- Situations Not under Control of Natural CICS Interface
- Recovery Mechanisms
- Special Considerations
- Example Dummy Program

See also:

- Installing a CICS Node Error Program
For information on installing a CICS node error program, refer to the section concerning user-replaceable programs in the **CICS Customization Guide** for your CICS version.
-

Normal Situation

Whenever a Natural session is active, CICS resources such as thread storage, roll facility entries (that is, records in a VSAM RRDS file or in a CICS temporary storage queue), swap pool slots etc. are used.

If these resources are under the control of the Natural CICS interface, they are correctly released whenever a session terminates normally or abnormally.

Situations Not under Control of Natural CICS Interface

The following cases cannot be controlled by the Natural CICS interface:

1. A non-Natural program called by Natural issues an EXEC CICS ABEND CANCEL command or the equivalent CICS macro request: the CICS task is canceled without the Natural CICS interface receiving control to properly release all session resources.
2. Some CICS monitor products offer tools to purge CICS tasks, bypassing any abnormal termination exit set by the application. If a Natural task is canceled this way, the Natural CICS interface has no chance to release the resources still owned by the session.
3. A user disconnects a terminal from the CICS region (by switching the power off or using an adequate session manager function) while a Natural session is currently not active in CICS (pseudo-conversational screen I/O).

Recovery Mechanisms

The Natural CICS interface provides some recovery mechanisms to recover from such situations; for example:

Whenever a new Natural session is to be started, a table is scanned for another Natural session still active with the same terminal ID. If such a session exists, it is logically terminated, and all its resources are released prior to starting the new one.

However, it may take quite a long time between logically terminating the session and releasing its resources, and there may also be a security concern:

When the NCIPARM generation parameter COMARET is set to NO, the information to resume a Natural session is kept in a CICS temporary storage record with the terminal ID being part of the temporary storage queue name. If another CICS user tries to start Natural with this terminal ID, he/she will resume the old Natural session rather than starting a new one.

The third case in the above list is the most crucial one. CICS provides a node error program (NEP) exit interface, which can be used in these cases to trigger the Natural CICS interface to terminate the lost session. An appropriate program called NCIZNEP is provided in the Natural CICS source library (see Natural CICS Sample Programs); it must be called by a DFHZNEP node error program.

Special Considerations

There are still some items to be considered:

- With CICS versions prior to CICS/ESA 3.2, the CICS node error program had to be written using CICS macro level calls, whereas with CICS/ESA 3.2 and above, the CICS node error program must be a command level program.

Therefore, the Natural CICS source library contains two sample node error programs: XNCINEP1 for CICS versions prior to CICS/ESA 3.2, and XNCINEP2 for CICS/ESA 3.2 and above.

Both sample programs do not perform anything special for the Natural CICS environment, they merely call (via LINK) the NCIZNEP program, which then deals with Natural under CICS.

- DFHZNEP may already be customized for a specific installation; as only one node error program is possible, the logic of the relevant XNCINEPx program should be adapted to the existing DFHZNEP logic.
- In MRO environments, DFHZNEP must be installed in the TOR.
- When you are using the CICS storage protection feature with CICS 3.3 or above, the NCIZNEP program must be defined with EXECKEY(CICS).
- In the case described under 3. above, DFHZNEP may receive control more than once for various internal error codes, since each internal error code is related to a specific CICS error message, but there may be more than one error message resulting from a given action.
- The CICS control block constellation may have changed each time a node error program has been invoked, for example, the COMMAREA and NEXTRANSID information in the TCTTE may have been lost after a certain node error event.

In this case the NCIPARM parameter COMARET must be set appropriately, which means that you cannot choose a node error event for your node error program to be invoked when passing the Natural pseudo-conversational session restart data in a CICS COMMAREA that has already been cleaned up by CICS.

Example Dummy Program

If you want to know how many times and with what error codes DFHZNEP is invoked on certain actions and how the TCTTE should look, write a dummy node error program, which only issues CICS trace requests showing the requested information.

The following sample enables a DFHZNEP error processor to receive control for all possible error codes passed to DFHZNEP:

```
.
DFHSNEP TYPE=INITIAL
ORG NEPTT
DC 256X'03' invoke error processor '3' for ALL error codes
ORG ,
DFHSNEP TYPE=ERRPROC ,GROUP=3 ,CODE=49
```

```
.  
set up requested information and issue trace request(s)
```