

Natural Buffer Pool

The buffer pool is a storage area into which Natural programs are placed in preparation for their execution. Programs are moved into and out of the buffer pool as Natural users request Natural objects. Conceptually, it serves a function similar to that of an operating system in loading programs in and out of a reentrant area. The Natural buffer pool is an integral part of Natural in all supported environments.

The following topics are covered:

- Natural Buffer Pool Principle of Operation
 - Natural Turbo Performance Plug-In
 - Buffer-Pool Monitoring and Maintenance
 - Natural Global Buffer Pool
-

Natural Buffer Pool Principle of Operation

Natural generates reentrant Natural object code. A compiled program is loaded into the buffer pool and executed from the buffer pool. Thus, it is possible that a single copy of a Natural program can be executed by more than one user at the same time.

This section covers the following topics:

- Objects in the Buffer Pool
- Directory Entries
- Text Pool
- Buffer Pool Initialization
- Local and Global Buffer Pools

Objects in the Buffer Pool

Objects in the buffer pool can be programs, subprograms, maps and global data areas. Global data areas are placed in the buffer pool only for compilation. In this case, two objects with the same name are loaded in the buffer pool: the GDA itself and the corresponding symbol table.

Directory Entries

When a Natural object is loaded into the buffer pool, a control block called a directory entry is allocated to this object.

A directory entry contains such information as the name of the object, what library it belongs to, what database ID and Natural system file number the object was retrieved from, and some statistical information (for example, the number of users who are concurrently executing the program at a given point in time).

When a user executes a program, Natural checks the directory entries to see if the program has already been loaded into the buffer pool. If it is not already in the buffer pool, a copy of the program is retrieved from the appropriate Natural system file and loaded into the buffer pool.

When an object is loaded in the buffer pool, one or more other Natural objects which are currently not being executed may be deleted from the buffer pool in order to make room for the newly loaded object. When the new object is loaded, a new directory entry is created in order to identify this object.

When an object is deleted from the system file, it will also be deleted from the buffer pool as soon as it is no longer being used. When an object is newly cataloged or stowed, its old version will also be deleted from the buffer pool as soon as it is no longer being used; when it is requested for execution again, the new version will then be loaded from the system file into the buffer pool.

Text Pool

The actual object code of a program that is loaded into the buffer pool is placed into an area called the text pool and must be allocated as a contiguous piece of memory within this text pool. This text pool is divided into a number of 4 KB buffers. This is an arbitrary size and can be changed at the Natural administrator's discretion. When an object is loaded, one or more text buffers that are contiguous to each other are allocated to store the object code of the object.

Buffer Pool Initialization

The initialization time of the buffer pool varies depending on the environment. In batch mode, for example, initialization occurs when you begin the execution of the batch Natural nucleus.

Initialization in a TP monitor generally occurs when the TP monitor is started or, in some cases, when the first user invokes Natural under this TP monitor.

Local and Global Buffer Pools

Local Buffer Pool

Using Natural as supplied on the installation tape, you are running a *local* buffer pool. This is a buffer pool area that is allocated in the same partition (or region or address space) of the particular environment in use.

For example, in a batch TSO or CMS environment, each user has his/her own local buffer pool. In a TP monitor environment such as Com-plete, CICS or IMS/TM, there is one buffer pool per TP monitor from which all TP users execute.

Global Buffer Pool

In an OS/390 environment, a global buffer pool is allocated from CSA or ECSA storage. In such an environment, all TSO users, batch users and TP monitor users could be executing from one common global area.

In a VSE/ESA environment, a global buffer pool is allocated from System GETVIS Area (below or above). In such an environment, all batch users and TP monitor users could be executing from one common global area.

In a VM/CMS environment, a global buffer pool can be installed as a writeable Discontiguous Shared Segment that is dynamically attached to the user's virtual machine when Natural/CMS is invoked; see also the sections Installing Natural under CMS and Preparing the VM System for Natural (in the Natural Installation Guide for Mainframes).

In a BS2000/OSD environment, a global buffer pool is a common memory pool, see Natural Global Buffer Pool under BS2000/OSD.

For further information on the global buffer pool, see Natural Global Buffer Pool.

Natural Turbo Performance Plug-In

The following topics are covered:

- Benefits of the Turbo Performance Plug-In
- Enabling the Turbo Performance Plug-In
- Hash Table Available with Performance Plug-In
- Buffer Pool Cache Available with Performance Plug-In

Benefits of the Turbo Performance Plug-In

The Natural turbo performance plug-in is generally available for the operating systems OS/390, VSE/ESA, BS2000/OSD and VM/ESA (except the buffer pool cache for VM/ESA).

Significant performance improvements are provided by the following features:

- a new buffer-pool search algorithm reduces the time required to search for an object in the buffer pool,
- a buffer-pool cache reduces system file access,
- using the Adabas multi-fetch option speeds up the process of loading objects into the buffer pool.

Enabling the Turbo Performance Plug-In

The performance plug-in must be explicitly enabled to become functional. This is done with the `PLUGIN` parameter.

Hash Table Available with Performance Plug-In

This section applies to buffer pools of `TYPE=NAT` and is applicable only in conjunction with the Natural Performance plug-in (parameter `PLUGIN=BP`).

To speed up the search time for looking up an object in the buffer pool directory, an optional hash table is used. The number of entries in the hash table is twice the number of directory entries, rounded up to the next prime number. This will ensure that only half of the table is filled at any point in time and that the probability of collisions is near zero. As a consequence, the average number of tests to find an existing object in the hash table is theoretically less than 2.

The hash criterion is the eight character long program name. If more than one program name is hashed to the same location in the hash table, an overflow technique resolves the collisions.

The storage required for the hash table is roughly 16 bytes per text block. Thus, the available storage in the text pool is reduced by between 1.6% (1 KB text blocks) and 0.1% (16 KB text blocks) compared to a buffer pool without hash table.

The use of the hash table is optional and is available only in conjunction with the enhanced buffer pool manager. It can be turned on and off during buffer pool initialization with the parameter `PLUGIN`.

- In the case of a global buffer pool, the `PLUGIN` setting specified with the startup parameters when starting the global buffer pool is used.
- In the case of a local buffer pool, the `PLUGIN` profile parameter setting of the first Natural session which connects to this buffer pool is used.
- In the case of the `INITIALIZE` command of the `SYSBPM/SYBPMT` utility, the `PLUGIN` setting of the current Natural session is used.

As a buffer pool with a hash table has an incompatible layout compared to a buffer pool without a hash table, only Natural sessions which use a compatible `PLUGIN` setting can access a buffer pool with a hash table.

The rules for the Natural sessions are summarized in the following chart:

PLUGIN=NOBP	The use of the hash table is disabled and the buffer pool manager works in the same way as in the standard Natural version.
PLUGIN=OFF	If a Natural session tries to access a buffer pool initialized with PLUGIN=BP/ON, the access is rejected with error message 1071. If a Natural session has to initialize the buffer pool, this buffer pool is initialized without the hash table.
PLUGIN=BP	The use of the hash table is enabled.
PLUGIN=ON	If a Natural session accesses a buffer pool initialized with PLUGIN=BP/ON, the fast access via the hash table is used. If a Natural session accesses a buffer pool initialized with PLUGIN=NOBP/OFF, this Natural session works in the same way as in the standard Natural version. If a Natural session has to initialize the buffer pool, this buffer pool is initialized with a hash table.

Problem Handling

In case of problems, the following fallback strategies using SYSBPM are available:

- **CLOSE HASH**
This command disables the hash table support and the enhanced buffer pool manager behaves in the same way as the default (earlier version) buffer pool manager. Nevertheless, it can be accessed only by sessions with PLUGIN=BP parameter setting.
- **INITIALIZE OLD**
This command reinitializes the whole buffer pool without hash table support. This means, that each Natural session can access the buffer pool, regardless of its PLUGIN parameter setting.

In order to be always able to execute these commands, you are recommended to have an additional buffer pool initialized with PLUGIN=NOBP. This enables you to use SYSBPM from a Natural session connected to this additional buffer pool and to issue commands to the affected buffer pool.

Buffer Pool Cache Available with Performance Plug-In

This section applies to buffer pools of TYPE=NAT and is available only in conjunction with the Natural Performance plug-in (parameter PLUGIN=BP). The buffer pool cache is available in conjunction with a global buffer pool only. It is not available with VM/ESA. The use of a buffer cache implies the use of a hash table.

The buffer pool cache option is only available in conjunction with a global buffer pool. It is used only for Natural programming objects (programs, subprograms, maps, etc.), whereas it is not used for example for objects generated by Natural for DL/I.

When a buffer pool is not large enough to take up all objects requested by the different users, special overload strategies are used to replace existing objects with requested objects. The number of overload situations has a direct relation to the efficiency of the buffer pool. The best and most efficient way to reduce the disliked overloads, hence to improve the buffer pool performance, is simply to increase its size.

However, this choice is not applicable at most customer sites, due to a lack of available storage in the primary address space and/or the OS/390 (E)CSA, VSE/ESA system GETVIS area or BS2000/OSD Common Memory Pool.

In order to improve the situation described above, a buffer pool cache is used. The main purpose of this mechanism is to prevent a loss of all objects which were deleted from the buffer pool due to "short-on-buffer-pool-storage" situations. This means, that an object delete results in a "swap out to buffer pool cache". The intended benefit of this feature is a reduction of database calls used for object loading and consequently a performance improvement.

The buffer pool cache area is allocated in a data space. When a data space is created for a buffer pool (BPCSIZE parameter specified for OS/390 or VSE/ESA, or DATA parameter specified for BS2000/OSD), the ownership is assigned to the creator task. If this task terminates, the system automatically deletes the data space. Therefore, the

creator task will stay alive in this case, regardless of whether RESIDENT=Y has been set or not.

Buffer-Pool Monitoring and Maintenance

The Natural utility SYSBPM (described in the section Debugging and Monitoring) provides statistical information on the current status of the buffer pool. SYSBPM also allows you to adjust the buffer pool to your requirements.

Preload List

A preload list is a list of objects that will be loaded into the buffer pool and remain there as resident. When a user requests such an object for execution, it is always already in the buffer pool and need not be loaded from the system file.

This may improve performance, may avoid buffer pool fragmentation, or may be useful to ensure that central error transactions are always available, even if the database containing the system file is not active.

At the start of the Natural session, Natural checks which of the objects on the preload list are already in the buffer pool. Those which are not will then be loaded from the system file into the buffer pool. This checking and loading is also performed whenever the buffer pool is connected, re-connected and refreshed.

A preload list is identified by name, and is attached to a specific buffer pool by specifying its name as startup parameter (for a global buffer pool) or in the NTBPI macro (for a local buffer pool). Thus, a different preload list may be defined for each buffer pool; or the same preload list may be used for different buffer pools.

If the specified preload list cannot be located, or if objects contained on the preload list cannot be loaded, Natural will issue a corresponding warning message at session initialization. In either case, the preloading will be repeated for each subsequent session initialization.

As the objects on the preload list are the first to be loaded, they are located at the beginning of the buffer pool (except if the initial preloading could not load all objects, in which case the objects may be located anywhere in the buffer pool).

To maintain preload lists, you use the SYSBPM utility (described in the section Debugging and Monitoring).

Blacklist

To prevent a Natural object from being executed, you can put it on a so-called "blacklist": the object can then not be loaded into the buffer pool; and if it is already in the buffer pool, it cannot be executed. A user requesting such an object to be executed will then receive an appropriate error message.

You can put not only individual objects on the blacklist, but also entire libraries.

Examples:

- The blacklist may be useful, when you upgrade a Natural application and do not wish users to continue to work with that application until you have finished the upgrade.
Without the blacklist, a user might execute a new module which in turn would invoke an old module - which might lead to an abnormal termination of the Natural session.
With the blacklist, the user can will receive a message that the requested object can currently not be executed, and can then continue his/her Natural session.
- Performance aspects may be another reason for using the blacklist to prevent certain resource-consuming objects from being executed in a specific environment.
- You may use the blacklist to prevent the execution of test programs in a production environment.

To maintain the blacklist, you use the SYSBPM utility (described in the section Debugging and Monitoring).

Propagation of Buffer-Pool Changes

Note: Under OS/390, the propagation of buffer-pool changes is always restricted to the Natural subsystem in which the change has occurred (for details on the Natural subsystem, see Natural Subsystem (OS/390) or Natural Subsystem (VSE/ESA). Thus, "all global buffer pools" in this context means "all global buffer pools within the same subsystem".

In some environments, it is important that changes which occur in one (local or global) buffer pool are also propagated to all other global buffer pools - that is, the same changes are also automatically made in the other global buffer pool - so as to ensure the consistency of the buffer pools and the Natural applications being used. This is particularly important in a Sysplex environment (Sysplex in this context means the parallel Sysplex that was introduced by MVS/ESA SP5.1).

For example, if a Natural program is newly cataloged in one OS/390 image, the propagation will cause the program to be deleted from all other global buffer pools in the Sysplex, so that its new version has to be loaded from the system file when the program is to be executed again.

The following changes are propagated:

- an object is deleted from the buffer pool,
- the buffer pool's blacklist is modified,
- the buffer pool is re-initialized.

Changes can be propagated to all other global buffer pools within the current OS/390 image, or within the entire Sysplex, or all other global buffer pools of the same name within the Sysplex.

The propagation does not affect those objects in another global buffer pool which are defined as resident in that buffer pool.

The propagation is activated and its scope controlled by the Natural profile parameter BPPROP.

Attention: As the propagation is performed asynchronously and an object is only deleted from a buffer pool when it is no longer being used, it may take some time until the current version of an object is available in all buffer pools.

Propagation to other **local** buffer pools is not possible.

Natural Global Buffer Pool

The Natural global buffer pool is an optional Natural component.

It is available for the following operating systems

- OS/390 (Global Buffer Pool under OS/390)
- VSE/ESA (Global Buffer Pool under VSE/ESA)
- BS2000/OSD (Global Buffer Pool under BS2000/OSD).

Profile Parameters Used

The following Natural profile parameters are used to establish a global buffer pool:

BPNAME	Specifies the name of the global buffer pool (see BPNAME). BPNAME=' ' (blank) is used to establish a connection to the <i>local</i> buffer pool.
SUBSID	Specifies the ID of the Natural subsystem to be used (see SUBSID; applies only under OS/390 and VSE/ESA).

During Natural startup, Natural attempts to locate the global buffer pool using these parameters.

Buffer Pool Opening / Closing Procedure

With the NTBPI macro of the Natural parameter module or the corresponding profile parameter BPI, you can define more than one buffer pool.

At session initialization, Natural attempts to establish a connection to the first buffer pool defined. If this fails, Natural attempts to establish a connection to the second buffer pool defined. If that fails, too, it tries the next buffer pool defined, etc. Whenever an attempt to establish a connection to a buffer pool fails, Natural will issue a corresponding message.

The same procedure applies when a buffer pool is stopped: if you close the currently connected buffer pool while a Natural session is still active, Natural attempts to connect to another buffer pool (in the order in which they are defined) and continue the session. Thus, it is possible for the Natural administrator to close a global buffer pool without having to terminate all active Natural sessions.