Natural-Nukleus

Der Natural-Nukleus ist das Programm, das den Kern von Natural enthält. Der Nukleus läuft auf allen Großrechnerbetriebssystemen, z.B. z/OS, z/VSE und BS2000/OSD, und unter allen TP-Monitoren, z.B. Com-plete, CICS und *open*UTM.

Der Natural-Nukleus stellt dem Natural-Benutzer (z.B. Anwendungsentwickler oder Administrator) alle benötigten Funktionen zur Verfügung, z.B. Kommandointerpretation, Objektkompilierung und Objektausführung. Der Natural-Nukleus kann als Shared Nucleus installiert werden und kann dann zeitgleich von mehreren Benutzer genutzt werden.

Dieser Abschnitt behandelt die Hauptbestandteile des Natural-Nukleus:

- Natural-Laufzeitsystem
- Natural-Compiler
- Natural-Kommando-Interpreter
- Konfiguration

Natural-Laufzeitsystem

Das Natural-Laufzeitsystem (Runtime System) funktioniert ähnlich einer virtuellen Maschine, die die zur Ausführung von Objekten in einer mit Natural erstellten Anwendung benötigte Umgebung bereitstellt. Das Natural-Laufzeitsystem interpretiert den Natural-internen Objektcode (binärer Metacode) und führt ihn aus.

Dieser Abschnitt behandelt folgende Themen:

- Objektausführung
- Object Starter und Object Executor

Objektausführung

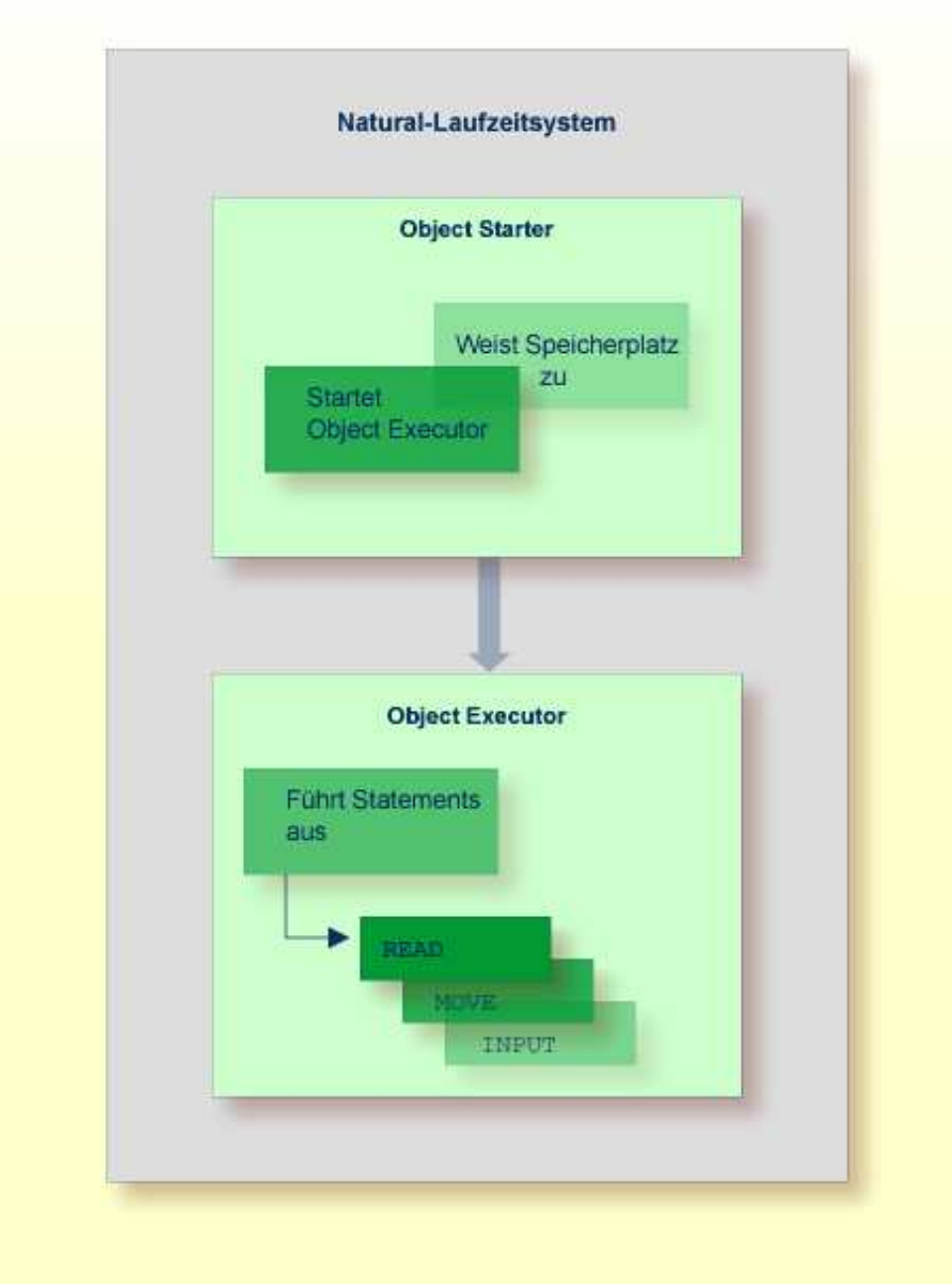
Die Ausführung des Natural-internen Objektcodes erfolgt, wenn die Ausführung eines Natural-Objekts angefordert wird.

Dies geschieht entweder direkt durch einen Benutzer oder indirekt, wenn das Objekt, das zurzeit ausgeführt wird, ein Natural-Statement absetzt, das die Ausführung eines anderen Objekts anfordert. Beispiele: Das Natural-Systemkommando `EXECUTE` bewirkt, dass das mit diesem Kommando angegebene vom Benutzer geschriebene Natural-Programm direkt ausgeführt wird. Das in einem Natural-Programm enthaltene Natural-Statement `CALLNAT` fordert die Ausführung eines Subprogramms an.

Das *Beispiel für das Laden eines Objekts* im Abschnitt *Natural Buffer Pool* veranschaulicht den Verarbeitungsablauf bei der Ausführung eines Natural-Programms.

Object Starter und Object Executor

In dem folgenden Diagramm ist die Ausführung eines Objekts durch das Natural-Laufzeitsystem schematisch dargestellt:



Das Natural-Laufzeitsystem hat zwei Hauptbestandteile: Den Object Starter und den Object Executor.

Der Object Starter ermittelt das auszuführende Objekt im Natural Buffer Pool und weist Speicherplatz für die vom Objekt als Benutzer-Session-Daten verarbeiteten Daten zu. Schließlich übergibt er die Kontrolle an den Object Executor.

Der Object Executor interpretiert die in dem Objekt enthaltenen Natural-Statements und führt sie der Reihe nach aus. In dem obigen Beispiel verarbeitet der Object Executor zunächst das `READ`-Statement, indem er einen Datenbankaufruf absetzt, die angeforderten Datensätze abruft und dann mit dem `MOVE`-Statement weitermacht.

Verwandte Themen:

- *Benutzer-Session-Daten*
- *Natural Buffer Pool*
- *Beispiel für das Laden eines Objekts - Natural Buffer Pool*

Natural-Compiler

Der Natural-Compiler erzeugt die ausführbare Form des Natural-Sourcecode. Beim Natural-Sourcecode handelt es sich um einen menschenlesbaren Programmcode, der im Wesentlichen aus einer Abfolge von Natural-Statements besteht. (Informationen zu den Natural-Statements und Hinweise zur Programmierung finden Sie in der *Statements*-Dokumentation bzw. im *Leitfaden zur Programmierung*.)

Der Natural-Compiler liest den Sourcecode aus dem Arbeitsbereich. Dieser Sourcecode ist Teil der Benutzer-Session-Daten. Er wird mit dem Natural-Systemkommando `READ` oder `EDIT` in den Arbeitsbereich geladen. Der Compiler prüft die Syntax des Sourcecode auf Korrektheit und erzeugt dann den Natural-internen Objektcode, der vom Natural-Laufzeitsystem interpretiert und ausgeführt wird.

Mit Hilfe des entsprechenden Natural-Systemkommandos kann der Benutzer den Sourcecode entweder kompilieren und ausführen oder kompilieren und speichern. Der folgende Abschnitt behandelt die verschiedenen Objektmodularten und die zur Objektkompilierung und –ausführung zur Verfügung stehenden Natural-Systemkommandos:

- Katalogisiertes Objekt
- Source-Objekt
- Kommandos zum Kompilieren
- Beispiel für eine Kompilierung

Katalogisiertes Objekt

Ein katalogisiertes Objekt ist die ausführbare (kompilierte) Form eines Natural-Objekts. Es wird vom Natural-Compiler erzeugt und als Objektmodul in einer Natural-Systemdatei gespeichert. Unter dem Katalogisieren eines Objekts versteht man das Kompilieren des Sourcecodes und das Erstellen eines katalogisierten Objekts. Erzeugt wird ein katalogisiertes Objekt mit Hilfe des Natural-Systemkommandos `CATALOG` oder `STOW`.

Zur Ausführungszeit wird das katalogisierte Objekt in den Natural Buffer Pool geladen und vom Natural-Laufzeitsystem ausgeführt. Natural-Objekte können nur ausgeführt werden oder sich gegenseitig referenzieren, wenn sie als katalogisierte Objekte in einer der Natural-Systemdateien gespeichert wurden.

Ein katalogisiertes Objekt kann nicht geändert oder dekompiliert werden.

Source-Objekt

Ein Source-Objekt (oder ein gespeichertes Objekt) enthält die menschenlesbare Form des Natural-Sourcecode. Der Sourcecode wird mit Hilfe des Natural-Systemkommandos SAVE oder STOW als Source-Objekt in einer der Natural-Systemdateien gespeichert.

Um den in einem Source-Objekt enthaltenen Sourcecode auszuführen, müssen Sie den Sourcecode kompilieren, um generierten Objektcode zu erzeugen, der vom Natural-Laufzeitsystem interpretiert und ausgeführt werden kann.

Kommandos zum Kompilieren

Natural bietet verschieden Systemkommandos zum Kompilieren des Sourcecodes. Je nach verwendetem Systemkommando wird beim Kompilieren noch eine der folgenden Aktionen ausgeführt:

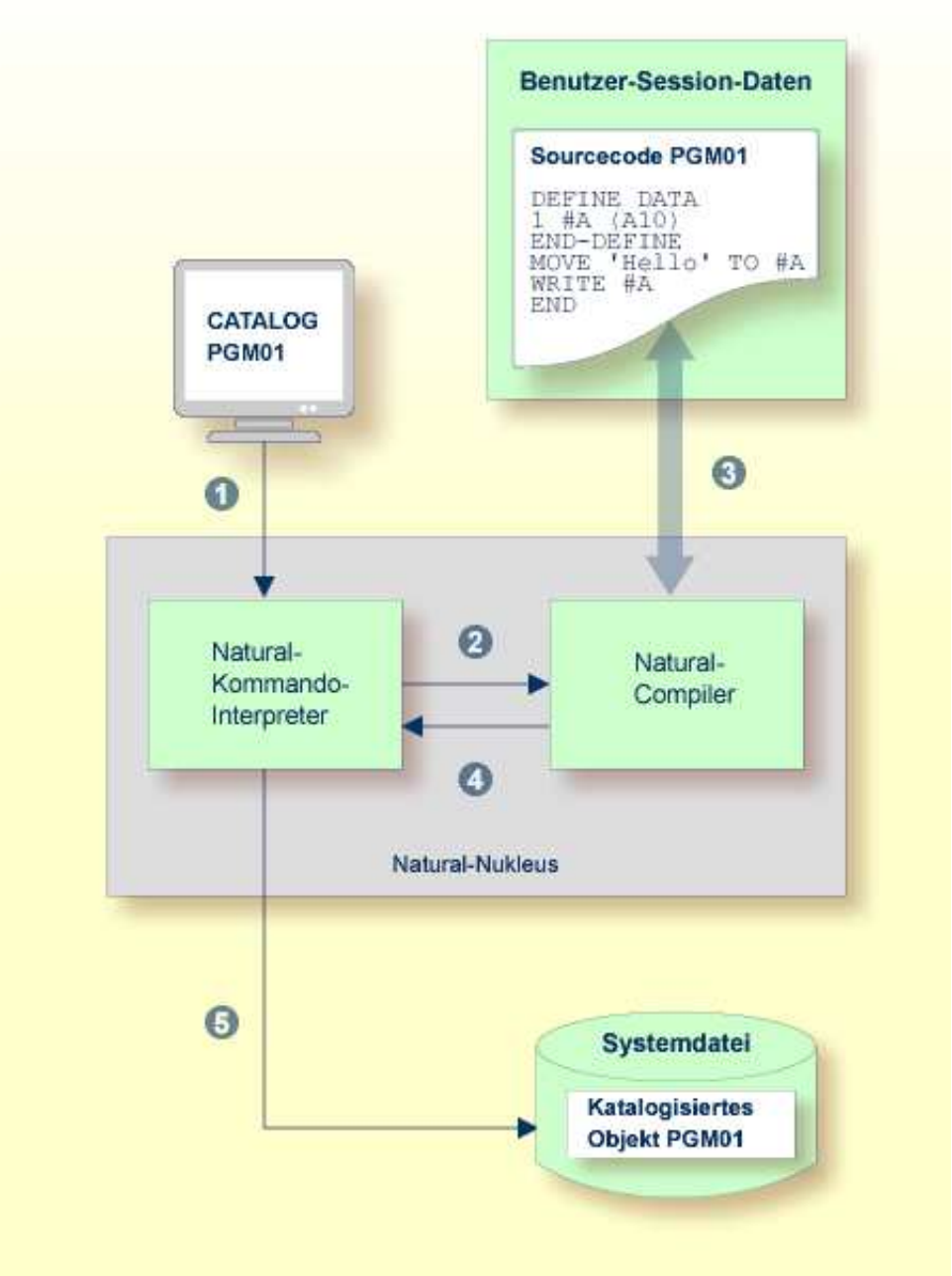
Aktion	Systemkommando
Sourcecode kompilieren. Syntaxprüfung durchführen und Objektcode generieren. Der generierte Objektcode wird nicht als Objektmodul in einer Natural-Systemdatei gespeichert.	CHECK
Sourcecode kompilieren. Nach erfolgreicher Kompilierung den generierten Objektcode als katalogisiertes Objekt in einer Natural-Systemdatei speichern.	CATALOG
Sourcecode kompilieren. Nach erfolgreicher Kompilierung den generierten Objektcode als katalogisiertes Objekt in einer Natural-Systemdatei speichern. Außerdem den ursprünglichen Sourcecode als separates Source-Objekt in einer Natural-Systemdatei speichern.	STOW
Sourcecode eines Natural-Objekts des Typs Programm kompilieren. Nach erfolgreicher Kompilierung den generierten Objektcode sofort ausführen. Der generierte Objektcode wird nicht als Objektmodul in einer Natural-Systemdatei gespeichert.	RUN

Verwandte Themen:

- *Systemkommandos*-Dokumentation
- *Objekttypen - Leitfaden zur Programmierung*

Beispiel für eine Kompilierung

Das folgende Diagramm veranschaulicht den Verarbeitungsablauf beim Kompilieren des Sourcecodes mit Hilfe des Systemkommandos CATALOG:


Benutzer-Session-Daten
Sourcecode PGM01
DEFINE DATA
1 #A (A10)
END-DEFINE
MOVE 'Hello' TO #A
WRITE #A
END
CATALOG
PGM01
1
2
3
4
5
Natural-
Kommando-
Interpreter
Natural-
Compiler
Natural-Nukleus
Systemdatei
Katalogisiertes
Objekt PGM01

Legende

1. Der Benutzer setzt das Natural-Systemkommando `CATALOG PGM01` ab, mit dem er die Kompilierung und Speicherung des zurzeit im Arbeitsbereich befindlichen Objektcodes als katalogisiertes Objekt mit dem Namen `PGM01` anfordert.
2. Der Natural-Kommando-Interpreter interpretiert das Kommando `CATALOG` und leitet die Anforderung an den Natural-Compiler weiter.
3. Der Natural-Compiler liest die zurzeit im Arbeitsbereich befindlichen Natural-Statements, prüft die Syntax auf Korrektheit und erzeugt dann den Objektcode.
4. Der Natural-Compiler gibt die Kontrolle an den Natural-Kommando-Interpreter zurück.
5. Der Natural-Kommando-Interpreter speichert den generierten Objektcode als katalogisiertes Objekt unter dem Namen `PGM01` in der aktuellen Natural-Systemdatei.

Natural-Kommando-Interpreter

Der Natural-Kommando-Interpreter prüft das in einer der Natural-Eingabeaufforderungszeilen eingegebene Kommando und führt es aus.

Wenn das Add-On-Produkt Natural Security installiert ist, kann der Administrator die Ausführung bestimmter Kommandos auf einzelne Benutzer oder eine Benutzergruppe beschränken.

Verwandte Themen:

- *Systemkommandos*-Dokumentation
- *Natural Security*-Dokumentation

Konfiguration

Die Konfiguration einer Natural-Systemumgebung wird mit Natural-Parametern verwaltet.

Diese Parameter dienen zur Standardisierung und Automatisierung der Entwicklungs- und Produktionsvorgänge und zur Anpassung der standardmäßig vorhandenen Einstellungen an die Bedürfnisse einzelner Benutzer. Mittels eines Natural-Parameters können zum Beispiel die Standardeinstellungen für die Report-Erstellung vorgenommen, die Größe eines Reports oder die Größe des erforderlichen Speicherplatzes, z.B. des Arbeitsbereichs eines Editors, festgelegt werden.

Die meisten Eigenschaften einer Natural-Umgebung sind schon von der Software AG vor der Produktauslieferung festgelegt worden. Ihr Natural-Administrator kann verschiedene Standardvorgaben konfigurieren, die für alle Benutzer in Ihrem Unternehmen gültig sind. Jeder Benutzer kann seinerseits die Einstellungen an seine Bedürfnisse anpassen, indem er die Standard-Umgebungseinstellungen mittels eines dynamischen Profilparameters oder eines Session-Parameters überschreibt.

Der folgende Abschnitt behandelt folgende Themen:

- Profilparameter
- Session-Parameter

- Parametrisierungsebenen

Profilparameter

Profilparameter können statisch oder dynamisch angegeben werden.

Statische Parameter werden anlässlich der Natural-Installation im Natural-Parametermodul NATPARM angegeben. Diese Angaben dienen als Standardvorgaben für jede Natural-Session.

Verwandte Themen:

- *Profilparameter* - *Parameter-Referenz*-Dokumentation
- *Profile Parameter Usage* - *Operations*-Dokumentation
- *Using a Natural Parameter Module* - *Operations*-Dokumentation
- *SYSPARM Utility* - *Utilities*-Dokumentation

Session-Parameter

Session-Parameter können in einer aktiven Natural-Session und/oder in einem Natural-Objekt angegeben werden. Hauptzweck der Session-Parameter ist die Steuerung der Ausführung von Natural-Programmen.

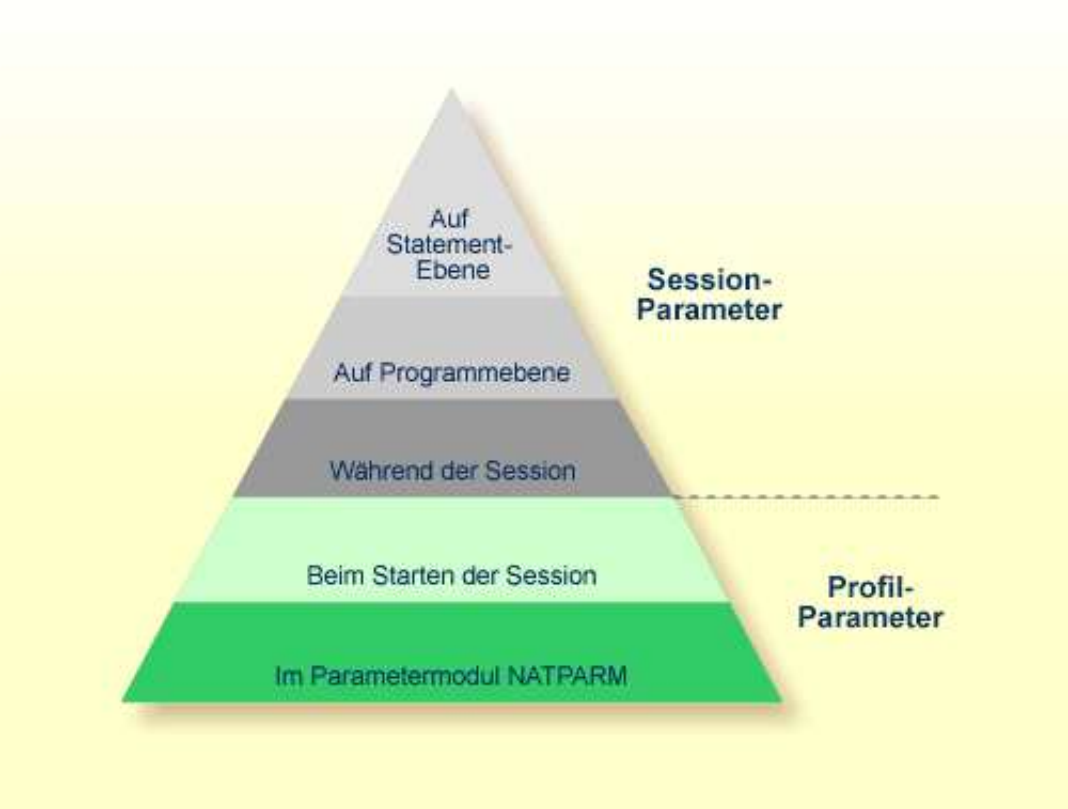
Verwandtes Thema:

- *Session-Parameter* - *Parameter Referenz*-Dokumentation

Parametrisierungsebenen

Die Ebenen, auf denen Natural parametrisiert werden kann, sind hierarchisch strukturiert. Ein auf einer höheren Ebene gesetzter Parameterwert hat Vorrang vor einem auf einer niedrigeren Ebene gesetzten Wert. Wenn zum Beispiel ein Parameter dynamisch angegeben wird, hat der neue Parameterwert Vorrang vor der entsprechenden statischen Parameterangabe, die im Natural-Parametermodul NATPARM vorhanden ist.

Das folgende Diagramm veranschaulicht, wann ein Parameter gesetzt werden kann und zeigt die Natural-Parameterhierarchie mit der niedrigsten Ebene an der Basis und der höchsten Ebene an der Spitze der Pyramide:



Verwandtes Thema:

- *Natural Parameter Hierarchy - Operations*-Dokumentation