Natural

zIIP Enabler for Natural

Version 8.2.6

April 2017

ADABAS & NATURAL

Document ID: NATMF-ZIIP-826-20170418

Table of Contents

Preface

The zIIP Enabler for Natural provides support of IBM System z Integrated Information Processors (zIIPs) available in a Natural z/OS batch, batch server, TSO or Com-plete environment.

Prerequisites	Requirements for zIIP support by Natural.
General Information on zIIP Processing	Brief description of zIIP processing.
Natural zIIP Processing: TCBs, SRBs and Enclaves	Explanations of the TCB and SRB processes and the WLM enclaves Natural requires for zIIP processing.
Monitoring zIIP Usage	System information, reports and statistics available for controlling and evaluating zIIP-enabled Natural sessions.
Tuning zIIP Usage	Natural parameters, statements and operating system calls that can affect zIIP usage.

Note: The Natural profile parameters mentioned in this documentation are described in the *Parameter Reference* documentation, unless otherwise noted.

1 Prerequisites

Prerequisites for Installation

zIIP support by Natural requires that zIIP Enabler for Natural (including an extra product license) is installed at your site.

All prerequisites for installation are described in *Installing zIIP Enabler for Natural* in the *Installation for z/OS* documentation.

Requirements and Restrictions for Com-plete and CICS

The Natural Com-plete/SMARTS Interface or Natural CICS Interface version installed at your site must support zIIP Enabler for Natural and the prerequisites for installation mentioned earlier must be fulfilled.

The requirements and restrictions described in this section depend on the version of the Natural Com-plete/SMARTS Interface or Natural CICS Interface installed.

The requirements and restrictions that apply when using zIIP Enabler for Natural under Com-plete help optimize runtime performance and reduce the number of SRB/TCB switches to a minimum:

- The Natural startup application must be cataloged with the Com-plete ULIB attributes PV (privileged) and AF (OS task affinity) specified. For more information, see the appropriate *Com-plete* documentation.
- The number of threads and tasks for zIIP processing should be increased with the Com-plete startup parameters THREAD-GROUP and TASK-GROUP to prevent Com-plete from dispatching bottlenecks. For more information, see the appropriate *Com-plete* documentation.

  Specify the same number of threads and tasks for each group.

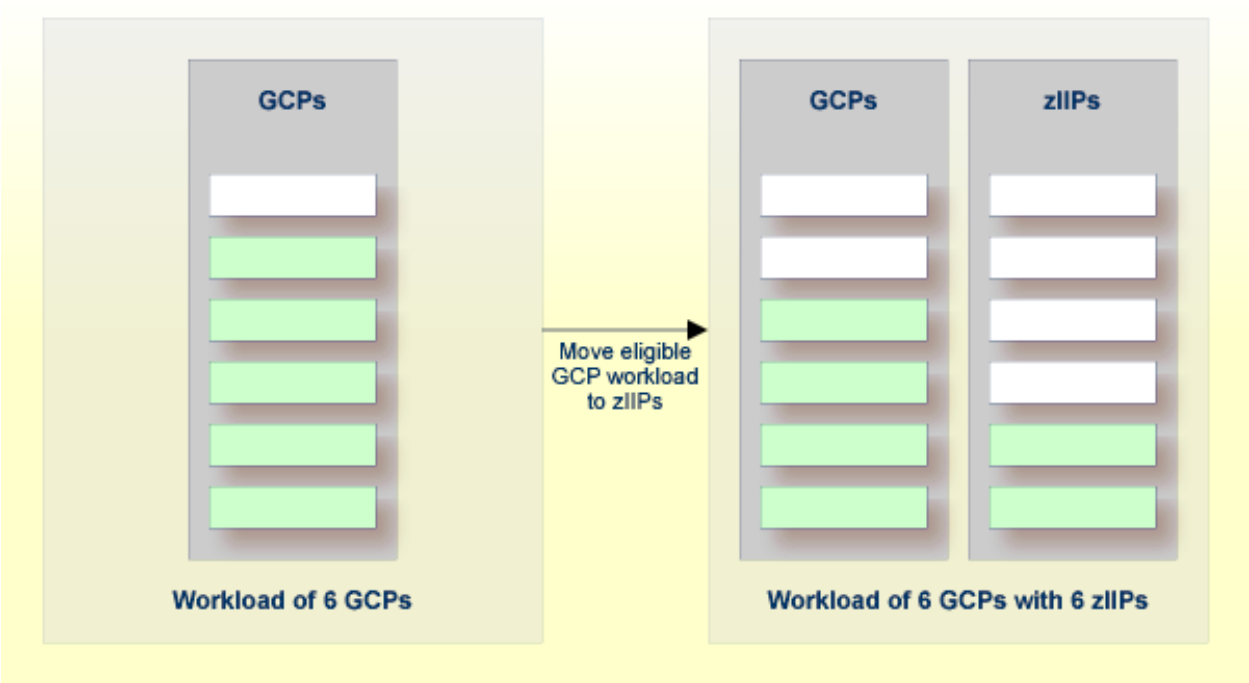
2 General Information on zIIP Processing

The IBM System z Integrated Information Processor (zIIP) is a speciality engine designed to offload eligible database workload from a GCP (general central processor) to a zIIP.

Offloading workload to a zIIP helps optimize resource capacities and expand the use of a GCP for new workloads, while lowering the mainframe TCO (total cost of ownership). In contrast to the expensive GCP which may even run throttled, the zIIP is inexpensive and always runs at full speed.

For detailed information on the zIIP, refer to the appropriate IBM literature.

The simple graphic below illustrates the purpose of the zIIP:



3 Natural zIIP Processing: TCBs, SRBs and Enclaves

This chapter provides information on how Natural enables zIIP support.

In general, all z/OS applications (including Natural applications) run as a TCB (task control block) process (TCB mode). However, running applications on a zIIP requires an SRB (service request block) process which must be assigned to an IBM WLM (z/OS Workload Manager) enclave. This enclave is a special WLM transaction unit that organizes several TCB and/or SRB processes which run in the same or in different regions. The WLM enclave created and used by Natural for zIIP processing is bound to a region, this means, the WLM enclave terminates when the corresponding job step terminates.

TCB/SRB Switches

The prerequisites for zIIP enablement (described in *Installing zIIP Enabler for Natural*) are checked during the initialization of a Natural session. If successful, an SRB process is started for the Natural session in parallel to the current TCB process which is placed into a wait state while the SRB continues processing the Natural session. The TCB wait state is revoked when a service call - supervisor call (SVC) or program call (PC) - is to be issued, because these calls cannot be processed in SRB mode. The TCB then continues processing while the SRB is placed into a wait state. In the Natural documentation, this procedure is called “SRB/TCB switch”. When the service call has finished, the wait state of the SRB is revoked and, again, the TCB is placed into a wait state, and so on. Consequently, this is called “TCB/SRB switch”. The zIIP processing reports produced by the Natural system command `ZIIP` (see ***zIIP Processing Reports Available***) only provides information on the TCB/SRB switches. This is because the number of SRB/TCB switches is usually exactly the same as the number of TCB/SRB switches. Exception: If an abnormal termination occurs in SRB mode, z/OS automatically terminates the SRB and the TCB wait state is revoked to handle the abnormal termination. When the session continues, Natural starts a new SRB process upon the next TCB/SRB switch.

Restrictions and Limitations in SRB Mode

There are a number of restrictions and limitations for SRB processing; for example, the SRB cannot execute normal system service calls such as SVCs or PCs. These calls can only be used in TCB mode. Moreover, in SRB mode, the TCB address cannot be loaded from low core storage (in the field PSATOLD) as some applications attempt in order to access the TCB storage and other important control blocks. In SRB mode, such load techniques most likely result in an S0C4 protection exceptions because PSATOLD is zero.

4 Monitoring zIIP Usage

This chapter provides information on how to view and control data on zIIP processing.

See also ***Monitoring the Cache Usage*** in the section *Print and Work File Caching*.

zIIP Processing Reports Available

You can use the Natural system command `ZIIP` to determine whether the current Natural session is running on a zIIP and analyze your CPU time savings by zIIP enablement.

The `ZIIP` command displays the number of GCPs and zIIPs available in the your z/OS environment, the CPU time consumed and, optionally, a list of components that caused SRB/TCB switches: see the example reports and explanations in *zIIP Processing Information* and *zIIP Component Switch Statistics* in the *System Commands* documentation.

For batch processing, you can also set the keyword subparameter `PRINT` of the profile parameter `ZIIP` to print zIIP processing information automatically at the end of the session.

You can also use the `ZIIP` command to display and check the `NAZ`*`vrs`*`.LICS` license file installed at your site.

For a detailed description of the `ZIIP` command, refer to the *System Commands* documentation.

Determining the Number of zIIPs Available

If you do not know how many zIIPs are available in your z/OS LPARs (logical partitions), you can check this with the following operator command:

```
D M=CPU
```

The following example result of this command shows that four GCPs (`00` to `03`) and one zIIP (`+I`) is running in the current LPAR, and that one additional GCP (`05`) and one additional zIIP (`-I`) are defined as spare processors.

```
D M=CPU
IEE174I 12.39.09 DISPLAY M 781
PROCESSOR STATUS
ID CPU                   SERIAL
00 +                      0FA10E2098
01 +                      0FA10E2098
02 +                      0FA10E2098
03 +                      0FA10E2098
04 +I                     0FA10E2098
05 -
06 -I
```

Using SDSF ENC to Watch zIIP Usage

If you have the z/OS SDSF (System Display and Search Facility) installed under TSO, you can use the command ENC to obtain information about all WLM enclaves currently active in your z/OS environment and their zIIP usage.

For detailed information, see the IBM literature *SDSF Operation and Customization*.

Calculating the CPU Time

Be aware that in the SDSF "D A" display the CPU time does not show the real values if running in a WLM enclave. You need to use the value of "ECPU Time" to evaluate the right CPU usage of your zIIP-enabled Natural session. The following definitions are quoted from the IBM z/OS SDSF documentation:

- "CPU Time" is the CPU time (TCB + SRB) for the address space, *excluding* any NP-SRB/enclave time.
- "ECPU Time" is the CPU time (TCB + SRB) for the address space, *plus* any NP-SRB/enclave time; that is, CPU used on behalf of this address space - the SDSF definition of "CPU Time".

Note: Com-plete `UQ A` and Natural ISPF only document the "CPU Time", there is no "ECPU Time" value available.

Evaluating the z/OS SMF Type 30 Records

You can evaluate the SMF type 30 records to obtain statistics about zIIP processing per address space. These records contain several fields with various job step CPU time values such as TCB, SRB, WLM enclave and zIIP times.

For detailed information, see the IBM literature *z/OS MVS System Management Facilities (SMF)*.

5 Tuning zIIP Usage

This chapter contains suggestions to avoid unnecessary SRB/TCB switches and reduce CPU overhead for zIIP support and to improve the offload to the ZIIP.

Note: You can use the **Component Switch Statistics** (see the Natural system command `ZIIP`) to view the list of components that cause TCB switches.

CPU Time Limit - FETCH Operations

The Natural profile/session parameter `MT` (maximum CPU time) has the effect that each time Natural starts a program at Level 1, a z/OS timer service request must be executed. This forces Natural to switch off from the zIIP. The default setting is `MT=60` to prevent endless loops in Natural applications. Consequently, each time Natural falls back to a program level of 0, the timer is restarted. This also happens with the `FETCH` statement: each time a program is fetched, the timer must be reset.

The default setting is `MT=60` to prevent endless loops in Natural applications. Software AG recommends to set `MT=0` when running a session that executes many `FETCH` statements. This will prevent Natural from using timer macros and avoids unnecessary SRB/TCB switches.

3GL Program Calls

Natural needs to switch off zIIP usage each time a 3GL program is executed, because Natural does not know whether any z/OS service calls are issued in the external subprogram(s).

In addition, Natural is only allowed to offload Natural-written code to a zIIP. User-defined code written in any other language must not be offloaded.

Calling 3GL programs within a Natural loop, forces many SRB/TCB switches, and therefore much CPU overhead. A high share of 3GL code will reduce the offload capabilities of Natural sessions. Such sessions are not suitable for running zIIP-enabled.

Natural Multi-Fetch Record Retrieval

Executing an Adabas access statement in Natural, causes at least one WAIT SVC call and forces Natural to switch off from the zIIP. You can reduce the number of switches by exploiting the multi-fetch capabilities of Natural:

	Without Multi-Fetch	With Multi-Fetch
Records read	100,000	100,000
Number of switches into SRB mode	100,000	12,500
Total enclave CPU time	5,389 ms	958 ms

For further information on the multi-fetch feature, see *Multi-Fetch Clause*, in *Accessing Data in an Adabas Database* in the *Programming Guide*.

Sort Processing - SORT Parameter

External sorts are not processed on the zIIP. An external sort would require several database SRB/TCB switches for `SORTIN` and `SORTOUT` processing (mostly four switches per record), thus significantly slowing down your session when running zIIP-enabled.

Using the Natural `SORT` reduces the number of SRB/TCB switches to one switch.

	Using External Sort	Using Natural SORT
Records read	1,145	1,145
Number of switches into SRB mode	3,435	1,148
Total enclave CPU time	155 ms	70 ms

SORT Parameter

You can specify the Natural profile parameter `SORT` to bypass the external sort and use the internal Natural `SORT`, for example:

```
SORT=(WRKSIZE=1000,EXT=OFF)
```

Print and Work File Caching

Any work file or printer usage causes I/O interrupts which force Natural to switch off from the zIIP. You can avoid unnecessary switches by defining cache buffers to be used for print and work file I/O processing. These cache buffers are used to keep the data in core as long as possible and read or write data in larger chunks.

The cache buffers are defined with the keyword subparameter `PCWSIZE` of the profile parameter `ZIIP`, for example:

```
ZIIP=(PCWSIZE=(300,200,300))
```

The buffer sizes are interpreted in KB. They specify the print buffer, read buffer and write buffer. Buffer sizes of several 100 KB are sufficient.

Rerouting Primary I/Os to a Cache Buffer

Primary I/Os are not subject to caching. Instead, the size of the terminal I/O buffer is relevant. This buffer is flushed either when it is full or when an I/O is triggered by an `INPUT` statement.

You can use the Natural profile parameter `MAINPR` to separate program output from Natural system output and reroute primary output for `CMPRINT` to an additional printer that is processed with a cache buffer.

Example of Cache Buffer Usage

When your application creates a printout with a line size of 132 characters, a print cache of 132 KB will reduce the SRB/TCB switches for printing by a factor of 1000. This means, Natural will not switch for every line, but only once per 1000 lines or 20 pages.

	WRITE (1) Without Cache	WRITE (1) With Cache
Lines written	10,000	10,000
Number of switches into SRB mode	10,003	10
Total WLM enclave CPU time	386 ms	68 ms

For the work file handling, you will get a similar result:

	READ WORK FILE (1) Without Cache	READ WORK FILE (1) With Cache
Records read	10,000	10,000
Number of switches into SRB mode	10,478	5
Total WLM enclave CPU time	641 ms	305 ms

Monitoring the Cache Usage

You can monitor the usage of the cache buffers by using the Natural system command `BUS`.

To monitor buffer cache usage

- Issue the following system command:

```
BUS
```

A **Buffer Usage Statistics** report then outputs the sizes used by the cache buffers:

```
12:50:28               ***** NATURAL BUS UTILITY *****              2012-04-03

User SAG                 - Buffer Usage Statistics -            OpSYS z/OS

 No. Name    Type    Size      Used   Perc.  MaxUsed   Perc.  MaxSize   Perc.
-------------------------------------------------------------------------------
 18  PCACHE    V   512000        32    0.0    511958  100.0
 21  WCACHE0   V   307200        32    0.0    306947   99.9
 26  WCACHE01  V   512000        32    0.0    511992  100.0

-------------------------------------------------------------------------------
ThrdSize   Total  1382656    166593   12.0      2000   54.9    33480      1.6
  2000K  (in KB)   1351K      163K             742K             33K
-------------------------------------------------------------------------------

Nat9995 Natural session terminated normally.
```

- `PCACHE` is the cache for all print output.
- `WCACHE0` is the cache for the `WRITE WORK FILE` statement.
- `WCACHE`*nn* is the cache for the `READ WORK FILE` statements.

For statements generating output (such as `WRITE` or `WRITE WORK FILE`), only one cache buffer is allocated, also for multiple files. For the `READ WORK FILE` statement, there is one buffer allocated per work file. The buffers are allocated only when used.

For detailed information on the **Buffer Usage Statistics**, refer to the relevant section in *SYSTP Utility* in the *Utilities* documentation.

Thread Size Specification - THSIZE Parameter

In batch mode and under TSO, Natural usually allocates an internal buffer with a `GETMAIN` or `FREEMAIN` requests to the operating system. Any storage request includes an SVC (supervisor call) and requires a switch back to TCB processing mode.

You can reduce the number of `GETMAIN` or `FREEMAIN` requests by specifying a thread size with the Natural profile parameter `THSIZE`. Natural will then allocate the specified amount of space with one `GETMAIN` and then serve all buffer requests from the allocated thread storage without calling the operating system again.

At the end of a Natural session, you can use the **Buffer Usage Statistics** report (see the **example screen**) to check the buffer and thread usage and determine whether the defined thread size (`MaxUsed`) is sufficient to allocate all buffers used by the session.