

Natural Storage Management

This section describes how Natural allocates and uses main storage. A piece of storage requested by a Natural nucleus component is called a "buffer".

The following topics are covered:

- Thread and Non-thread Environments
- Buffer Types
- Fixed Buffers
- Variable Buffers
- Customization of Buffer Characteristics

Configuring Natural - Other Topics:

Linking Natural Objects to the Natural Nucleus | Natural Application Programming Interfaces | Natural User Exits | Natural User Access Method for Print and Work Files | Natural Scratch-Pad File | Natural Text Modules | Natural Configuration Tables

Thread and Non-thread Environments

There are two different types of storage environments:

- Thread storage environment (typical for multi-user environments, e.g. CICS)
- Non-thread storage environment (typical for single-user environments, e.g. batch)

In a thread environment, a big piece of storage called "thread" is pre-allocated for a session. The thread size must be predefined by the system administrator. During a session each buffer allocation request (getmain) is satisfied within its thread by Natural itself. Free space due to release buffer requests (freemain) can be reused.

Upon certain events (terminal I/Os and long waits), the thread storage may be compressed and rolled out (or swapped out) to external storage (swap pool or roll file). The released thread can be reused by other Natural sessions. When a suspended session is to be resumed, it is rolled in from external storage into a free thread again.

The place on the swap pool or roll file where the compressed thread storage is stored, is called a "slot". The slot size has a fixed length and is defined by the system administrator. It must be large enough to contain the largest compressed thread storage. In the worst case, it may be equal to the thread size.

In a non-thread environment, all storage requests are directly passed to the operating (sub-)system. No roll-out/roll-in is performed, that is, the buffers for a session are kept until session termination, unless they were explicitly released before.

Buffer Types

There are 3 different types of buffers:

- fixed buffers
- variable buffers
- physical buffers

Fixed and **variable buffers** have a 32-byte prefix with a common layout for all environments. The buffer prefix starts with the buffer name followed by 5 buffer length fields (total, used low-end, max. used, used high-end, max. used high-end). The used length fields are maintained by the buffer-owning components and are used for thread compression. Each buffer has a unique ID number (1-255) and can exist only once. Some buffers are allocated

during session initialization, others are allocated when required. The `BUS` command can be used to show information about all fixed and variable buffers currently allocated. The characteristics of the buffers are defined in the source module `NATCONFIG`, which can be customized in exceptional cases (see [Customization of Buffer Characteristics](#) below). The size of some buffers can be specified by a profile parameter. For a complete list of such buffers, see the profile parameter `DS`.

Physical buffers are allocated outside the thread. They do not have a buffer prefix and they are not unique. They are used in exceptional cases and temporarily only. Physical buffers are automatically released at the next terminal I/O. It is possible to define work pools for physical buffers by profile parameter `WPSIZE`.

Fixed Buffers

In a thread environment, fixed buffers are allocated from the low end of the thread only. In contrast to variable buffers, fixed buffers cannot be moved relatively to the thread and their size cannot be increased or decreased.

Variable Buffers

In a thread environment, variable buffers are allocated from the high end of the thread. If there is no more space in the thread, variable buffers are allocated temporarily outside of the thread. Upon thread compression, all buffer parts used are compressed into the thread. If they do not fit into the thread, the session is terminated abnormally. This may happen especially when large dynamic variables are used.

After thread decompression, the variable buffers may have been moved to a different place inside or outside of the thread. Variable buffers can be increased or decreased in size on request by the owning component. Some variable buffers are defined to be reduced or released automatically during thread compression.

The total amount of storage allocated outside the thread can be limited by profile parameter `OVSIZE`.

Customization of Buffer Characteristics

All buffers are defined in the source module `NATCONFIG` by the macro `NTBUFID`.



Please, do not change any buffer characteristics except the minimum and maximum buffer size settings explained below, because the results may be unpredictable.

It is possible to change the buffer size limits by the parameters `MIN` and `MAX` of the macro `NTBUFID`. This makes sense for variable buffers (`TYPE=VAR`) only. There are limits for all buffers, either by default (0 - 2097151 KB) or by the limits of the corresponding profile parameters, the see profile parameter `DS`. The limits of the buffer size profile parameters in the Natural parameter module (`NATPARAM`) are not affected by the `MIN` and `MAX` parameters of `NTBUFID`, but the limits for the dynamic profile buffer size parameters are overwritten by `MIN` and `MAX`.

Setting the `MAX` parameter to a value in KB means that the size of this buffer cannot exceed this maximum during session execution. This may cause runtime errors if more buffer storage is requested for the desired buffer.

Setting the `MIN` parameter to a value in KB means that the size of this buffer cannot be less than this value during session execution. For example, in the case of the 3GL `CALLNAT` interface (`NAT3GCAN`), the setting of a buffer minimum value makes sense for the following buffers, because the sizes of these buffers may not be increased on a lower Natural program level called by a 3GL program.

DATSIZE	Data areas
GLBTOOL	Utility GDA
GLBUSER	User GDA
GLBSYS	System GDA
AIVDAT	AIV area
CONTEXT	Context variables