Natural Roll Server Functionality

This document covers the following topics:

- Natural Roll-Server Overview
- Roll Server in a Single z/OS System
- Roll Server in a z/OS Parallel Sysplex Environment
- Roll File and LRB

See also *Natural Roll Server Operation*.

Natural Roll-Server Overview

With the Natural Roll Server, Natural can execute in a multiple-address-space system like CICS or IMS TM; these address spaces may be located in multiple z/OS images (z/OS Parallel Sysplex). You can, of course, also use the Roll Server if you are running a single z/OS system.

When Natural performs terminal I/O, it must save the application's context data (the thread): Before the terminal I/O is started, the thread is given to the Roll Server which keeps it in its Local Roll Buffer, or in the roll file. When the terminal I/O is completed, Natural requests the thread from the Roll Server, and continues the application. In a z/OS Parallel Sysplex environment, the Roll Server keeps information about the threads (the roll file directory) in a data structure in the Coupling Facility. Thus, it is possible for a Natural application to execute in different z/OS systems at different times: A thread can be given to the Roll Server on one system, and requested back from another system.

The Roll Server runs in its own address space. It provides its services as PC routines. In a z/OS Parallel Sysplex environment, one instance of the Roll Server must be started in each participating z/OS image.

A list of applied Roll Server Zaps is displayed by the Natural command `DUMP ZAPS`. In addition, the list of applied Zaps is written to `JESMSGLG` during Roll Server startup.

Note concerning Natural under CICS: The CICS System Recovery Table should include the z/OS system abend code `0D6`.

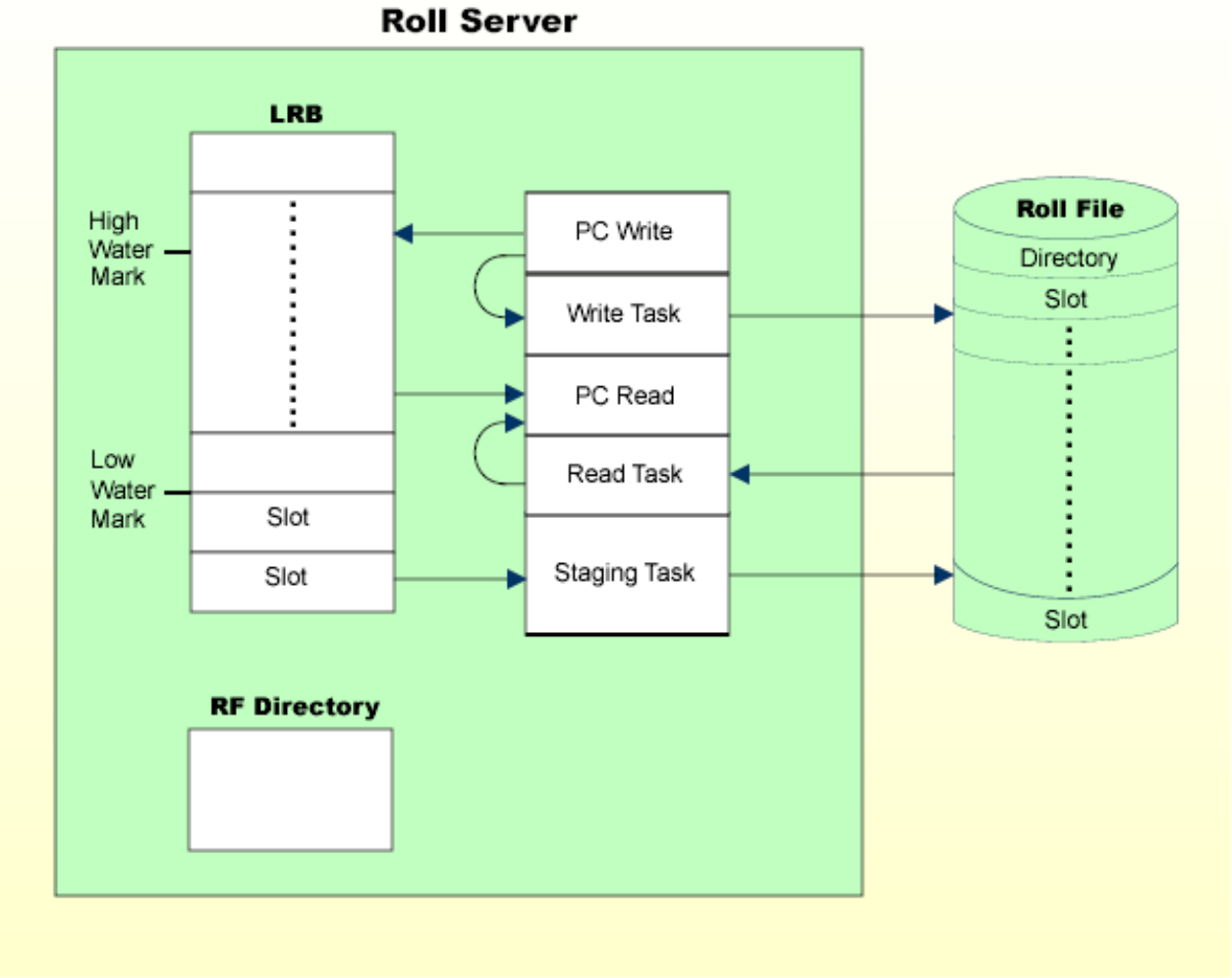
Roll Server in a Single z/OS System

When the Roll Server receives a thread through a write request (before terminal output), it checks whether enough space is available in the local roll buffer (LRB). If there is, the thread is copied to the LRB. If not, the thread is written to the roll file. The thread is also written to the roll file if it is larger than the LRB slot size. If the thread is larger than the roll file slot size, additional overflow slots are allocated to accommodate the thread. Allocation of overflow slots is restricted to the roll file that the Natural session was initially assigned to. If the roll file does not have enough free space to allocate the necessary overflow slots, an error is generated and the requesting Natural session terminates abnormally. Overflow slots are implicitely freed by a subsequent write request with a smaller thread.

When the Roll Server receives a read request for the thread (after terminal input), it tries to locate the thread in the LRB. If the thread is found, it is copied from the LRB to the requestor's address space. If not, the thread is read from the roll file and copied to the requestor's address space.

To ensure that the system performs well and that there is always enough space in the LRB, there are "water marks". If the LRB's high water mark is reached, the staging task is activated and copies the LRB content to the roll file until the low water mark is reached. Where the high water mark and the low water mark are placed is therefore an important issue of performance tuning. For more information on performance tuning, see the section Roll Server Performance Tuning.

Illustration of the Roll Server in a Single z/OS System:



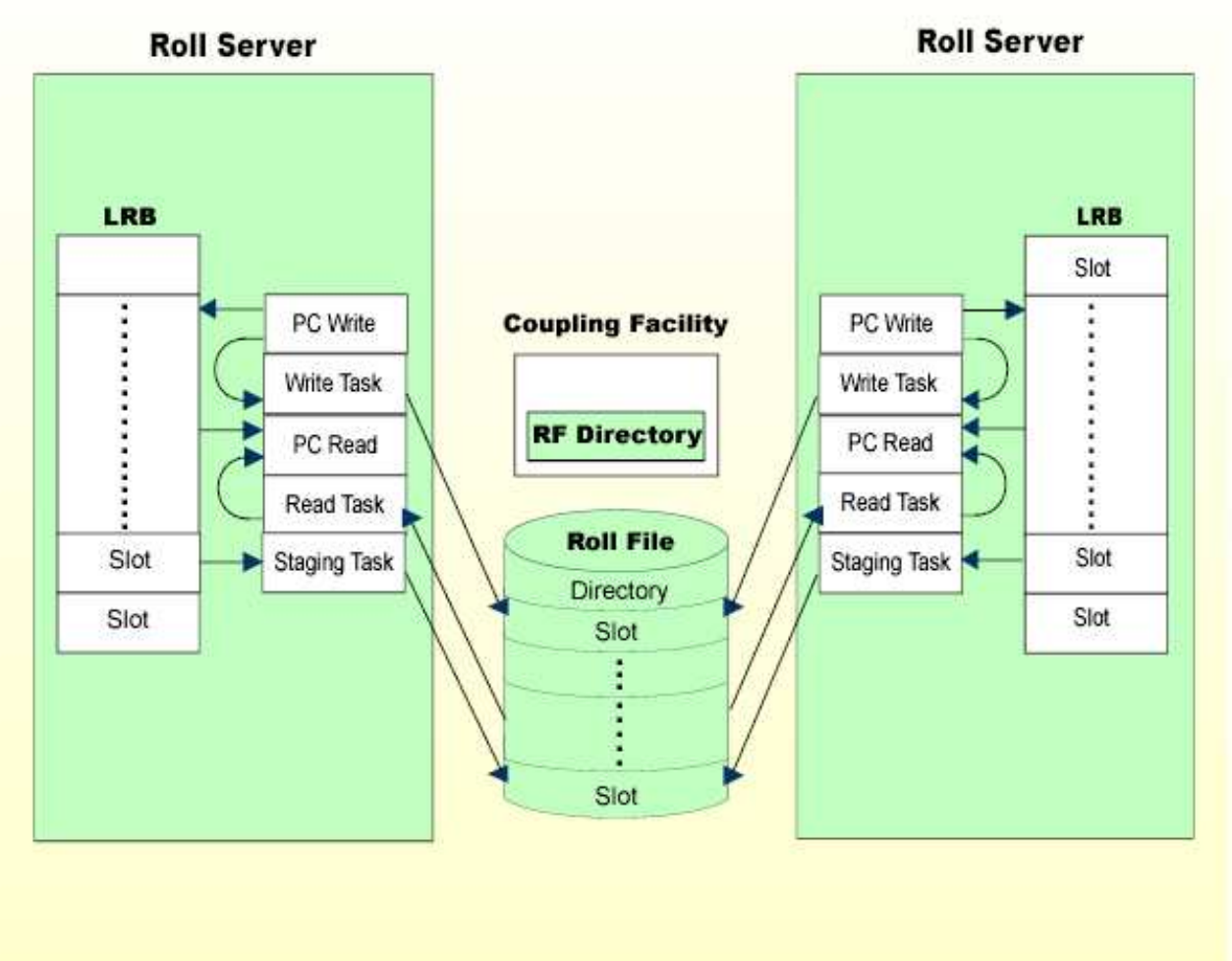
Roll Server in a z/OS Parallel Sysplex Environment

In a z/OS Parallel Sysplex environment, the Roll Servers in the participating z/OS images communicate through the Coupling Facility's (CF) XCF Signaling Services, and the roll file directory resides in a XES data structure.

When the Roll Server receives a thread through a write request (before terminal output), it checks whether enough space is available in the local roll buffer (LRB). If there is enough space, the thread is copied to the LRB, and written asynchronously from the LRB to the roll file. If there is not enough space in the LRB, the thread is written directly to the roll file. The roll file directory in the CF structure is updated accordingly. Thread overflow is handled as described under *Roll Server in a Single z/OS System*.

When the Roll Server receives a read request for a thread (after terminal input), and the last write request was issued in the same z/OS image, the Roll Server copies the thread directly from the LRB into the requestor's address space. If the last write request did not come from the same z/OS image, the thread is read from the roll file and then copied into the requestor's address space.

Illustration of Roll Servers in a z/OS Parallel Sysplex Environment:



Roll File and LRB

The roll file is a `BDAM` file logically subdivided into a directory and fixed-length slots. The slot size is a parameter of the roll-file formatting routine `NATRSRFI`. Slots must be larger than the largest compressed Natural thread expected.

The roll file directory contains one entry for each active Natural session, together with a timestamp of its last write request. In a single z/OS system, the directory resides in the Roll Server's address space. In a z/OS Parallel Sysplex environment, it resides in the Coupling Facility. The directory is written back to the roll file only when the Roll Server terminates or de-allocates its resources. Refer to *Roll Server Operation*, `DEAL` and `TERM` commands.

The local roll buffer is contained in a data space or z/OS memory object and subdivided into fixed-length slots. LRB slots may be smaller than roll-file slots. When a thread is larger than the LRB slot size, it is written directly to the roll file. The number of LRB slots and their size are Roll Server startup parameters; they are important factors in system performance.

The Roll Server can run with up to five different roll files. Each of these roll files is logically connected to one local roll buffer. If there are five roll files, there are five corresponding LRBs. Each roll file is accessed by its own dedicated read, write, and staging tasks. Thus, if the roll files are created on different disks on different channels, the roll files can be accessed simultaneously.

Natural users are allocated to roll files according to the following algorithm:

```
RN := (first four bytes of roll-server-user-id interpreted as positive integer) modulo number of roll files + 1
ALLOCNUM := 0
FOR I = RN TO  number of allocated roll files
   IF ROLLFILE(I) is not full THEN
      ALLOCNUM := I
      ESCAPE BOTTOM
   END-IF
END-FOR
IF ALLOCNUM = 0 THEN
   FOR I  = 1 TO RN-1
      IF ROLLFILE(I) is not full
         ALLOCNUM := I
         ESCAPE BOTTOM
      END-IF
   END-FOR
END-IF
IF ALLOCNUM = 0 THEN
   Return 'roll file full' error
ELSE
   Allocate userid to roll file number ALLOCNUM
END-IF
```

where *roll-server-user-id* is a 16-byte, unique string provided by the Natural interface; for more information, see the corresponding TP monitor interface section in this documentation.

Example:

- There are five roll files and the *roll-server-user-id* is `UF`. Roll File 2 is full, but Roll File 3 has free slots available:

  ```
  E4C64040 - 80000000 = 64C64040
             64C64040 modulo 5 = 1
  ```

- Roll File 2 is the first file to be checked for a free slot. Since the check fails, Roll File 3 is tried next, and a free slot is found.

- User `UF` is therefore allocated to Roll File 3.

If this algorithm does not guarantee that your user IDs are evenly distributed among the roll files, the Roll Server's user exit `NATRSU14` will help. This is especially relevant in server environments (see *Natural as a Server under z/OS*), because the first eight bytes of the roll server user ID are filled with the name of the server. For more information on this user exit, see *Natural Roll Server Operation, NATRSU14 User Exit*.

To see how evenly your user IDs are distributed, display the Roll-Server statistics using the Natural command SYSTP, selection "R".