Natural Nucleus

The Natural nucleus is the program that comprises the kernel of Natural. The nucleus runs on all mainframe operating systems, such as z/OS, z/VSE and BS2000/OSD, and under all TP monitors such as Com-plete, CICS and *open*UTM.

The Natural nucleus supplies a Natural user (for example, a developer or administrator) with all Natural functions such as command interpretation, object compilation and object execution. The Natural nucleus can be installed as a shared program that is used simultaneously by multiple users.

This section describes the main components of the Natural nucleus.

- Natural Runtime System
- Natural Compiler
- Natural Command Interpreter
- Configuration Settings

Natural Runtime System

The Natural runtime system can be considered a virtual machine that provides the environment necessary for executing Natural objects within an application. The Natural runtime system interprets and executes Natural internal object code (binary metacode).

The section below contains information on:

- Object Execution
- Object Starter and Object Executor

Object Execution

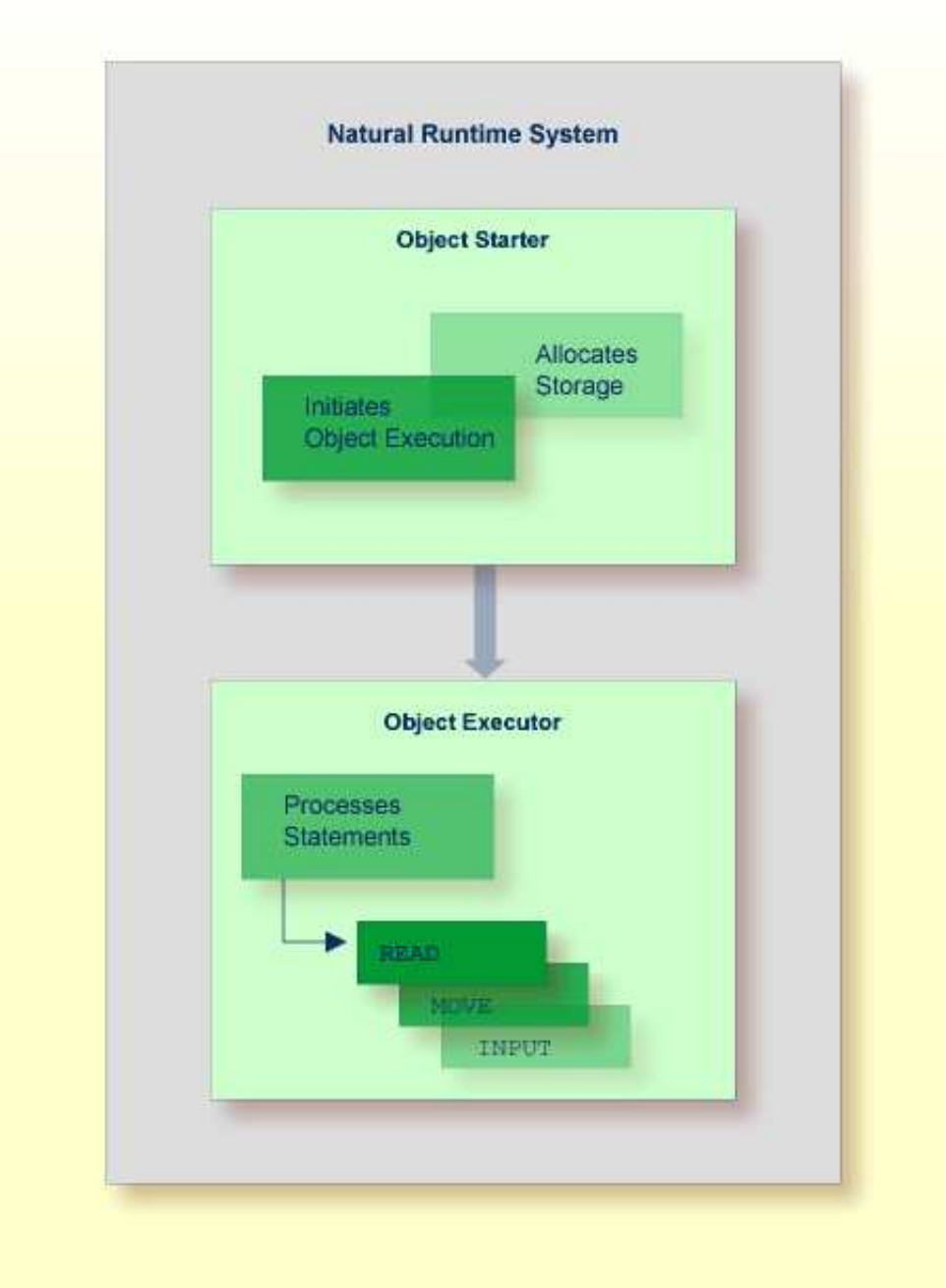
Internal Natural object code is executed when a Natural object is requested for execution.

Object execution is requested either directly by a user, or indirectly if the object currently being executed issues a Natural statement that requests execution of another object. The Natural system command `EXECUTE`, for example, directly executes the user-written program specified with the command. The Natural `CALLNAT` statement specified in a Natural program, for example, requests execution of a subprogram.

The *Example of Object Loading* in the section *Natural Buffer Pool* illustrates the process flow when executing a Natural program.

Object Starter and Object Executor

The diagram below is a schematic illustration of object execution by the Natural runtime system:



The Natural runtime system consists of two major components: the object starter and the object executor.

The object starter locates the object to be executed in the Natural buffer pool and allocates storage for variable data processed by the object as user session data. Finally, the object starter passes control to the object executor.

The object executor interprets the Natural statements in the object and executes them one after the other. In the diagram above, for example, the object executor first processes the `READ` statement by calling the database and retrieving the records requested and then continues with the `MOVE` statement.

Related Topics:

- *User Session Data*
- *Natural Buffer Pool*
- *Example of Object Loading - Natural Buffer Pool*

Natural Compiler

The Natural compiler generates the executable form of Natural source code. Natural source code is human-readable programming code that consists of a sequence of Natural statements. (For information on Natural statements and programming advice, see the *Statements* documentation and the *Programming Guide*.)

The Natural compiler reads the source code from the source area. The source code in the source area is part of the user session data. Source code is read into the source area by using the Natural system command `READ` or `EDIT`. The compiler checks the syntax of the source code and, if successful, generates Natural internal object code that can be interpreted and executed by the Natural runtime system.

When using the appropriate Natural system command, source code can be either compiled and executed, or compiled and stored. The section below describes the types of object module available for storage and the Natural system commands used for object compilation and execution:

- Cataloged Object
- Source Object
- Commands for Compilation
- Example of Compilation

Cataloged Object

A cataloged object is the executable (compiled) form of a Natural object. It is created by the Natural compiler and is stored as an object module in a Natural system file. Compiling source code and creating a cataloged object is referred to as cataloging an object. A cataloged object is created by using the Natural system command `CATALOG` or `STOW`.

At execution time, the cataloged object is loaded into the Natural buffer pool and executed by the Natural runtime system. Natural objects can only be executed or reference one another if they have been stored as cataloged objects in a Natural system file.

A cataloged object cannot be modified or decompiled.

Source Object

A source object (or a saved object) contains the human-readable form of Natural source code. Source code is saved as a source object in a Natural system file by using the Natural system command `SAVE` or `STOW`.

To execute source code contained in a source object, you need to compile the source code in order to create generated object code that can be interpreted and executed by the Natural runtime system.

Commands for Compilation

Natural provides different system commands for compiling source code. Depending on the Natural system command used, compilation is combined with any of the following actions:

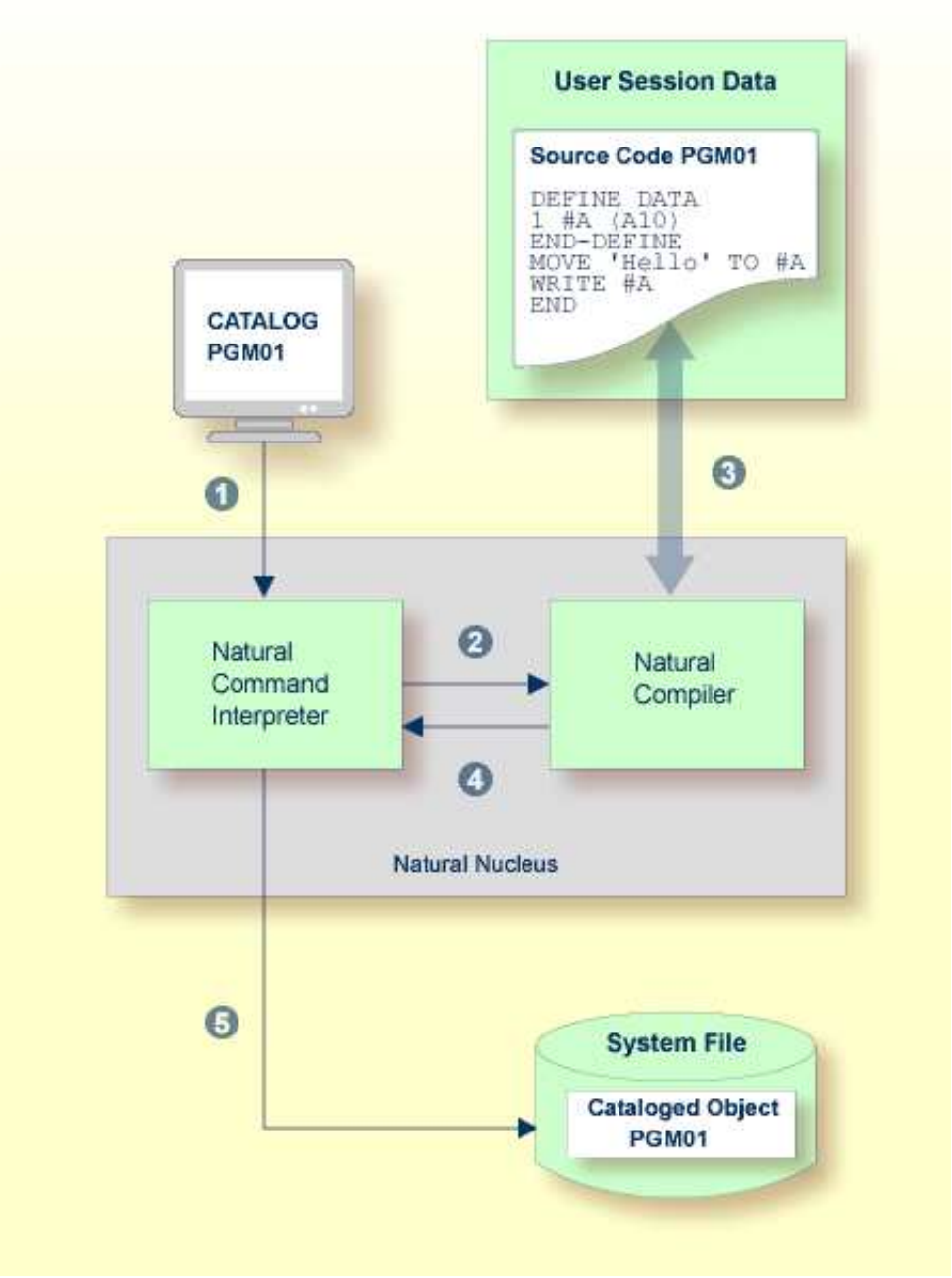
Action	System Command
Compile source code. Perform a syntax check and generate object code. The generated object code is not stored as an object module in a Natural system file.	CHECK
Compile source code. If successful, store the generated object code as a cataloged object in a Natural system file.	CATALOG
Compile source code. If successful, store the generated object code as a cataloged object in a Natural system file. Additionally, store the original source code as a separate source object in a Natural system file.	STOW
Compile source code of a Natural object of the type program. If successful, immediately execute the generated object code. The generated object code is not stored as an object module in a Natural system file.	RUN

Related Topics:

- *System Commands* documentation
- *Object Types - Programming Guide*

Example of Compilation

The diagram below illustrates the process flow when compiling source code with the Natural system command CATALOG:


User Session Data
Source Code PGM01
DEFINE DATA
1 #A (A10)
END-DEFINE
MOVE 'Hello' TO #A
WRITE #A
END
CATALOG
PGM01
1
2
3
4
5
Natural
Command
Interpreter
Natural
Compiler
Natural Nucleus
System File
Cataloged Object
PGM01

Legend

1. A user issues the Natural system command `CATALOG PGM01` requesting compilation of the source code currently contained in the source area and storage of the generated object code as a cataloged object with the name PGM01.
2. The Natural command interpreter interprets the command `CATALOG` and passes on the request to the Natural compiler.
3. The Natural compiler reads the Natural statements currently contained in the source area, checks the syntax and generates object code.
4. The Natural compiler returns control to the Natural command interpreter.
5. The Natural command interpreter stores the generated object code as a cataloged object with the name PGM01 in the current Natural system file.

Natural Command Interpreter

The Natural command interpreter checks and executes user command input from a Natural command prompt.

If Natural Security is installed, command execution can be restricted to a single user or a group of users.

Related Topics:

- *System Commands* documentation
- *Natural Security* documentation

Configuration Settings

Natural parameters manage the configuration of a Natural environment.

Natural parameters are used to standardize and automate development and production processes or adjust standard settings to the needs of individual users. A Natural parameter, for example, is used to set defaults for report creation, define the size of a report or define the size of storage area required such as the source area of an editor.

Most of the characteristics of a Natural environment are predefined by Software AG. The Natural administrator can configure different default environment settings valid for all Natural users. A user can adapt the settings to his needs by overriding default environment settings with a dynamic profile parameter or session parameter.

The section below contains information on:

- Profile Parameters
- Session Parameters
- Parameterization Levels

Profile Parameters

Profile parameters are specified statically or dynamically.

Static parameters are specified in the Natural parameter module NATPARM, during the installation of Natural. They are used as the default for each Natural session.

Dynamic parameters are specified at the startup of a Natural session. You can predefine a set of dynamic parameters with the Natural SYSPARM utility.

Related Topics:

- *Profile Parameters - Parameter Reference* documentation
- *Profile Parameter Usage - Operations* documentation
- *Using a Natural Parameter Module - Operations* documentation
- *SYSPARM Utility - Utilities* documentation

Session Parameters

Session parameters are specified within an active Natural session and/or within a Natural object. The main purpose of session parameters is to control the execution of Natural programs.

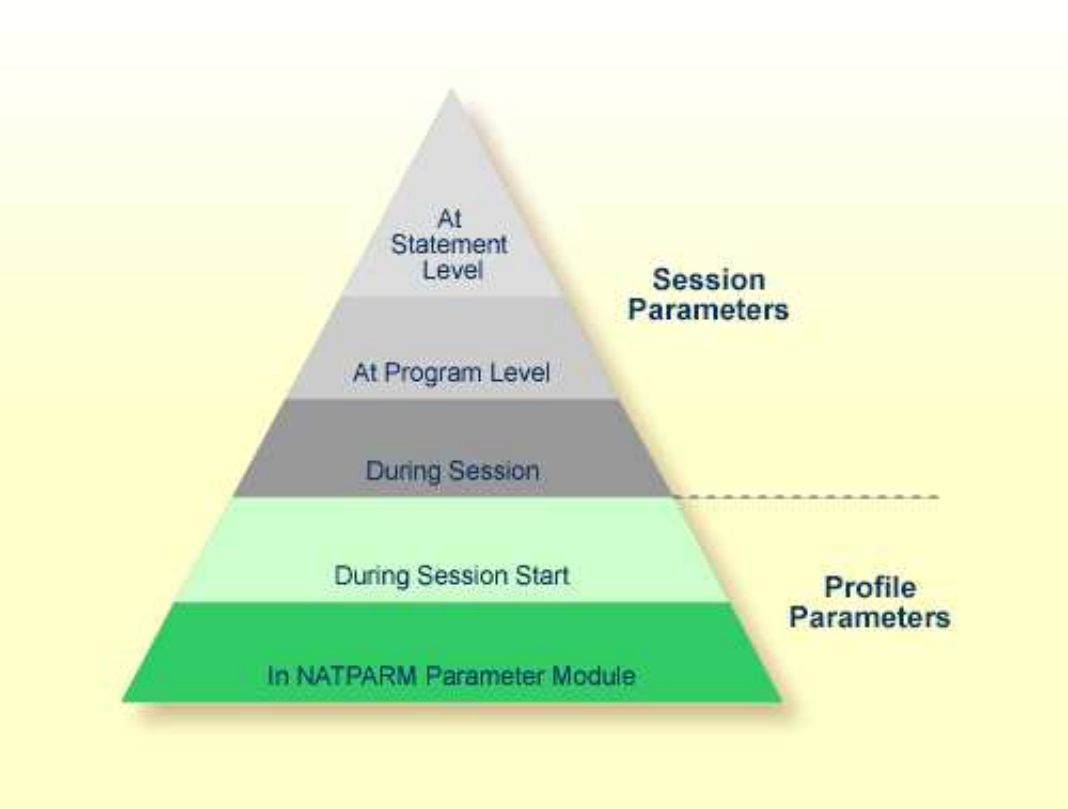
Related Topic:

- *Session Parameters - Parameter Reference* documentation

Parameterization Levels

There is a hierarchical structure of the levels at which Natural parameters can be set. A parameter value set on a higher level overrides the value defined on a lower level. For example, when you specify a parameter dynamically, the new parameter value overrides the static specification as set for the corresponding parameter in the Natural parameter module NATPARM.

The diagram below illustrates when a parameter can be set and the Natural parameter hierarchy from the lowest level at the base of the pyramid to the highest level at the apex:



Related Topic:

- *Natural Parameter Hierarchy - Operations* documentation