Natural für Großrechner

Systemkommandos

Version 4.2.6 für Großrechner

Februar 2010

Natural

Dieses Dokument gilt für Natural für Großrechner ab Version 4.2.6 für Großrechner.

Hierin enthaltene Beschreibungen unterliegen Änderungen und Ergänzungen, die in nachfolgenden Release Notes oder Neuausgaben bekanntgegeben werden.

Inhaltsverzeichnis

1 Systemkommandos

Diese Dokumentation beschreibt die Natural-Systemkommandos.

Systemkommandos führen Funktionen aus, die Sie zum Erstellen, Pflegen oder Ausführen von Natural-Prorammierobjekten benötigen. Außerdem gibt es Systemkommandos, die Sie zum Überwachen und Verwalten Ihrer Natural-Umgebung benutzen können.

Diese Dokumentation ist in die folgenden Abschnitte untergliedert:

Kommandoausführung	Beschreibt die Regeln, die für die Eingabe eines Systemkommandos gelten.
Kommando-Syntax	Erklärt die Symbole, die in den Diagrammen verwendet werden, in denen die Syntax der Systemkommandos beschrieben ist.
Systemkommandos nach Funktion	Liefert eine Übersicht über die nach Funktionen eingeteilten Systemkommandos.
Systemkommandos in alphabetischer Reihenfolge	Beschreibungen der Systemkommandos in alphabetischer Reihenfolge.

2 Kommandoausführung

Kommandoeingabe

Sie können ein Systemkommando an folgenden Stellen absetzen:

- in der **Kommandozeile**;
- in einer `NEXT`- oder `MORE`-Zeile.

Folgende Regeln gelten:

- Bei Kommandoeingaben wird nicht zwischen Groß- und Kleinschreibung unterschieden.
- Kommandoeingaben sind kontextabhängig.
- Manche Systemkommandos betreffen nicht das zur Zeit aktive Objekt, sondern ein anderes Objekt.

Eine Erklärung der in den Syntaxbeschreibungen der Systemkommandos enthaltenen Symbole finden Sie im Abschnitt ***Kommando-Syntax***.

Kommandozeile

Sie können ein Systemkommando in der Natural-Kommandozeile absetzen, die mit einem Pfeil (`===>`) als Eingabeaufforderungszeichnen gekennzeichnet ist.

Manche Systemkommandos können auch über PF-Tasten oder über das Hauptmenü abgesetzt werden.

NEXT-Zeile

Die `NEXT`-Zeile erscheint in einer Natural-Anwendung oder einem Natural-Programm, wenn keine weitere Ausgabe zu erwarten ist.

MORE-Zeile

Die MORE-Zeile wird am unteren Rand eines Ausgabeschirms angezeigt, um darauf hinzuweisen, dass keine weitere Ausabe mehr zu erwarten ist. Wenn Sie in der MORE-Zeile ein Systemkommando absetzen, wird die Ausführung des aktuellen Programms unterbrochen, und das eingegebene Systemkommando wird ausgeführt.

3 Systemkommando-Syntax

Syntax-Elemente

In den Diagrammen, in denen die Syntax der Systemkommandos beschrieben ist, werden folgende Symbole verwendet:

Element	Beschreibung
ABCDEF	Bei Angaben, die in Großbuchstaben dargestellt sind, handelt es sich um Natural-Schlüsselwörter, die Sie genauso eingeben müssen wie angegeben.
ABCDEF	Ist von mehreren wahlweise verwendbaren Elementen, die in Großbuchstaben dargestellt sind, eins unterstrichen (kein Hyperlink!), handelt es sich um das jeweils gültige Standardelement. Lassen Sie das Element weg, gilt der unterstrichene Wert.
ABCDEF	Ist ein Teil eines Wortes in Großbuchstaben unterstrichen (kein Hyperlink!), kann der unterstrichene Teil als Abkürzung für das jeweilie Wort verwendet werden.
abcdef	Bei Angaben, die in *kursiven* Kleinbuchstaben dargestellt sind, handelt es sich um variable Informationen, die Sie selbst angeben können.
[ ]	Syntax-Elemente, die in eckigen Klammern stehen, sind nicht unbedingt erforderlich, sie können gegebenenfalls weggelassen werden. Wenn innerhalb der eckigen Klammern mehrere Zeile übereinander stehen, dann enthält jede Zeile eine optionale Alternative. Sie dürfen nur eine diese Alternativen eingeben.
{ }	Von mehreren Elementen, die in einer geschweiften Klammer untereinander stehen, muß genau eines angegeben werden.
|	Eines der durch diesen senkrechten Strich voneinander getrennten Elemente muss eingegeben werden.
...	Drei Punkte hinter einem Element bedeuten, dass Sie das Element mehrmals angeben dürfen. Gegebenenfalls gibt eine Zahl hinter den drei Punkten an, wie oft das Element angegeben werden kann. Ist das Element vor den drei Punkten ein in eckige oder geschweifte Klammern eingeschlossener Ausdruck, gilt die Wiederholmöglichkeit für den gesamten in Klammern stehenden Ausdruck.
,...	Ein Komma und drei Punkte hinter einem Element bedeuten, dass Sie das Element mehrmals angeben dürfen, wobei die einzelnen Elemente durch Kommas voneinander getrennt werden müssen. Gegebenenfalls gibt eine Zahl hinter dem Komma und den drei Punkten an, wie oft das Element angegeben werden darf. Ist das Element vor dem Komma und den drei Punkten ein in eckige oder geschweifte Klammern eingeschlossener Ausdruck, gilt die Wiederholmöglichkeit für den gesamten in Klammern stehenden Ausdruck.
:...	Ein Doppelpunkt und drei Punkte hinter einem Element bedeuten, dass Sie das Element mehrmals angeben dürfen, wobei die einzelnen Elemente durch Doppelpunkte voneinander getrennt werden müssen. Gegebenenfalls gibt eine Zahl hinter dem Doppelpunkt und den drei Punkten an, wie oft das Element angegeben werden darf.

Element	Beschreibung
	Ist das Element vor dem Doppelpunkt und den drei Punkten ein in eckige oder geschweifte Klammern eingeschlossener Ausdruck, gilt die Wiederholmöglichkeit für den gesamten in Klammern stehenden Ausdruck.
Andere Symbole (außer [ ] { } | ... ,... :...)	Alle anderen Symbole außer den in dieser Tabelle definierten müssen genau so eingegeben werden wie angegeben. **Ausnahme:** Der skalare SQL-Verkettungsoperator wird durch zwei senkrechte Striche dargestellt, die genauso eingegeben werden müssen, wie sie in der Syntax-Definition erscheinen.

Kommandosyntax-Beispiel

```
CATALOG [object-name [library-id]]
```

- `CATALOG` ist ein Natural-Schlüsselwort, das Sie genauso eingeben müssen, wie es dasteht. Die Unterstreichung bedeutet, dass Sie es auch in abgekürzter Form als `CAT` eingeben können.
- *`object-name`* und *`library-id`* sind variable Operanden, an deren Stelle Sie den gewünschten Programmnamen und die ID der Library, in der das Programm enthalten ist, eingeben.
- Die eckigen Klammern bedeuten, dass Sie *`object-name`* und *`library-id`* angeben können, aber nicht müssen. Die Anordnung der Klammern besagt, dass Sie `CATALOG` alleine eingeben können oder gefolgt von entweder nur einem Programmnamen oder einem Programmnamen und einer Library-ID; allerdings können Sie keine Library-ID eingeben, ohne gleichzeitig einen Programmnamen einzugeben.

4 Systemkommandos nach Funktion

Dieses Kapitel liefert eine Übersicht über die nach Funktionen eingeteilten Systemkommandos.

In Natural navigieren

Kommando	Funktion
FIN	Beendet eine Natural-Session
LOGOFF	Setzt die Library-ID auf SYSTEM und das Adabas-Passwort auf Leerzeichen. Der Source-Inhalt des Editor-Arbeitsbereichs bleibt erhalten.
LOGON	Wählt eine bestimmte Library für den Benutzer. In dieser Library werden dann alle von Ihnen während der Session in Source- und/oder Objektform erstellten Objekte gespeichert (es sei denn, Sie geben bei einem SAVE-, CATALOG- oder STOW-Kommando ausdrücklich eine andere Library an).
MAINMENU	Schaltet die Anzeige des Natural-Hauptmenüs ein- bzw. aus oder ruft ein Programm auf, das einen eigenen Menüschirm anzeigt.
RETURN	Ermöglicht die Rückkehr zu einer bestimmten vorherigen Natural-Anwendung (oder der Ausgangsanwendung), die als Rückkehrpunkt mit dem SETUP Kommando gesetzt wurde.
SETUP	Ermöglicht es, Rückkehrpunkte zu setzen, zu denen Sie dann später mit dem Systemkommando RETURN zurückkehren können. Das erlaubt es Ihnen, während einer Natural-Session problemlos von einer Anwendung in eine andere zu gelangen.

Entwicklungsumgebung einstellen

Kommando	Funktion
COMPOPT	Mit diesem Kommando können Sie verschiedene Kompilierungsoptionen setzen. Die Optionen werden wirksam, wenn ein Natural-Programmierobjekt kompiliert wird.
GLOBALS	Mit diesem Kommando können Sie Natural-Session-Parameter setzen.
KEY	Mit diesem Kommando können Sie bestimmten Funktionstasten auf der Tastatur Ihres Video-Terminals eine Funktion zuweisen oder eine zugewiesene Funktion ändern.
SYSCP	Mit diesem Kommando rufen Sie die SYSCP-Utility auf, die Code-Page-Informationen liefert und benutzt werden kann, um Codepages für Natural-Source-Objekte zu verwalten.
SYSPARM	Mit diesem Kommando rufen Sie die SYSPARM-Utility auf. Sie können diese Utility benutzen, um Zeichenketten mit Natural-Profilparametern zu erstellen und zu pflegen, die als Natural-Profile gespeichert werden.
PROFILE	Dieses Kommando ist nur verfügbar, wenn Natural Security installiert ist. Es zeigt Ihr gegenwärtig gültiges Benutzungsprofil an. Dieses Profil informiert Sie über die Bedingungen und Voraussetzungen, die in Ihrer aktuellen Natural-Umgebung für Sie gelten.
SYSPROD	Mit diesem Kommando können Sie feststellen, welche Produkte in Ihrer Natural-Umgebung installiert sind: d.h. Natural selbst, sowie Produkte, die mit bzw. unter Natural laufen.

Kommando	Funktion
SYSPROF	Mit diesem Kommando können Sie sich die gegenwärtigen Definitionen der Natural-Systemdateien anzeigen lassen.

Programmobjekte bearbeiten und speichern

Kommando	Funktion
CATALL	Mit diesem Kommando können Sie *alle* Objekte in der aktuellen Library gleichzeitig in Source- und/oder Objektform speichern.
CATALOG	Mit diesem Kommando kompilieren Sie das im Arbeitsbereich eines Editors befindliche Source-Objekt und speichern nach erfolgreicher Prüfung das resultierende Objektmodul in der Natural-Systemdatei.
CHECK	Dieses Kommando dient dazu, den gerade im Arbeitsbereich des Programm-Editors befindlichen Sourcecode auf Syntaxfehler zu überprüfen. Die Syntaxprüfung erfolgt auch im Rahmen der Ausführung der Kommandos RUN, CATALL, CATALOG und STOW.
CLEAR	Mit diesem Kommando löschen Sie den Inhalt des Editor-Arbeitsbereichs. Falls Sie ein neues Objekt erstellen wollen, und es befindet sich noch ein anderes Source-Objekt im Arbeitsbereich, empfiehlt es sich, den Arbeitsbereich mit diesem Kommando zu löschen.
DELETE	Dieses Kommando dient dazu, Natural-Programmierobjekte aus der Natural-Systemdatei zu löschen. Sie können angeben, ob der Sourcecode, das Objektmodul oder beide gelöscht werden sollen
EDIT	Mit diesem Kommando rufen Sie einen der Natural-Editoren auf, um die Source-Form eines Natural-Programmierobjekts zu editieren.
LIST	Mit diesem Kommando können Sie sich den Sourcecodes eines einzelnen Objekts anzeigen oder mehrere in Ihrer aktuellen Library gespeichterte Objekte auflisten lassen.
READ	Mit diesem Kommando können Sie ein in Sourceform gespeichertes Objekt in den Arbeitsbereich des entsprechenden Editors einlesen. Ein bereits im Editor befindliches Objekt wird dadurch überschrieben.
RENAME	Mit diesem Kommando können Sie ein Natural-Objekt umbenennen und außerdem seinen Objekttyp ändern.
RENUMBER	Mit diesem Kommando erreichen Sie, dass die Sourcecodezeilen des Objekts, das sich gerade im Programm-Editor befindet, neu durchnumeriert werden.
SAVE	Dieses Kommando dient dazu, ein Source-Objekt in der Natural-Systemdatei zu speichern. Der Inhalt des Editor-Arbeitsbereichs wird dadurch nicht beeinflußt.
SCAN	Mit diesem Kommando können Sie den Sourcecode von Objekten nach einer bestimmten Zeichenkette absuchen; darüber hinaus besteht die Möglichkeit, die gesuchte Zeichenkette durch eine andere Zeichenkette zu ersetzen.
STOW	Dieses Kommando dient dazu, ein Objekt gleichzeitig in Sourceform und Objektform in der Natural-Systemdatei zu kompilieren und zu speichern. Es hat die gleiche Wirkung wie ein CATALOG-Kommando mit anschließend abgesetztem SAVE-Kommando.

Kommando	Funktion
STRUCT	Dieses Kommando können Sie verwenden, um die Sourcecode-Zeilen eines Programms entsprechend der Programmstruktur einzurücken. Diverse zusätzliche Anzeigen verdeutlichen die Programmstruktur und erlauben es Ihnen so, etwaige Inkonsistenzen in der Struktur aufzuspüren.

Programme ausführen

Kommando	Funktion
EXECUTE	Dieses Kommando dient dazu, ein Natural-Objektmodul des Typs Programm auszuführen. Das Objektmodul muß in der Natural-Systemdatei katalogisiert (d.h. in Objektform gespeichert) oder in den Natural-Nukleus eingebunden sein.
RUN	Dieses Kommando dient dazu, ein Source-Programm zu kompilieren und auszuführen. Das auszuführende Programm kann sich entweder in der Natural-Systemdatei oder im Arbeitsbereich des Editors befinden.

Utilities aufrufen

Kommando	Funktion
SYSDDM	Mit diesem Kommando rufen Sie die SYSDDM-Utility auf. Diese bietet Funktionen, die zum Erstellen und zum Pflegen von Natural Data Definition Modules (DDMs) benötigt werden.
SYSERR	Mit diesem Kommando rufen Sie die SYSERR-Utility auf, mit der Sie Ihre eigenen anwendungsspezifischen Meldungen schreiben und pflegen können.
SYSNCP	Mit diesem Kommando rufen Sie die SYSNCP-Utility auf, mit der Sie die in Ihren Anwendungen zu verwendenden Kommado-Prozessoren erstellen und pflegen können.
SYSRPC	Mit diesem Kommando rufen Sie die SYSRPC-Utility auf. Diese Utility bietet Funktionen zum Verwalten von Natural Remote Procedure Calls (RPC).

Programmierobjekte übertragen

Kommando	Funktion
SYSMAIN	Mit diesem Kommando rufen Sie die SYSMAIN-Utility auf. Mit dieser Utility können Sie Natural-Objekte kopieren, verschieben und löschen. Die Utility dient außerdem dazu, innerhalb des Natural-Systems Objekte mittels der Import-Funktion von einer Umgebung in eine andere Umgebung zu übertragen.
SYSOBJH	Mit diesem Kommando rufen Sie den Object Handler auf. Mit dem Object Handler können Sie Natural- und Nicht-Natural-Objekte zwecks Verteilung in Natural-Umgebungen handhaben.

Monitor- und Debug-Funktionen benutzen

Kommando	Funktion
BUS	Mit diesem Systemkommando rufen Sie die Funktion Buffer Usage Statistics der SYSTP-Utility auf. Diese Funktion liefert Ihnen Informationen darüber, welche Puffer mit welcher Größe für die aktive Natural-Session angelegt wurden und wieviel Puffer-Platz tatsächlich benutzt wird. **Anmerkung:** Das Kommando BUS hat die gleiche Funktion wie das Kommando SYSBUS, das nicht mehr zur Verfügung steht.
DUMP	Liefert Informationen, die das technische Personal der Software AG benötigt, um einen Fehler zu finden, der zum Abbruch (Abend) des Natural-Systems geführt hat.
RPCERR	Zeigt die Nummer und die Meldung des letzten Natural-Fehlers, falls dieser den Remote Procedure Call (RPC) betrifft, sowie den letzten Reason Code des Brokers und die zugehörige Meldung an.
SYSADA (ADACALL)	Mit diesem Kommando rufen Sie die ADACALL-Utility auf, die sich in der Library SYSADA befindet. Mit der Utility können Sie von einem Großrechner-Natural aus Adabas-Direktkommandos (Native Commands) direkt an eine Adabas-Datenbank absetzen.
SYSBPM	Mit diesem Kommando rufen Sie die SYSBPM-Utility auf. Diese liefert statische Informationen über den aktuellen Status des Natural Buffer Pool, über den Buffer Pool Cache und über die Objekte, die sich gerade im Buffer Pool und im Buffer Pool Cache befinden.
SYSEDT	Mit diesem Kommando rufen Sie die SYSEDT-Utility for Editor Buffer Pool Services auf. Die SYSEDT-Utility ist nur für Natural-Administratoren gedacht.
SYSTP	Mit diesem Kommando rufen Sie die SYSTP-Utility auf, mit der Sie verschiedene TP-Monitor-spezifische Charakteristika von Natural überwachen und steuern können.
TECH	Mit diesem Kommando können Sie sich technische und andere Informationen über Ihre Natural-Session anzeigen lassen.
TEST	Mit diesem Kommando rufen Sie die Debugging-Funktion auf.

Kommando	Funktion
TEST DBLOG	Mit diesem Kommando rufen Sie die DBLOG-Utility auf, mit der Sie Datenbankaufrufe protokollieren können.

SQL-Databank-spezifische Kommandos

Command	Function
LISTDBRM	Ist nur mit Natural for DB2 und Natural for SQL/DS verfügbar. Anzeige von vorhandenen DBRMs (Natural for DB2) bzw. vorhandene Pakete (Natural for SQL) von Natural-Programmen oder Anzeige von Natural-Programmen, welche ein gegebenes DBRM referenzieren.
LISTSQL	Ist nur mit Natural for DB2, Natural SQL Gateway und Natural for SQL/DS verfügbar. Listet diejenigen Natural-Statements im Sourcecode eines Natural-Objekts auf, die in Verbindung mit einem Datenbankzugriff stehen, sowie die entsprechenden SQL-Kommandos, in die sie übersetzt wurden.
LISTSQLB	Ist nur mit Natural for DB2 verfügbar. Dient zum Aufrufen der Funktion **Explain all SQL Statements**.
SQLDIAG	Ist nur mit Natural for DB2 verfügbar. Liefert Diagnoseinformationen über das zuletzt ausgeführte SQL-Statement (anders als das GET DIAGNOSTICS-Statement). Diese Informationen werden gesammelt, während das vorhergehende Statement ausgeführt wird. Einige der beim GET DIAGNOSTICS-Statement verfügbaren Informationen sind auch in der SQLCA vorhanden.
SQLERR	Ist nur mit Natural for DB2, Natural SQL Gateway und Natural for SQL/DS verfügbar. Dient zur Anzeige von Informationen über den zuletzt aufgetretenen SQL-Fehler.
SYSDB2	Mit diesem Kommando rufen Sie die Natural Tools for DB2 auf, wenn Natural for DB2 installiert ist. Weitere Informationen siehe *Using Natural Tools for DB2* im Teil *Natural for DB2* der *Database Management System Interfaces*-Dokumentation.

Kommando	Funktion
LISTSQL	Ist nur mit Natural for DB2 und Natural for SQL/DS verfügbar. Listet diejenigen Natural-Statements im Sourcecode eines Natural-Objekts auf, die in Verbindung mit einem Datenbankzugriff stehen, sowie die entsprechenden SQL-Kommandos, in die sie übersetzt wurden.
SQLERR	Ist nur mit Natural for DB2 und Natural for SQL/DS verfügbar. Mit dem Kommando erhalten Sie Informationen über einen SQL-Fehler.

Kommando	Funktion
SYSDB2	Mit diesem Kommando rufen Sie die Natural Tools for DB2 auf, wenn Natural for DB2 installiert ist.

Natural Optimizer Compiler-spezifische Kommandos

Die folgenden Systemkommandos sind nur verfügbar, wenn der Natural Optimizer Compiler installiert ist. Weitere Informationen siehe *Natural Optimizer Compiler*-Dokumentation.

Kommando	Funktion
NOCOPT	Mit diesem Kommando können Sie verschiedene beim Start von Natural gesetzte Optionen für den Natural Optimizer Compiler anzeigen und ändern.
NOCSHOW	Liefert Buffer-Informationen zur Ausgabe der PGEN-Option des Natural Optimizer Compiler.
NOCSTAT	Liefert Statistikdaten zu Programmen, die für die Verarbeitung durch den Natural Optimizer Compiler geeignet sind.

Sonstige Kommandos

Kommando	Funktion
AIV	Zeigt an, welche anwendungsunabhängigen Variablen (Application-Independent Variables, AIVs) zur Zeit aktiv sind.
HELP	Ruft das Natural-Hilfesystem auf. Es liefert Informationen zu Natural-Statements, -Kommandos und Fehlermeldungen.
INPL	Ruft die INPL-Utility auf. Diese Utility dient *nur* zum Laden von Software-AG-Installationsdatasets in die Systemdateien.
LAST	Mit diesem Kommando können Sie sich die zuletzt ausgeführten Systemkommandos anzeigen lassen und sie erneut ausführen.
LASTMSG	Zeigt zusätzliche Informationen zu der zuletzt aufgetretenen Fehlersituation an.
MAIL	Dieses Kommando ist nur verfügbar, wenn Natural Security installiert ist. Mit diesem Kommando können Sie eine Mailbox aufrufen, um ihren Inhalt sowie das Ablaufdatum zu ändern. Eine Mailbox ist eine Art „Schwarzes Brett“ zur Übermittlung von Informationen, das mit Natural Security definiert wird.
ROUTINES	Mit diesem Kommando können Sie feststellen, welche Objekte in Ihrer aktuellen Library welche externen Subroutinen verwenden.
SYSEXT	Mit diesem Kommando rufen Sie die SYSEXT-Anwendung auf. Sie enthält verschiedene Natural-Programmierschnittstellen (API).

Kommando	Funktion
SYSEXV	Mit diesem Kommando rufen Sie die SYSEXV-Anwendung mit Beispielen der neuen Merkmale der aktuellen Natural-Version auf.
SYSFILE	Mit diesem Kommando können Sie die SYSFILE-Funktion der SYSTP-Utility aufrufen. Diese Funktion liefert Informationen zu den verfügbaren Arbeitsdateien und Druckdateien (Work und Print Files).
UNLOCK	Mit diesem Kommando können Sie Natural-Sourceobjekte lokal in einer Natural-Großrechnerumgebung entsperren.
UPDATE	Mit diesem Kommando können Sie verhindern (bzw. ermöglichen), dass ein auszuführendes Programm Datenänderungen auf der Datenbank durchführt.
XREF	Ist nur verfügbar, wenn Predict installiert ist. Mit diesem Kommando können Sie die Verwendung der Predict-Funktion Active Cross-References steuern. Mit Hilfe der aktiven Referenzen wird im Predict Data Dictionary automatisch dokumentiert, welche Objekte von einem Programm bzw. einer Data Area referenziert werden.

5 AIV

AIV

Mit diesem Systemkommando können Sie sich anzeigen lassen, welche anwendungsunabhängigen Variablen (Application-Independent Variables, AIVs) zur Zeit aktiv sind.

Wenn Sie das Kommando ausführen, wird eine Liste mit Namen, Format und Länge aller AIVs angezeigt. Auf der Liste können Sie eine AIV mit dem Kommando DI (Display) markieren, um sich den Inhalt der AIV anzeigen zu lassen. Sie können sich den Inhalt in alphanumerischer und in hexadezimaler Darstellung anzeigen lassen. Mit den Tasten PF10 und PF11 können Sie zwischen alphanumerischer und hexadezimaler Anzeige hin- und herschalten.

Weitere Informationen siehe:

- DEFINE DATA-Statement (*Definition von anwendungsunabhängigen Variablen*) in der *Statements*-Dokumentation
- *Benutzervariablen* im *Leitfaden zur Programmierung*

6 BUS

BUS

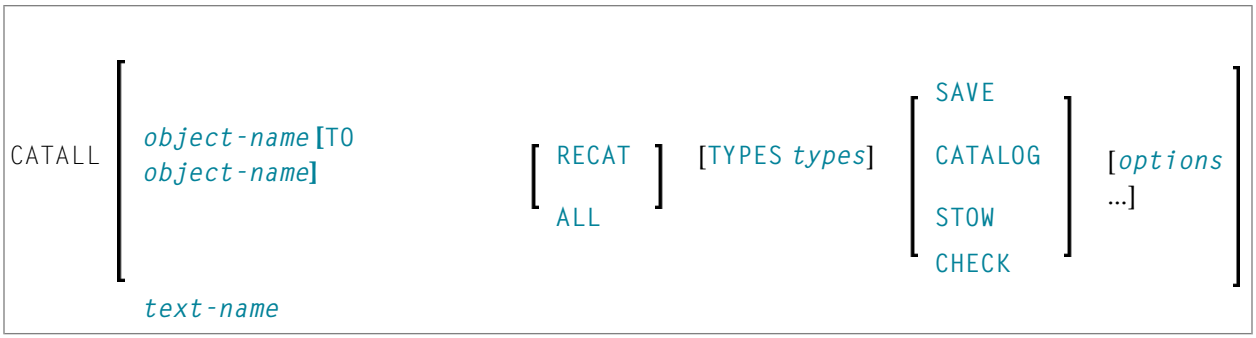
Mit diesem Systemkommando rufen Sie die Funktion **Buffer Usage Statistics** der SYSTP-Utility auf. Diese Funktion liefert Ihnen Informationen darüber, welche Puffer mit welcher Größe für die aktive Natural-Session angelegt wurden und wieviel Puffer-Platz tatsächlich benutzt wird. Näheres zu dieser Funktion finden Sie in der Utilities-Dokumentation.

Anmerkung: Das Kommando BUS hat die gleiche Funktion wie das Kommando SYSBUS, das nicht mehr zur Verfügung steht.

Näheres zu der Funktion Buffer Usage Statistics finden Sie in der *Utilities*-Dokumentation.

Programmierschnittstelle (API): USR1019N. Siehe auch *SYSEXT - Natural Application Programming Interfaces* in der *Utilities*-Dokumentation.

7 CATALL



Mit dem Systemkommando CATALL können Sie alle Objekte in der aktuellen Library gleichzeitig in Source- und/oder Objektform speichern.

Wenn Sie das Kommando CATALL ohne zusätzliche Optionen eingeben, erscheint der Schirm **Catalog Objects in Library** auf dem Sie die nachfolgend beschriebenen Funktionen auswählen können. Sie können das Kommando CATALL aber auch als Direktkommando unter Verwendung der oben dargestellten **Syntax** benutzen.

Außerdem können Sie mit Hilfe des Subprogramms CATALLU2 standardmäßig Funktionen auf dem **Catalog Objects in Library**-Schirm auswählen. Darüber hinaus können Sie mit CATALLU2 auswählen, ob der User Exit in Batch und/oder im Command Mode aufgerufen wird. Dieses Subprogramm wird in Source-Form in der Library SYSTEM (FNAT) mitgeliefert. Um das Subprogramm zu aktivieren, müssen Sie es entsprechend den Angaben in der Source modifizieren, es anschließend katalogisieren und in die Library SYSLIB kopieren. Das Subprogramm wird aufgerufen, bevor der Schirm **Catalog Objects in Library** ausgegeben wird.

Siehe auch *Object Naming Conventions* in der *Using Natural*-Dokumentation.

Dieses Dokument beschreibt die Funktionen des Catalog Objects in Library-Schirms.

Catalog Objects from/to

Wenn Sie CATALL für *alle* Objekte der ausgewählten Typen in der aktuellen Library ausführen möchten, geben Sie einen Stern (*) als Objektname im **from**-Feld ein.

Wenn Sie CATALL nur für einen bestimmten Bereich von Objekten ausführen möchten, können Sie im **from**-Feld Stern-Notation (*) und Wildcard-Notation (?) für den Namen verwenden, und zwar in der gleichen Weise wie beim Systemkommando LIST beschrieben.

Sie können auch einen Anfangs- und Endwert für einen bestimmten Bereich von Objekten angeben, indem Sie in den beiden Feldern **from** und **to** entsprechende Objektnamen (ohne Stern- oder Wildcard-Notation) eingeben.

Statt in diese Felder Werte einzugeben, können Sie auch Objekte von einer Auswahlliste auswählen (siehe unten).

Sie haben auch die Möglichkeit, im **from**-Feld den Namen eines Objekts vom Typ Text anzugeben, der eine Liste von CATALL-Kommandos enthält. Die in dem Text enthaltenen CATALL-Kommandos werden dann ausgeführt. Sie können einen solchen Text entweder von Hand erstellen oder bei Verwendung der **Auswahlliste** (siehe unten) automatisch erstellen.

Recatalog Only Existing Modules, or Catalog All Sources

Diese Option gilt nur für die Funktionen **Catalog** und **Stow**:

- Wenn Sie das erste der beiden Felder markieren, werden nur die Objekte neu katalogisiert, die in der aktuellen Library bereits in Objekt-Form vorhanden sind; Objekte, die nur in Source-Form vorhanden sind, werden nicht katalogisiert.
- Wenn Sie das zweite der beiden Felder markieren, werden *alle* ausgewählten Objekte katalogisiert.

Anmerkung: Diese Option gilt nicht für Objekte vom Typ Copycode und Text.

Select Object Types

Standardmäßig gilt CATALL für Objekte jeglichen Objekttyps in der aktuellen Library (alle Objekttypen sind mit X markiert).

Wenn Sie bestimmte Objekttypen vom CATALL ausnehmen möchten, überschreiben Sie das betreffende X mit einem Leerzeichen.

Select Function

Sie können eine der folgenden Funktionen auswählen, die für die ausgewählten Objekte ausgeführt werden soll: SAVE, CATALOG, STOW oder CHECK. Die Funktionen entsprechen den gleichnamigen Systemkommandos.

Anmerkung: Objekte vom Typ Copycode und Text werden immer mit SAVE gespeichert, auch wenn Sie die Funktion STOW auswählen. Sie werden jedoch nicht gespeichert, wenn Sie CATALOG auswählen.

Select Options

Sie können eine oder mehrere der folgenden Optionen für die CATALL-Verarbeitung auswählen:

Condition Code in Batch	Wenn Sie CATALL im Batch-Betrieb ausführen und diese Option mit einem Zeichen markieren, wird Condition Code 55 zurückgegeben, falls entweder bei der CATALL-Ausführung ein Syntaxfehler entdeckt wird oder innerhalb des angegebenen Bereiches zu verarbeitender Objekte keine Objekte gefunden wurden (nur bei CATALOG und STOW möglich).
Renumber Source-Code Lines	Standardmäßig werden mit den Funktionen SAVE und STOW die Sourcecode-Zeilen von Source-Objekten auch neu numeriert. Wenn Sie keine automatische Neunumierung der Zeilen wünschen, überschreiben Sie das X in diesem Feld mit einem Leerzeichen.
Keep Result List	CATALL erzeugt eine Ergebnisliste. Wenn Sie diese Liste für den weiteren Gebrauch aufheben möchten, markieren Sie dieses Feld mit einem Zeichen. Die Library SYSEXT enthält eine Programmierschnittstelle (API) USR1024N, mit der Sie die Ergebnisliste ausgeben können. Sie können die Ergebnisliste auch mit einem weiteren CATALL-Kommando wieder zur Anzeige bringen. Weil die Parameter ebenfalls in der Ergebnisliste gespeichert werden, sind die Parameter des CATALL-Kommandos, mit dem die Ergebnisliste erstellt wurde, gültig. In diesem Fall bewirkt das Batch-CATALL-Kommando die Ausgabe einer Meldung, und weil keine Module katalogisiert werden, wird der Job mit dem Condition Code 56 beendet. Online werden Sie, falls eine Library eine Ergebnisliste enthält, gefragt, ob die vorherige Liste angezeigt oder ein neuer CATALL-Lauf gestartet werden soll.
Processing Information	Online zeigt CATALL während der Verarbeitung eine laufende Anzeige von Verarbeitungsstatus-Informationen. Während der Batch-Verarbeitung gibt CATALL nur diejenigen Module aus, die einen Fehler verursacht haben. Wenn Sie diese Anzeige nicht wünschen, überschreiben Sie das X in diesem Feld mit einem Leerzeichen.
Error Report	Am Ende der Verarbeitung zeigt CATALL eine Liste der aufgetretenen Fehler an. Wenn Sie diese Fehlerliste nicht wünschen, überschreiben Sie das X in diesem Feld mit einem Leerzeichen.
Extended Error Report	Der Fehlerbericht wird in erweiterter Form ausgegeben, einschließlich der Verzeichnisinformationen, Fehlerzeile und Fehlermeldung. Wenn Sie den erweiterten Fehlerbericht wünschen, markieren Sie dieses Feld mit einem X.
PF4 AddOp	Nach Drücken von PF4 erscheint ein Fenster, in dem Sie zusätzliche Optionen auswählen oder eingeben können.

	`Error Text Member`: Geben Sie hier den Namen eines Natural-Text-Members ein. Es wird eine Liste der bei einem `CATALL`-Lauf aufgetretenen Fehler in dieses Text Member geschrieben.

Auswahlliste

Wenn Sie `CATALL` nur für bestimmte Objekte ausführen möchten, können Sie die gewünschten Objekte von einer Auswahlliste auswählen.

Hierzu machen Sie zunächst die gewünschten Angaben unter **Select Function** und **Select Options** und drücken dann PF5. Sie erhalten dann eine Liste der in der aktuellen Library gespeicherten Objekte.

Die Liste entspricht der des Systemkommandos `LIST`. Auch das Blättern in der Auswahlliste und die Angabe neuer Kriterien für die Auswahlliste erfolgt in der gleichen Weise wie beim `LIST`-Kommando.

Auf der Liste wählen Sie die gewünschten Objekte aus, indem Sie sie in der Spalte **Cmd** mit einem Zeichen markieren. Um *alle* Objekte der aktuellen Auswahlliste gleichzeitig auszuwählen, drücken Sie PF5. Anschließend können Sie in der Liste blättern, andere Auswahlkriterien angeben und weitere Objekte auswählen.

Nachdem Sie alle gewünschten Objekte ausgewählt haben, drücken Sie PF3.

Dann erscheint ein Fenster, das Ihnen die Möglichkeit bietet, die getroffene Objektauswahl zu speichern, um sie für spätere `CATALL`-Verarbeitungen wiederzuverwenden:

- Falls Sie in das Fenster einen Namen eingeben, wird die getroffene Auswahl (in Form von `CATALL`-Kommandos) automatisch in einem Objekt vom Typ Text dieses Namens gespeichert. Dieser Text-Name kann später im Feld **Catalog Objects from** des **Catalog Objects in Library**-Schirms eingegeben werden (vgl. oben).
- Falls Sie dies nicht wünschen, drücken Sie EINGABE ohne einen Namen in das Fenster einzugeben.

Anschließend beginnt `CATALL`, die ausgewählten Objekte zu verarbeiten.

Direktkommando-Syntax

Für die verschiedenen Angaben, die Sie auf dem **Catalog Objects in Library**-Schirm machen können, gibt es auch entsprechende Optionen, die Sie direkt mit dem Systemkommando `CATALL` angeben können:

object-name **TO** *object-name*	Entspricht den Feldern **Catalog Objects from** und **to** des **Catalog Objects in Library**-Schirms, siehe ***Catalog Objects from/to***.
RECAT / ALL	Entspricht den Optionen **Recatalog Only Existing Modules** oder **Catalog All Sources** des **Catalog Objects in Library**-Schirms. `RECAT` ist der Standardwert, siehe ***Recatalog Only Existing Modules, or Catalog All Sources***.
TYPES *types*	Entspricht den markierten Objekttypen des **Catalog Objects in Library**-Schirms, siehe ***Select Object Types***. Mögliche *types* sind: G - Global Data Areas A - Parameter Data Areas L - Local Data Areas C - Copycodes T - Texte S - Subroutinen N - Subprogrammme H - Helproutinen M - Maps P - Programme 4 - Klassen * - Alle Typen (gilt standardmäßig) Die *types* müssen als *eine* Zeichenkette angegeben werden (z.B. `LAG` für Local, Parameter und Global Data Areas). Standardmäßig gilt `CATALL` für alle Objektarten in der aktuellen Library.
SAVE / CATALOG / STOW / CHECK	Entspricht den gleichnamigen Funktionen des **Catalog Objects in Library**-Schirms, siehe ***Select Function***. `CATALOG` ist der Standardwert.

options	Diese Optionen entsprechen den **Select Options** auf dem **Catalog Objects in Library**-Screen, siehe ***Select Options***. Mögliche *options* sind:	
	CC	Condition Code wird zurückgegeben.
	NOREN	Keine automatische Neunummerierung von Sourcecode-Zeilen.
	KEEP	Ergebnisliste wird aufbewahrt.
	NOSCROLL	Online: Keine laufende Anzeige von Verarbeitungsstatus-Informationen. Batch: Ausgabe derjenigen Module, die einen Fehler verursacht haben.
	NOREPORT	Keine Fehlerliste.
	FULL	Erweiterte Fehlerliste.
	EL <*text-member*> [R]	
	EL <*text-member*>	Ausgabe der Fehlerliste in ein Natural Text Member.
	R	Wenn ein Text Member existiert, ist der EL-Parameter funktionslos, wenn kein R (Replace) nach <*text-member*> angegeben wird.
	Anmerkung: Bei Angabe von NOREPORT *und* NOSCROLL, gilt automatisch auch KEEP.	
text-name	Entspricht der Angabe eines Text-Namens im **Catalog Objects from**-Feld des **Catalog Objects in Library**-Schirms, siehe ***Catalog Objects from/to***.	

Beispiele:

Um nur die Objekte in Source- und Objektform zu speichern, für die schon Objektmodule vorhanden sind

- Geben Sie folgendes Kommando ein:

```
CATALL *
STOW KEEP CC NOREN
```

Dieses Kommando enthält ein implizites RECAT und bewirkt dasselbe wie folgendes Kommando:

```
CATALL * RECAT STOW KEEP CC NOREN
```

Um alle Objekte in Source- und Objektform zu speichern

- Geben Sie folgendes Kommando ein:

```
CATALL * ALL STOW KEEP CC
NOREN
```

Anmerkung: Die einzelnen Bestandteile des Kommandos müssen durch ein Leerzeichen oder durch das Eingabebegrenzungszeichen (wie mit dem Session-Parameter ID festgelegt) voneinander getrennt werden.

8 CATALOG

CATALOG [*object-name* [*library-id*]]

Verwandte Kommandos: SAVE | STOW | UNCATALOG.

Mit dem Systemkommando CATALOG kompilieren Sie das im Arbeitsbereich eines Editors befindliche Source-Objekt und speichern nach erfolgreicher Prüfung das resultierende Objektmodul in der Natural-Systemdatei.

Siehe auch:

Natural-Compiler in Natural *System-Architektur*
Namenskonventionen für Objekte in *Natural benutzen*

Wichtig: Es kann sein, dass Sie das CATALOG-Kommando nicht verwenden können, weil der Profilparameter RECAT auf ON gesetzt ist oder Natural Security-Einschränkungen in Kraft sind. In diesem Fall benutzen Sie stattdessen das Systemkommando STOW, um zu gewährleisten, dass Source- und Objektcode zusammenpassen.

CATALOG	Falls Sie keinen *object-name* angeben, wird das Programm in der aktuellen Library unter dem Namen des Source-Objekts gespeichert, das zuletzt in den Arbeitsbereich gelesen wurde (z.B. mit einem EDIT-, READ- oder RUN-Kommando). Eventuell vorhandener Objektcode wird ersetzt.
CATALOG *object-name*	Ein neues Objekt wird angelegt. Als *object-name* geben Sie den Namen an, unter dem das neue Objekt katalogisiert werden soll. Es wird in der aktuellen Library gespeichert. Falls das Objekt schon vorhanden ist, wird das Kommando zurückgewiesen.
CATALOG *object-name* *library-id*	Wollen Sie das neue Objekt in einer anderen Library katalogisieren, müssen Sie zusätzlich zum Objektnamen die *library-id* der gewünschten Library angeben.

Anmerkung: Falls im Parametermodul eine ungültige Systemdatei `FDIC` angegeben ist, erscheint eine entsprechende Natural-Fehlermeldung, wenn das `CATALOG`-Kommando abgesetzt wird.

9 CHECK

CHECK

Dieses Kommando dient dazu, den gerade im Arbeitsbereich des Programm-Editors befindlichen Sourcecode auf Syntaxfehler zu überprüfen.

Siehe auch:

Natural-Compiler in Natural *System-Architektur*
Namenskonventionen für Objekte in *Natural benutzen*

Modus	Reaktion
Online	Die Syntaxprüfung wird abgebrochen, sobald ein Syntaxfehler entdeckt wird, und die fehlerhafte Sourcecode-Zeile wird im Programm-Editor mit E markiert.
Batch	Das Programm wird vollständig auf Syntaxfehler untersucht. Alle Fehler werden in der Ausgabeliste mit ausgegeben.

Anmerkung: Die Syntaxprüfung erfolgt auch im Rahmen der Ausführung der Kommandos RUN, STOW, CATALOG und CATALL.

10 CLEAR

CLEAR

Mit diesem Kommando löschen Sie den Inhalt des Editor-Arbeitsbereichs. Falls Sie ein neues Objekt erstellen wollen, und es befindet sich noch ein anderes Source-Objekt im Arbeitsbereich, empfiehlt es sich, den Arbeitsbereich mit diesem Kommando zu löschen.

11 CMS

Dieses Kommando gilt nur in VM/CMS-Umgebungen. Es ermöglicht die Ausführung von CMS-Kommandos aus Natural heraus.

```
CMS CMS-command
```

Das anstelle von *CMS-command* angegebene Kommando wird dem CMS-Kommandoprozessor übergeben, der das Kommando ebenso ausführt wie im normalen, interaktiven CMS-Kommandomodus.

Weitere Informationen zu CMS siehe *Natural under VM/CMS* (in der *Operations*-Dokumentation).

12 COMPOPT

```
COMPOPT [option=value ...]
```

Mit diesem Kommando können Sie verschiedene Kompilierungsoptionen setzen. Die Optionen werden wirksam, wenn ein Natural-Programmierobjekt kompiliert wird.

Wenn Sie nur das Kommando COMPOPT eingeben, wird ein Schirm angezeigt, auf dem Sie die unten beschriebenen Optionen ein- bzw. ausschalten können.

Die Standardwerte für die einzelnen Optionen werden mit dem Makro NTCMPO im Natural-Parametermodul bzw. dem entsprechenden Profilparameter CMPO gesetzt. Wenn Sie die Library wechseln, werden die COMPOPT-Optionen wieder auf ihre Standardwerte zurückgesetzt.

Syntax-Erklärung

COMPOPT	Wenn Sie nur das Kommando COMPOPT eingeben, wird der **Compilations Options**-Schirm angezeigt, auf dem Sie die unten beschriebenen Optionen ein- bzw. ausschalten können. Die dort vorhandenen Schlüsselwörter werden nachfolgend beschrieben.
COMPOPT *option=value*	Die Schlüsselwörter für die einzelnen Optionen werden nachfolgend beschrieben. Die einer Compiler-Option zugewiesene Einstellung bleibt solange gültig, bis Sie das nächste LOGON-Kommando absetzen, um in eine andere Library zu wechseln. Beim LOGON werden wieder die mit dem Parametermakro NTCMPO und/oder dem Profilparameter CMPO eingestellten Standardeinstellungen gültig.

Compiler-Schlüsselwortparameter angeben

Compiler-Schlüsselwortparameter können Sie auf verschiedenen Ebenen angeben:

1. **Als Standardeinstellungen**

   Die Standardeinstellungen der einzelnen Compiler-Schlüsselwortparameter werden mit dem Parameter-Makro NTCMPO vorgenommen. Diese Einstellungen werden im Natural-Parametermodul NATPARM gespeichert.

2. **Zu Beginn einer Natural-Session**

   Zu Beginn einer Natural-Session können Sie die Compiler-Schlüsselwortparameter überschreiben, indem Sie die Werte im entsprechenden Profilparameter CMPO setzen.

3. **Während einer aktiven Natural-Session**

   Während einer aktiven Natural-Session haben Sie zwei Möglichkeiten, die Compiler-Parameterwerte mit dem Systemkommando COMPOPT zu ändern: Entweder direkt mittels Kommando-

zuordnung (`COMPOPT option=value`) oder indem Sie das Kommando `COMPOPT` ohne Optionen absetzen, woraufhin der Schirm **Compilation Options** erscheint.

Bei einem `LOGON` auf eine andere Library, werden die mit dem Makro `NTCMPO` und/oder dem Profilparameter `CMPO` vorgenommenen Standardeinstellungen (siehe Punkt 1) wieder hergestellt. Beispiel:

```
OPTIONS KCHECK=ON
DEFINE DATA LOCAL
1 #A (A25) INIT <'Hello World'>
END-DEFINE
WRITE #A
END
```

4. **In einem Natural-Programmierobjekt**

   In einem Natural-Programmierobjekt (z.B. Programm, Subprogramm) können Sie Compiler-Parameter mit dem `OPTIONS`-Statement setzen. Beispiel:

```
OPTIONS KCHECK=ON
WRITE 'Hello World'
END
```

   Die in einem `OPTIONS`-Statement angegebenen Compiler-Optionen betreffen nur die Kompilierung dieses Programmierobjekts, sie aktualisieren jedoch nicht die mit dem Systemkommando `COMPOPT` gesetzten Optionen.

Allgemeine Compiler-Optionen

- KCHECK - Keyword Checking
- PCHECK - Parameter Checking for CALLNAT Statements
- DBSHORT - Interpretation of Database Short Field Names
- PSIGNF - Internal Representation of Positive Sign of Packed Numbers
- TSENABL - Applicability of TS Profile Parameter
- GFID - Generation of Global Format IDs
- LOWSRCE - Allow Lower-Case Source
- TQMARK - Translate Quotation Mark
- THSEP - Dynamic Thousands Separator
- CPAGE - Code Page Support for Alphanumeric Constants
- DB2ARRY - Support DB2 Arrays in SQL SELECT and INSERT Statements

- CHKRULE - Validierung von INCDIR-Statements in Maps

Diese Optionen entsprechen den gleichnamigen Schlüsselwortparametern des Profilparameters CMPO bzw. des Parameter-Makros NTCMPO.

KCHECK - Keyword Checking

Prüfung auf Schlüsselwörter.

ON	Felddeklarationen in einem Programmierobjekt werden gegen einen Satz kritischer Natural-Schlüsselwörter geprüft. Falls ein Variablenname mit einem dieser Schlüsselwörter übereinstimmt, wird ein Syntaxfehler ausgegeben, wenn das Programmierobjekt geprüft oder katalogisiert wird.
OFF	Es erfolgt keine solche Prüfung. Dies ist der Standardwert.

Der Abschnitt *Prüfung der für Natural reservierten Schlüsselwörter durchführen* im *Leitfaden zur Programmierung* enthält eine Liste der Natural-Schlüsselwörter, die von der KCHECK-Option abgeprüft werden.

Der Abschnitt *Alphabetische Liste der für Natural reservierten Schlüsselwörter* im *Leitfaden zur Programmierung* enthält eine Übersicht über alle Natural-Schlüsselwörter und reservierten Wörter.

PCHECK - Parameter Checking for CALLNAT Statements

Prüfung der Parameter bei CALLNAT-Statements.

ON	Der Compiler prüft Anzahl, Format, Länge und Array-Index-Grenzen der Parameter, die in einem Objekt aufrufenden Statement (zum Beispiel CALLNAT, PERFORM, INPUT USING MAP, PROCESS PAGE USING, Helproutine-Aufruf) angegeben sind. Darüber hinaus wird bei der Parameterprüfung das OPTIONAL-Feature des DEFINE DATA PARAMETER-Statements berücksichtigt. Die Parameterprüfung basiert auf dem Vergleich der Parameter des Objekt aufrufenden Statements mit den DEFINE DATA PARAMETER-Definitionen in dem aufzurufenden Subprogramm. Dazu ist folgendes erforderlich: ■ Der Name des aufzurufenden Subprogramms muss als alphanumerische Konstante (nicht als alphanumerische Variable) definiert sein. ■ Das aufzurufende Subprogramm muss als katalogisiertes Objekt zur Verfügung stehen. Andernfalls hat die Einstellung PCHECK=ON keine Auswirkung. **Probleme bei der Benutzung des CATALL-Kammandos mit PCHECK=ON** Wenn ein CATALL-Kommando zusammen mit der Einstellung PCHECK=ON benutzt wird, ist folgendes zu beachten:

	Wenn ein CATALL-Vorgang aufgerufen wird, hängt die Reihenfolge, in der die Programmierobjekte kompiliert werden, in erster Linie vom Objekttyp und in zweiter Linie vom alphabetischen Namen des Objekts ab. Die Reihenfolge der Objekttypen beim Kompilieren ist: GDAs, LDAs/PDAs, externe Subroutinen, Subprogramme, Helproutinen, Maps/Adapter, Programme/Klassen. Bei Objekten desselben Typs bestimmt die alphabetische Reihenfolge der Namen die Abfolge, in der sie katalogisiert werden. Wie zuvor erwähnt werden die Parameter des des Objekt aufrufenden Statements gegen die kompilierte Form des aufgerufenen Subprogramms geprüft. Wenn das rufende Objekt (das momentan kompiliert wird und das Objekt aufrufende Statements enthält) vor dem aufgerufenen Objekt katalogisiert wird, kann das PCHECK-Ergebnis falsch sein, falls die Parameter im CALLNAT-Statement und im aufgerufenen Subprogramm geändert wurden. Das hat zur Folge, dass das *neue* Parameterlayout im Objekt aufrufenden Statement mit dem *alten* Parameterlayout im DEFINE DATA PARAMETER-Statement des aufgerufenen Subprogramms verglichen wird. Lösung: ■ Setzen Sie die Compiler-Option auf PCHECK=OFF. ■ Führen Sie mit dem CATALL-Kommando eine generelle Kompilierung in der gesamten Library durch oder, wenn ein einzelnes oder nur wenige Objekte geändert wurden, führen Sie eine separate Kompilierung dieser Objekte durch. ■ Setzen Sie die Compiler-Option auf PCHECK=ON. ■ Führen Sie mit der Funktion PCHECK des CATALL-Kommandos eine generelle Kompilierung in der gesamten Library durch.
OFF	Es erfolgt keine solche Prüfung. Dies ist der Standardwert.

DBSHORT - Interpretation of Database Short Field Names

Interpretation von Datenbankfeld-Namen in Programmierobjekten.

Ein in einem DDM definiertes Datenbankfeld wird durch zwei Namen beschrieben:

- durch den Kurznamen mit einer Länge von zwei Zeichen, der von Natural für die Kommunikation mit der Datenbank (insbesondere mit Adabas) verwendet wird;
- durch den Langnamen mit einer Länge von 3 bis 32 Zeichen (1 bis 32 Zeichen, wenn die darunter liegende Datenbank DB2/SQL ist), der zum Referenzieren des Feldes im Natural-Programmcode verwendet werden soll.

Unter speziellen Bedingungen können Sie in einem Natural-Programm ein Datenbankfeld mit seinem Kurznamen statt mit dem Langnamen referenzieren. Dies gilt, falls Sie Natural im Reporting Mode ohne Natural Security benutzen und falls das für den Datenbankzugriff verwendete Statement statt einer Referenz auf ein View eine Referenz auf ein DDM enthält.

Die Entscheidung, ob ein Feldname als eine Kurznamen-Referenz betrachtet wird, erfolgt abhängig von der Länge des Namens. Wenn die Feldkennung (Identifier) aus zwei Zeichen besteht, wird davon ausgegangen, dass eine Kurznamen-Referenz vorliegt; ein Feldname mit einer anderen Länge wird als Langnamen-Referenz betrachtet. Diese standardmäßige Interpretation können Sie zusätzlich beeinflussen, indem Sie die Compiler-Option `DBSHORT` auf `ON` oder `OFF` setzen:

ON	Die Verwendung eines Kurznamens zum Referenzieren eines Datenbankfeldes ist erlaubt, jedoch wird die Verwendung eines Datenbank-Kurznamens *nicht* generell gestattet (selbst wenn `DBSHORT=ON`) ■ für die Definition eines Feldes, wenn ein View erstellt wird; ■ wenn im Programmcode ein View verwendet wird; ■ wenn im Programmcode ein `DEFINE DATA LOCAL`-Statement zur Definition von Variablen verwendet wird; ■ wenn Natural Security aktiv ist. Dies ist der Standardwert.
OFF	Ein Datenbankfeld kann nur über seinen Langnamen referenziert werden. Jede Datenbankfeldkennung wird unabhängig von ihrer Länge als Langnamen-Referenz betrachtet. Wenn ein zwei Zeichen langer Name geliefert wird, der nur als Kurzname und nicht als Langname gefunden werden kann, wird bei der Kompilierung ein Syntaxfehler NAT0981 ausgegeben. Dies ermöglicht die Verwendung von Langnamen, die in einer DDM mit einer Kennungslänge von 2 Byte definiert wurden. Diese Option ist wichtig, wenn die darunter liegende Datenbank, auf die Sie mit diesem DDM zugreifen wollen, vom Typ SQL (DB2) ist und wenn dort Tabellenspalten mit einem zwei Zeichen langen Namen existieren. Dagegen wird bei allen anderen Datenbanktypen (zum Beispiel Adabas) jeder Versuch, ein Langnamenfeld mit einer Namenslänge von 2 Bytes zu definieren bei der DDM-Generierung zurückgewiesen. Darüber hinaus wird das Programm, wenn keine Kurznamenreferenzen verwendet werden (was durch `DBSHORT=OFF` erzwungen werden kann), unabhängig davon, ob es unter Natural Security kompiliert wird oder nicht.

Beispiele:

Gegeben sei die folgende Datenbankfelddefinition in dem DDM `EMPLOYEES`:

Kurzname	Langname
AA	PERSONNEL-ID

Beispiel 1:

```
OPTIONS DBSHORT=ON
READ EMPLOYEES
  DISPLAY AA      /* Datenbankfeldkurzname AA ist erlaubt
END
```

Beispiel 2:

```
OPTIONS DBSHORT=OFF
READ EMPLOYEES
  DISPLAY AA      /* Syntaxfehler NAT0981, weil DBSHORT=OFF
END
```

Beispiel 3:

```
OPTIONS DBSHORT=ON
DEFINE DATA LOCAL
1 V1 VIEW OF EMPLOYEES
  2  PERSONNEL-ID
END-DEFINE
READ V1 BY PERSONNEL-ID
  DISPLAY AA     /* Syntaxfehler NAT0981, weil PERSONNEL-ID im View definiert ist;
                 /* (selbst wenn DBSHORT=ON)
END-READ
END
```

PSIGNF - Internal Representation of Positive Sign of Packed Numbers

Interne Darstellung des positiven Vorzeichens von gepackten Zahlen.

ON	Das positive Vorzeichen einer gepackten Zahl wird intern als H'F' dargestellt. Dies ist der Standardwert.
OFF	Das positive Vorzeichen einer gepackten Zahl wird intern als H'C' dargestellt.

TSENABL - Applicability of TS Profile Parameter

Diese Option bestimmt, ob der Profilparameter `TS` für Natural-System-Libraries (d.h. für Libraries, deren Namen mit `SYS` beginnt, außer `SYSTEM`) gilt oder auch für alle Benutzer-Libraries.

Natural-Objekte, die mit der Einstellung `TSENABL=ON` katalogisiert werden, bestimmen die Verwendung des `TS`-Parameters auch dann, wenn sie sich nicht in einer System-Library befinden.

ON	Der Profilparameter `TS` gilt für alle Libraries.
OFF	Der Profilparameter `TS` gilt nur für Natural-System-Libraries. Dies ist der Standardwert.

GFID - Generation of Global Format IDs

Mit dieser Option können Sie Naturals interne Generierung von Global Format IDs steuern, um damit das Leistungsverhalten von Adabas bei der Wiederverwendbarkeit von Format-Buffer-Übersetzungen zu beeinflussen.

ON	Global Format IDs werden für alle Views generiert. Dies ist der Standardwert.
VID	Global Format IDs werden nur für in Local/Global Data Areas definierte Views generiert, aber nicht für innerhalb von Programmen definierte Views.
OFF	Es werden keine Global Format IDs generiert.

Weitere Informationen zu Global Format IDs siehe Adabas-Dokumentation.

Regeln für die Generierung von GLOBAL FORMAT-IDs in Natural

- **Für Natural-Nukleus-interne Systemdateiaufrufe:**

```
GFID=abccddee
```

dabei steht	für
a	x'F9'
b	x'22' oder x'21' in Abhängigkeit vom DB-Statement
cc	physische Datenbanknummer (2 Bytes)
dd	physische Dateinummer (2 Bytes)
ee	zur Laufzeit generierte Nummer (2 Bytes)

- **Für Benutzerprogramme oder Natural Utilities:**
  - `GFID=abbbbbbc`

Für Dateinummern kleiner als oder gleich 255 und Adabas Version kleiner als 6.2 (siehe NTDB-Makro).

dabei steht	für
a	x'F8' oder x'F7' oder x'F6'
bbbbbb	Bytes 1-6 des STOD-Werts
c	physische Dateinummer

- GFID=*axbbbbbc*

Für Dateinummern über 255 und Adabas Version kleiner als 6.2.

dabei steht	für
a	x'F8' oder x'F7' oder x'F6'
x	physische Dateinummer - high order byte
bbbbb	Bytes 2-6 des STOD-Werts
c	physische Dateinummer - low order byte

- GFID=*abbbbbb*

Für Adabas Version 6.2 oder höher.

dabei steht	für
a	x'F8' oder x'F7' oder x'F6' wobei: F6=UPDATE SAME F7=HISTOGRAM F8=alle übrigen
bbbbbbb	Bytes 1-7 des STOD-Werts

Anmerkung: STOD ist der Rückmeldewert der Store Clock Machine Instruction (STCK).

LOWSRCE - Allow Lower-Case Source

Diese Option unterstützt die Verwendung von Programmen in Sourceform mit Kleinschreibung oder gemischter Schreibweise auf Großrechnerplattformen und erleichtert so die Übertragung von derartigen Programmen von anderen Plattformen in eine Großrechnerumgebung.

ON	Gestattet jede Art von Zeichen in Klein-/Großschreibung in Source-Programmen.
OFF	Gestattet nur Großschreibung. Das bedeutet, dass Schlüsselwörter, Variablennamen und Identifier in Großbuchstaben definiert werden müssen. Dies ist der Standardwert.

Wenn Sie Zeichen in Kleinschreibung mit `LOWSRCE=ON` verwenden, müssen Sie folgendes berücksichsichtigen:

- Die Syntaxregeln für Variablennamen erlauben es, ab der zweiten Stelle Zeichen in Kleinschreibung zu benutzen. Sie können deshalb zwei Variablen gleichen Namens definieren, indem Sie die eine in Kleinschreibung und die andere in Großschreibung angeben, zum Beispiel:

  ```
  DEFINE DATA LOCAL
  1 #Vari (A20)
  1 #VARI (A20)
  ```

  Mit `LOWSRCE=OFF` werden die im Beispiel angegebenen Variablen als zwei verschiedene Variablen behandelt.

  Mit `LOWSRCE=ON` unterscheidet der Compiler nicht zwischen Klein- und Großschreibung. Dies führt zu einem Syntaxfehler, weil eine doppelte Namensvergabe bei Variablen nicht erlaubt ist.

- Bei Verwendung des Session-Parameters `EM` (Edit Mask) in einem I/O-Statement oder in einem `MOVE EDITED`-Statement gibt es Zeichen, die das Daten-Layout, das einer Variablen (`EM`-Steuerzeichen) zugeordnet ist, und Zeichen, die Textfragmente in das Datenfeld einfügen.

  Beispiel:

  ```
  #VARI :='1234567890'
  WRITE #VARI (EM=XXXXXxxXXXXX)
  ```

  Bei `LOWSRCE=OFF` ist die Ausgabe `12345xx67890`, weil bei Variablen im Alpha-Format nur X-, H- und Zirkumflex-(^)-Zeichen in Großschreibung verwendet werden können.

  Bei `LOWSRCE=ON` ist die Ausgabe `1234567890`, weil das Zeichen x wie der Großbuchstabe C behandelt und deshalb als `EM`-Steuerzeichen für dieses Feldformat interpretiert wird. Um dieses Problem zu vermeiden, sollten Sie Textkonstantenfragmente in Apostrophe (') einschließen.

Beispiel:

```
WRITE #VARI(EM=XXXXX'xx'XXXXX)
```

Das Textfragment wird, unabhängig von den LOWSRCE-Einstellungen, *nicht* als ein EM-Steuerzeichen betrachtet.

- Da bei der Einstellung LOWSRCE=ON alle Variablennamen in Großbuchstaben umgesetzt werden, ist die Anzeige von Variablennamen in I/O-Statements (INPUT, WRITE oder DISPLAY) unterschiedlich.

Beispiel:

```
 MOVE 'ABC' to #Vari
DISPLAY #Vari
```

Bei LOWSRCE=OFF ist die Ausgabe:

```
#Vari -------------------- ABC
```

Bei LOWSRCE=ON ist die Ausgabe:

```
#VARI -------------------- ABC
```

TQMARK - Translate Quotation Mark

Mit dieser Option können Sie Anführungszeichen (") in Apostrophe (') umsetzen.

ON	Jedes Anführungszeichen (") in einer Textkonstante wird als Apostroph (') ausgegeben. Dies ist der Standardwert.
OFF	Anführungszeichen (") in einer Textkonstante werden nicht umgesetzt, sondern als Anführungszeichen beibehalten.

Beispiel:

```
RESET A (A5) A:= 'AB"CD'
WRITE '12"34' / A / A (EM=H(5))
END
```

Mit TQMARK ON sieht die Ausgabe so aus:

```
 12'34 AB'CD C1C27DC3C4
```

Mit TQMARK OFF sieht die Ausgabe so aus:

```
12"34 AB"CD C1C27FC3C4
```

THSEP - Dynamic Thousands Separator

Mit dieser Option können Sie beim Kompilieren die Verwendung des Tausender-Trennzeichens ermöglichen oder unterdrücken. Siehe auch Profil- und Session-Parameter THSEPCH sowie Abschnitt *Trennzeichen-Angaben standardisieren* (im *Leitfaden zur Programmierung*).

ON	Das Tausender-Trennzeichen wird verwendet. Jedes Tausender-Trennzeichen, das nicht Teil eines Zeichenketten-Literals ist, wird intern durch ein Steuerzeichen ersetzt.
OFF	Das Tausender-Trennzeichen wird nicht verwendet, d.h. der Compiler erzeugt kein Steuerzeichen für Tausender-Trennzeichen. Dies ist die Kompatibilitätseinstellung.

CPAGE - Code Page Support for Alphanumeric Constants

(Codepage-Unterstützung bei alphanumerischen Konstanten)

Mit der CPAGE-Option können Sie eine Konvertierungsroutine aktivieren, die, wenn das Objekt zur Laufzeit gestartet wird, alle alphanumerischen Konstanten von der Codepage, die bei der Kompilierung aktiv war, in die Codepage, die zur Laufzeit aktiv ist, umsetzt.

Siehe auch Abschnitt *CPAGE Compiler Option* in der Dokumentation *Unicode and Code Page Support*.

ON	Codepage-Unterstützung für alphanumerische Zeichenketten ist eingeschaltet.
OFF	Codepage-Unterstützung für alphanumerische Zeichenketten ist ausgeschaltet. Dies ist der Standardwert.

DB2ARRY - Support DB2 Arrays in SQL SELECT and INSERT Statements

(Unterstützung von DB2-Arrays in SQL SELECT- und INSERT-Statements)

Mit der Option DB2ARRY können Sie die Recherche und/oder das Einfügen in mehreren Reihen in DB2 mit nur einer Ausführung eines SQL SELECT- oder INSERT-Statements aktivieren. Dies ermöglicht die Angabe von Arrays als Zielfelder im SQL SELECT-Statement oder als Quellfelder im SQL INSERT-Statement. Wenn DB2ARRY auf ON gesetzt ist, ist es nicht mehr möglich, alphanumerische Natural-Arrays für DB2 VARCHAR/GRAPHIC-Spalten zu verwenden. Stattdessen müssen Sie lange alphanumerische Natural-Variablen verwenden.

ON	DB2-Array-Unterstützung ist eingeschaltet.
OFF	DB2-Array-Unterstützung ist ausgeschaltet. Dies ist der Standardwert.

CHKRULE - Validierung von INCDIR-Statements in Maps

Mit der Option `CHKRULE` können Sie eine während des Katalogisierungsvorgangs für Maps stattfindende Validierungsprüfung ein- bzw. ausschalten.

ON	Die `INCDIR`-Validierung ist eingeschaltet. Wenn die bzw. das im `INCDIR`-Kontrollstatement referenzierte Datei (DDM) bzw. Feld nicht existiert, wird zur Kompilierungszeit ein Syntaxfehler (NAT0721) ausgegeben. Beim Anlegen einer Map können Sie Felder einfügen, die schon in einem anderen existierenden Programmierobjekt definiert sind. Das funktioniert bei fast allen Arten von Objekten, in denen Sie Variablen definieren können, und ebenso bei DDMs. Wenn es sich bei dem eingefügten Feld um eine Datenbankvariable handelt, fügt der Map Editor automatisch (neben dem eingefügten Feld) ein `INCDIR`-Statement in den Map-Statement-Rumpf ein, um damit das Laden und die Übernahme einer Predict-Regel auszulösen, wenn die Map (per `STOW`-Befehl) kompiliert wird. Die Funktionsweise ist ähnlich dem Vorgang bei der Verarbeitung eines `INCLUDE`-Statements. Anstatt jedoch Sourcecode-Zeilen aus einem Copycode-Objekt zu übernehmen, werden sie in diesem Fall aus Predict übergeben. Als Suchschlüssel zum Auffinden der Regel(n) dienen der DDM-Name (der als der Feld-Name betrachtet wird) und der Feld-Name. Beide werden im `INCDIR`-Statement angegeben. Eine während der Kompilierung angeforderte `INCDIR`-Regel muss aber nicht zwangsläufig in Predict gefunden werden, weil für ihr Vorhandensein durchaus keine Notwendigkeit besteht. Das bedeutet, dass keine Fehlersituation vorliegt, wenn eine gesuchte Regel nicht gefunden wird. Bei Übernahme von Feldern aus einer DDM in eine Map, werden entsprechende `INCDIR`-Statements erzeugt, die den aktuelle DDM- und Feldnamen als „Suchschlüssel“ enthalten, mit dem eventuell existierende Regeln aus Predict abgerufen werden. Falls aber die DDM nach dem Kopiervorgang umbenannt wird, bleibt der alte (nicht mehr gültige) DDM-Name im `INCDIR`-Statement erhalten, was zur Folge hat, dass keine Regel geladen und der Programmierer darüber nicht in Kenntnis gesetzt wird. Eine solche Situation tritt aber nicht nur im Falle einer DDM-Umbenennung auf. Wahrscheinlicher ist, dass sie durch versehentliche Zuordnung einer falschen FDIC-Datei verursacht wird. In diesem Fall ist der DDM-Name zwar gültig, kann aber in der aktuellen Predict-Systemdatei nicht gefunden werden. Das Ergebnis ist das gleiche wie wenn die DDM überhaupt nicht existiert, das heißt, die erwartungsgemäß aus Predict hinzuzufügenden Verarbeitungsregeln werden nicht eingefügt.
OFF	Die `INCDIR`-Validierung ist ausgeschaltet. Dies ist der Standardwert.

Kompilierungsoptionen für Versionskompatibilität

Diese Optionen entsprechen den gleichnamigen Schlüsselwortparametern des Profilparameters `CMPO` bzw. des Parameter-Makros `NTCMPO`.

FINDMUN - Detect Inconsistent Comparison Logic in FIND Statements

Mit Natural Version 2.3 hat sich die Vergleichslogik für multiple Felder in der `WITH`-Klausel des `FIND`-Statements geändert.

Das hat zur Folge, dass Sie andere Ergebnisse erhalten, wenn Sie Version-2.3-Programme, die bestimmte Formen von `FIND`-Statements enthalten, unter Version 2.3 kompilieren. Mit dieser Option können Sie nach `FIND`-Statements suchen, deren `WITH`-Klausel multiple Felder in einer Weise verwendet, die nicht mehr mit der verbesserten Vergleichslogik von Version 3.1 konsistent ist.

ON	Fehler NAT0998 wird für jedes bei der Kompilierung entdeckte derartige `FIND`-Statement ausgegeben.
OFF	Es erfolgt keine Suche nach derartigen `FIND`-Statements. Dies ist der Standardwert.

Die Vergleichslogik für multiple Felder in der `WITH`-Klausel des `FIND`-Statement wurde ab Natural Version 2.3 so geändert, dass sie mit der Vergleichslogik bei anderen Statements (z.B. `IF`) konsistent ist.

Man kann vier verschiedene Formen des `FIND`-Statement unterscheiden. Dabei wird davon ausgegangen, dass das Feld `MU` in den folgenden Beispielen ein multiples Feld ist.

1. ```
   FIND
   XYZ-VIEW WITH MU = 'A'
   ```

   Bei Version 2.2 und höher liefert dieses Statement Datensätze zurück, in denen mindestens eine Ausprägung von `MU` den Wert `A` hat.

2. ```
   FIND XYZ-VIEW WITH
   MU NOT EQUAL 'A'
   ```

   Bei Version 2.2 liefert dieses Statement Datensätze zurück, in denen keine Ausprägung von `MU` den Wert `A` hat (wie bei Punkt 4). Bei Version 2.3 und höher liefert dieses Statement Datensätze zurück, in denen mindestens eine Ausprägung von `MU` nicht den Wert `A` hat.

3.
```
FIND XYZ-VIEW WITH NOT MU NOT EQUAL
'A'
```

   Bei Version 2.2 liefert dieses Statement Datensätze zurück, in denen *mindestens eine Ausprägung* von `MU` den Wert `A` hat (wie bei Punkt 1). Bei Version 2.3 und höher liefert dieses Statement Datensätze zurück, in denen *jede Ausprägung* von `MU` den Wert `A` hat.

4.
```
FIND XYZ-VIEW WITH NOT MU
= 'A'
```

   Bei Version 2.2 und höher liefert dieses Statement Datensätze zurück, in denen *keine Ausprägung* von `MU` den Wert `A` hat. Das bedeutet, wenn Sie vorhandene Version-2.2-Programme, die `FIND`-Statements der Formen 2 und 3 enthalten, unter der Version 2.3 kompilieren, liefern diese unterschiedliche Ergebnisse.

Wenn Sie die Option `FINDMUN=ON` benutzen, wird bei jedem beim Kompilieren festgestellten `FIND`-Statement der Form 2 oder 3 ein Fehler (NAT0998) ausgegeben.

Wenn Sie in diesen Fällen weiterhin dieselben Ergebnisse wie bei der Version 2.2 erhalten möchten, müssen Sie diese Statements wie folgt schreiben:

Bei Form 2:

```
FIND XYZ-VIEW WITH MU NOT EQUAL 'A'
```

ändern in

```
FIND XYZ-VIEW WITH NOT MU = 'A'
```

Bei Form 3:

```
FIND XYZ-VIEW WITH NOT MU NOT EQUAL 'A'
```

ändern in

```
FIND XYZ-VIEW WITH MU = 'A'
```

MASKCME - MASK Compatible with MOVE EDITED

Mit dieser Option können Sie die MASK-Option mit dem MOVE EDITED-Statement kompatibel machen.

ON	Der Bereich der gültigen Jahreswerte, die zu den YYYY-Maskenzeichen passen, ist 1582 bis 2699, damit die MASK-Option mit dem MOVE EDITED-Statement kompatibel wird. Wenn der Profilparameter MAXYEAR auf 9999 gesetzt ist, erstreckt sich der Bereich der gültigen Jahreswerte von 1582 bis 9999.
OFF	Der Bereich der gültigen Jahreswerte, die zu den YYYY-Maskenzeichen passen, ist 0000 - 2699. Dies ist der Standardwert. Wenn der Profilparameter MAXYEAR auf 9999 gesetzt ist, erstreckt sich der Bereich der gültigen Jahreswerte von 0000 bis 9999.

NMOVE22 - Assignment of Numeric Variables of Same Length and Precision

ON	Die Zuordnung von numerischen Variablen, bei denen Source und Target dieselbe Länge und Genauigkeit haben, erfolgt wie bei Natural Version 2.2.
OFF	Die Zuordnung von numerischen Variablen, bei denen Source und Target dieselbe Länge und Genauigkeit haben, erfolgt wie bei Natural Version 2.3 und höher, d.h., sie werden so verarbeitet, als ob Source und Target eine unterschiedliche Länge oder Genauigkeit hätten. Dies ist der Standardwert.

V41COMP - Disable New Version 4.2 Syntax

Wichtig: Diese Compiler-Option ist nur bei der Natural Version 4.2 vorhanden, um einen sanften Übergang zu ermöglichen. Sie wird in einem späteren Release im Anschluss an Natural Version 4.2 entfallen.

Einige der mit Natural Version 4.2 eingeführten Funktionen und Programmierungsmerkmale verursachen Probleme, wenn ein unter Version 4.1 entwickeltes und kompiliertes Programm neu kompiliert werden soll, um es in einer Version-4.1-Umgebung in Betrieb zu nehmen. Die betreffenden Funktionen und Merkmale sind **nachfolgend** aufgeführt.

Die Option V41COMP dient dazu, deratige Unverträglichkeiten zu finden und eine Fehlermeldung auszugeben, die durch einen Reason Code aussagt, warum die Neukompilierung nicht erfolgreich war.

Mögliche Werte sind:

ON	Wenn ein Programm unter Version 4.2 kompiliert wird, dann wird jeder Versuch, ein von Version 4.2, aber nicht von Version 4.1 unterstütztes Syntaxkonstrukt zu verwenden, zurückgewiesen, und es wird ein Syntaxfehler NAT0647 und ein entsprechender Reason Code ausgegeben (siehe **unten**).
OFF	Es erfolgt keine Prüfung auf Kompatibilität mit Version 4.1. Dies ist der Standardwert.

Kompilierungsrelevante Unterschiede zwischen Version 4.2 und 4.1

Die folgende Tabelle enthält eine Übersicht über kompilierungsrelevante Unterschiede zwischen Version 4.2 und 4.1 und zeigt den Reason Code, der ausgegeben wird, wenn eine inkompatible Syntax festgestellt wird:

Funktion oder Merkmal	Version 4.2	Version 4.1	Reason Code
Neues Format U (Unicode)	möglich	unbekannt	001
Array mit variabler Anzahl an Ausprägungen: X-array, zum Beispiel: `DEFINE DATA LOCAL 1 #ARR (A10/1:*)`	möglich	unbekannt	002
Mögliche Länge von alphanumerischen Konstanten und Literalen (Konstanten)	1 Byte - 1 GB	1 Byte - 253 Bytes (NAT0264)	003
Neue Kompilerparameter `THSEP` Tausender-Trennzeichen in Editiermasken `CPAGE` Codepage-Umsetzung für alphanumerische Konstante ermöglichen	möglich	unbekannt	004

Neue Statements
 	
`MOVE NORMALIZED`
`MOVE ENCODED`
`PARSE XML`
`REQUEST DOCUMENT`
`EXPAND / REDUCE / RESIZE ARRAY`
	möglich	unbekannt	005

Statement `SET GLOBALS` ■ Session-Parameter `CPCVERR=ON/OFF` ■ zulässig im Structured-Modus (`SM=ON`)	möglich	unbekannt	006
Neue Systemvariablen `*PARSE-COL`	möglich	unbekannt	007

Funktion oder Merkmal	Version 4.2	Version 4.1	Reason Code
*PARSE-LEVEL *PARSE-NAMESPACE-URI *PARSE-ROW *PARSE-TYPE *CODEPAGE *LOCALE *TYPE *CURRENT-UNIT *UBOUND *LBOUND			
Nicht benutzt	-	-	008

Länge und Typ von Source-Parametern, die mit INCLUDE-Statement geliefert werden
 	
Beispiel:
 	
`INCLUDE COPY01 'WRITE *LINE'`
`'WRITE *PROGRAM'`
	jede Länge und Format U (Unicode) erlaubt	nur alphanumerisch mit einer Länge von max. 80 Bytes	009

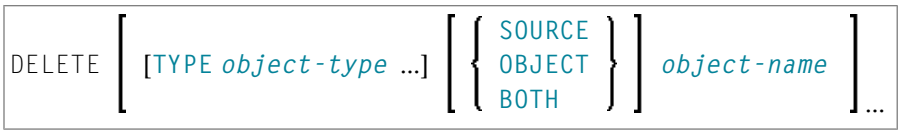
Definition eines Adabas LA-Feldes in einer Data View ■ mit Größe > 253 Bytes oder ■ des Typs DYNAMIC	möglich	unbekannt	010

13 CPINFO

Dieses Kommando dient zum Anzeigen der relevanten Natural Codepage-Einstellungen, z. B. Inhalt der Systemvariablen `*LOCALE`, `*CODEPAGE`, aktuelle Codepage des Arbeitsbereichs des Editors, aktuelle Einstellungen der relevanten Parameter, NATICU-Version, Unicode-Version, usw., und um die im Modul `NATCONFG` definierten Codepages anzuzeigen.

Weitere Informationen siehe *Unicode and Code Page Support*-Dokumentation.

14 DELETE



Das DELETE-Kommando dient dazu, Natural-Programmierobjekte aus der Natural-Systemdatei zu löschen.

Anmerkung: Das DELETE-Kommando hat keine Auswirkung auf das momentan im Arbeitsbereich des Editors befindliche Source-Programm .

Siehe auch *Namenskonventionen für Objekte* in der Dokumentation *Natural benutzen*.

Syntax-Erklärung

object-name	Als *object-name* geben Sie den Namen des Objekts bzw. die Namen der Objekte an, das bzw. die Sie löschen wollen. Zusätzlich können Sie angeben, ob der Sourcecode, das Objektmodul oder beide gelöscht werden sollen: SOURCE – Sourcecode OBJECT – Objektmodul BOTH – Sowohl Sourcecode als auch Objektmodul. Dies ist der Standardwert. Eine SOURCE/OBJECT/BOTH-Angabe gilt für alle nachfolgenden Objektnamen, d.h. solange, bis die nächste SOURCE/OBJECT/BOTH-Angabe auftritt. Um alle Objekte zu löschen, deren Namen mit einer bestimmten Zeichenkette beginnt, können Sie für *object-name* Stern-Notation (*) verwenden.
object-type	In Verbindung mit der Stern-Notation (*) für *object-name* können Sie auch einen Objekttyp (*object-type*) angeben, wenn Sie nur Objekte eines bestimmten Typs löschen möchten. Die möglichen Angaben für *object-type* entsprechen denen des Systemkommandos EDIT. Außerdem können Sie X (= Global, Local und Parameter Data Areas) und U (= Subprogramme, Subroutinen und Helproutinen) angeben. **Anmerkung:** Wenn Sie die vollständigen Namen bestimmter Objekte angeben, erübrigt sich die Angabe des Objekttyps.

Auswahlliste

Wenn Sie Stern-Notation (*) verwenden, erhalten Sie eine Auswahlliste, auf der Sie die Objekte, die Sie löschen möchten, markieren. Dabei können Sie für jedes einzelne Objekt bestimmen, ob sein Sourcecode, sein Objektmodul oder beides gelöscht werden soll, indem Sie das Objekt mit dem entsprechenden Buchstaben (`S`, `O` oder `B`) markieren.

Wenn Sie nur das `DELETE`-Kommando selbst eingeben, erhalten Sie ebenfalls eine Auswahlliste, und zwar aller in Ihrer aktuellen Library gespeicherten Objekte.

Schutz gegen versehentliches Löschen

Als Schutz gegen versehentliches Löschen erscheint automatisch ein Fenster, in dem Sie das Löschen eines Objekts durch Eingabe seines Namens bestätigen müssen.

Wenn Sie mehr als ein Objekt angegeben bzw. ausgewählt haben, erscheint zusätzlich ein Fenster, indem Sie angeben können, ob Sie das Löschen für jedes Objekt einzeln bestätigen möchten oder alle angegebenen/ausgewählten Objekte ohne Bestätigung gelöscht werden sollen.

Beispiele

Mit diesem Kommando löschen Sie drei Programmierobjekte namens `TOM`, `DICK` und `HARRY`:

```
DELETE TOM DICK HARRY
```

Mit diesem Kommando löschen Sie Source und Objektmodul des Programmierobjekts `JOHN`, die Sourcen der Programmierobjekte `PAUL` und `GEORGE` sowie das Objektmodul des Programmierobjekts `RINGO`:

```
DELETE JOHN SOURCE
PAUL GEORGE OBJECT RINGO
```

Mit diesem Kommando erhalten Sie eine Auswahlliste aller Programmierobjekte in der aktuellen Library:

```
DELETE
```

Mit diesem Kommando erhalten Sie eine Auswahlliste der Sourcen aller Maps in der aktuellen Library:

```
DELETE TYPE M SOURCE *
```

Mit diesem Kommando erhalten Sie eine Auswahlliste aller in Source- und/oder Objektform in der aktuellen Library gespeicherten Global, Local und Parameter Data Areas, deren Namen mit `D` anfangen:

```
DELETE TYPE GLA D*
```

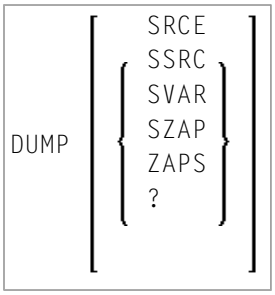
Mit diesem Kommando erhalten Sie eine Auswahlliste aller in Objektform in der aktuellen Library gespeicherten Programmierobjekte, deren Namen mit „YYZ“ anfangen:

```
DELETE OBJECT YYZ*
```

Mit diesem Kommando löschen Sie die Sourcen und Objektmodule der Maps `TOM` und `DICK`, die Source der Map `HARRY`, die Source des Programms `JOHN` und das Objektmodul des Programms `PAUL`:

```
DELETE TYPE M TOM DICK SOURCE HARRY TYPE
P JOHN OBJECT PAUL
```

15 DUMP



Das Systemkommando DUMP liefert Informationen, die das technische Personal der Software AG benötigt, um einen Fehler zu finden, der zum Abbruch (Abend) des Natural-Systems geführt hat. Übergeben Sie diese Informationen zwecks Fehlerdiagnose und -korrektur dem Software AG Technical Support.

DUMP	Zeigt Abbruchinformationen (Kernspeicher-Inhalt) an.
DUMP SRCE	Zeigt den Bestand an Source-Änderungen an, die pro Produkt eingespielt wurden.
DUMP SSRC	Zeigt den Bestand an speziellen Source-Änderungen an, die pro Produkt eingespielt wurden.
DUMP SVAR	Zeigt TP-Monitor-abhängige und betriebssystemspezifische Systemvariablen und zusätzliche Informationen an.
DUMP SZAP	Zeigt eine Liste aller eingespielten Spezial-Zaps an.
DUMP ZAPS	Zeigt eine Liste aller eingespielten Zaps an.
DUMP ?	Das DUMP-Kommando bietet noch eine Reihe weiterer Optionen (die auf den Hilfeschirmen erklärt sind, die Sie durch Eingabe eines Fragezeichens (?) auf dem DUMP-Menü erhalten). Falls es für die Fehlerdiagnose erforderlich ist, erfahren Sie vom Software AG Technical Support, wann und wie Sie diese Optionen einsetzen sollen.

16 EDIT

Mit dem Systemkommando `EDIT` rufen Sie einen der Natural-Editoren auf, um die Source-Form eines Natural-Programmierobjekts zu editieren.

Es gibt drei verschiedene Formen der Kommandosyntax. Diese sind nachfolgend in getrennten Abschnitten dokumentiert.

Verwandtes Kommando: `READ`

Siehe auch *Namenskonventionen für Objekte* in der Dokumentation *Natural benutzen*.

Syntax 1

```
EDIT [object-type] [object-name [library-id]]
```

object-type

Folgende Objekttypen können editiert werden:

```
( { CLASS      } )
( { 4          } )
  COPYCODE
  GLOBAL
  HELPROUTINE
  LOCAL
{ MAP          }
  PARAMETER
  PROGRAM
( { SUBPROGRAM } )
( { N          } )
  SUBROUTINE
( TEXT         )
```

Welcher Editor aufgerufen wird, hängt von dem zu editierenden Objekt ab:

- Ist das zu editierende Objekt eine Local Data Area, Global Data Area oder Parameter Data Area, wird der Data Area Editor aufgerufen.
- Ist das zu editierende Objekt eine Map oder eine Helproutine, die eine Map ist, wird der Map Editor aufgerufen.
- Ist das zu editierende Objekt eine Klasse, wird der Program Editor aufgerufen.

- Alle anderen Objektypen (PROGRAM, SUBPROGRAM, SUBROUTINE, HELPROUTINE, COPYCODE, TEXT, DESCRIPTION) werden im Programm-Editor editiert.

Anmerkung: Das Textobjekt DESCRIPTION ist eine Programmbeschreibung, die im Predict-Datendiktionär gespeichert ist und gepflegt wird; diese Objekte können nur editiert werden, wenn Predict installiert ist.

Die Objekttypen werden im *Leitfaden zur Programmierung* beschrieben. Die Editoren werden in der *Editors*-Dokumentation beschrieben.

Wenn Sie den Namen des Objekts, das Sie editieren möchten, angeben, brauchen Sie keinen Objekttyp anzugeben.

object-name

Mit dem Systemkommando EDIT geben Sie den Namen des Objekts an, das Sie editieren möchten. Die maximale Länge des Objektnamens beträgt 8 Zeichen.

Natural lädt dann das Objekt in den Arbeitsbereich des entsprechenden Editors, wo Sie es editieren können. Wenn Sie das Objekt anschließend unter demselben Namen speichern wollen, brauchen Sie bei einem anschließenden SAVE-, CATALOG- oder STOW-Kommando keinen Namen anzugeben.

Falls der Arbeitsbereich des Editors nicht leer ist, wird das angegebene Objekt in den entsprechenden Editor geladen.

Anmerkung: Bei EDIT DESCRIPTION muß der *object-name* dem betreffenden Natural-Member-Namen in der Predict-Programmdefinition entsprechen.

Falls Sie keinen *object-name* angeben und es befindet sich kein Objekt im Arbeitsbereich, erhalten Sie den leeren Programm-Editor-Schirm, in dem Sie ein Programm erstellen können.

library-id

Befindet sich das Objekt in einer anderen Library als der, in der Sie gerade arbeiten, so müssen Sie die *library-id* der Library angeben, in der das zu editierende Objekt enthalten ist.

Anmerkung: Eine *library-id*, die mit SYS beginnt, darf nicht angegeben werden (Ausnahme: SYSTEM).

Wenn Natural Security aktiv ist, können Sie keine *library-id* angeben, d.h. Sie können nur Objekte aus ihrer aktuellen Library editieren

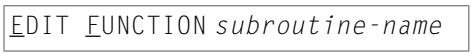
Syntax 2

Wenn Sie den Namen des Objekts, das Sie editieren möchten, nicht wissen, haben Sie mit dieser Form des EDIT-Kommandos die Möglichkeit, von einer Liste von Objekten das gewünschte Objekt auszuwählen.

EDIT *	Liefert Ihnen eine Liste aller Objekte, die in Ihrer aktuellen Library gespeichert sind.
EDIT *object-type* *****	Liefert Ihnen eine Liste aller Objekte des angegebenen Typs aus Ihrer aktuellen Library.

Um ein Objekt aus einem bestimmten Bereich von Objekten auszuwählen, können Sie Stern-Notation (*) und Wildcard-Notation (?) für den *object-name* verwenden, und zwar in der gleichen Weise wie beim Systemkommando LIST beschrieben.

Syntax 3



Mit dem Kommando EDIT FUNCTION können Sie eine Subroutine unter ihrem internen Namen (also dem Namen, unter dem sie aufgerufen wird, nicht dem Namen, unter dem das Objekt gespeichert ist, in dem sie enthalten ist) zum Editieren aufrufen.

Der *subroutine-name* darf maximal 32 Zeichen lang sein.

Beispiel:

```
DEFINE SUBROUTINE CHECK-PARAMETERS
...
END-SUBROUTINE
END
```

Angenommen, obige Subroutine ist unter dem Objektnamen CHCKSUB gespeichert, dann haben Sie folgende Möglichkeiten, um die Subroutine CHECK-PARAMETERS aufzurufen:

Entweder mit

```
EDIT S CHKSUB
```

oder mit

```
EDIT F CHECK-PARAMETERS
```

17 EDT

Es empfiehlt sich, statt des EDT-Kommandos das EDIT-Kommando zu verwenden.

```
EDT [object-name [library-id]]
```

Mit dem Systemkommando EDT rufen Sie den Natural-Programm-Editor zum zeilenorientieren Editieren auf, d.h. Sie können ein Natural-Objekt (Programm, Subprogramm, Subroutine, Helproutine oder Copycode) Zeile für Zeile editieren, hierbei aber jeweils nur eine Zeile zur Zeit editieren. Hierzu können Sie die unten aufgeführten Kommandos und PF-Tasten verwenden.

Mit dem Kommando .E beenden Sie das zeilenorientierte Editieren.

Syntax-Erklärung

object-name	Als *object-name* geben Sie den Namen des Objekts an, das Sie editieren möchten (maximal 8 Zeichen lang). Das Objekt wird in den Arbeitsbereich des Programm-Editors geladen, wo Sie es editieren können. Wenn Sie das Objekt anschließend unter demselben Namen speichern wollen, brauchen Sie bei einem SAVE-, CATALOG-, oder STOW-Kommando keinen Namen anzugeben. Wenn Sie keinen Objektnamen angeben, können Sie das gerade im Programm-Editor befindliche Objekt im EDT-Modus editieren; befindet sich kein Objekt im Editor, können Sie Zeile für Zeile ein neues Objekt erstellen und erhalten hierzu zunächst Zeile 0010. If you do not specify an object name and there is an object in the source program work area, the first lines of that object will be displayed.
library-id	Befindet sich das zu editierende Objekt in einer anderen Library als der, in der Sie gerade arbeiten, so müssen Sie die *library-id* dieser Library angeben. Eine Library–ID, die mit SYS beginnt, darf nicht angegeben werden (Ausnahme: SYSTEM). Wenn Natural Security aktiv ist, können Sie keine *library-id* angeben, d.h. Sie können nur Objekte aus Ihrer aktuellen Library editieren

EDT-Kommandos

Im EDT-Modus können Sie folgende Kommandos verwenden:

Command	Function
.B	Blättert zur letzten Zeile.
.C*nnnn*(*m*)	Kopiert *m* Zeilen, und zwar ab Zeile *nnnn*.
.C'*text*'(*m*)	Kopiert *m* Zeilen, und zwar ab der Zeile, die mit *text* beginnt.
.D	Löscht die Zeile.
.D(*n*)	Löscht die Zeile und die *n* minus 1 nachfolgenden Zeilen.
.E	Beendet das zeilenorientierte Editieren.
.I	Fügt eine Zeile ein.
.I(*program*)	Fügt das Objekt *program* ein.
.M*nnnn*	Verschiebt Zeile *nnnn*.
.M'*text*'(*m*)	Verschiebt *m* Zeilen, und zwar ab der Zeile, die mit *text* beginnt.
.R	Entspricht dem Systemkommando RENUMBER.
.S'*text*'	Sucht die Zeile mit der Zeichenkette *text*.
.T	Blättert zur ersten Zeile.
.*nnnn*	Blättert zu Zeile *nnnn*.
.+*n*	Blättert *n* Zeilen vor.
.-*n*	Blättert *n* Zeilen zurück.

EDT-Funktionstasten

Im EDT-Modus können Sie folgende Funktionstatsten benutzen:

Taste	Kommando	Funktion
PF1	.-18	Blättert 18 Zeilen zurück.
PF2	.T	Blättert zur ersten Zeile.
PF3	.B	Blättert zur letzten Zeile.
PF4	.+5	Blättert 5 Zeilen vor.
PF5	.+10	Blättert 10 Zeilen vor.
PF6	.+18	Blättert 18 Zeilen vor.
PF7	.R	RENUMBER-Kommando.
PF8	.I	Fügt Leerzeile ein.

Taste	Kommando	Funktion
PF9	.E	Beendet das zeilenorientierte Editieren.
PF10	.E,RUN	Beendet das zeilenorientierte Editieren und führt das Programm aus.
PF11	.E,SAVE,RUN	Beendet das zeilenorientierte Editieren, speichert das Programm in Sourceform und führt es aus.
PF12	.E,CAT,SAVE,EX	Beendet das zeilenorientierte Editieren, speichert das Programm in Source- und Objektform und führt es aus.

18 EXECUTE

```
{ EXECUTE  [REPEAT]     program-name      [library-id] }
{ program-name [parameter ...]                         }
```

Das Systemkommando `EXECUTE` dient dazu, ein Natural-Objektmodul des Typs Programm auszuführen.

Das Objektmodul muß in der Natural-Systemdatei katalogisiert (d.h. in Objektform gespeichert) oder in den Natural-Nukleus eingebunden sein.

Die Ausführung eines Objektmoduls hat keinen Einfluß auf die Source, die sich gerade im Editor-Arbeitsbereich befindet.

Syntax-Erklärung

EXECUTE	Das Schlüsselwort `EXECUTE` ist nicht erforderlich; es genügt, *program-name*, d.h. den Namen des auszuführenden Programms anzugeben. **Vorsicht:** Wenn Sie es in der Kommandozeile des Programm-Editors eingeben, können Sie das Systemkommando `EXECUTE` nicht mit `EX` abkürzen, da der Programm-Editor dies als das Editierkommando `EX` interpretieren würde.
REPEAT	Wenn das auszuführende Programm mehrere Ausgabeschirme erzeugt und Sie möchten, dass die Schirme unmittelbar nacheinander, d.h. ohne zwischengeschaltete Eingabezeilen, ausgegeben werden, verwenden Sie das Schlüsselwort `EXECUTE` zusammen mit dem Schlüsselwort `REPEAT`.
program-name	Der Name des Programms, das Sie ausführen möchten. Geben Sie keine Library-ID an, so kann das Programm nur ausgeführt werden, wenn es entweder in Ihrer aktuellen Library oder der aktuellen Steplib-Library (die Standard-Steplib ist `SYSTEM`) gespeichert ist.
library-id	Befindet sich das Objekt in einer anderen Library als der, in der Sie gerade arbeiten, so müssen Sie die Library-ID dieser Library angeben. Das Programm kann nur ausgeführt werden, wenn es auch tatsächlich in der angegebenen Library gespeichert ist. Eine Library-ID, die mit `SYS` beginnt, darf nicht angegeben werden (Ausnahme: `SYSTEM`). Wenn Natural Security aktiv ist, ist es nicht möglich, eine Library-ID anzugeben, d.h. ein Objekt aus einer anderen Library auszuführen.
parameter	Wenn Sie ein Programm ausführen, indem Sie den Programmnamen ohne das Schlüsselwort `EXECUTE` eingeben, haben Sie die Möglichkeit, Parameter an das Programm zu übergeben. Diese Parameter werden dann vom ersten `INPUT`-Statement des ausgeführten Programms gelesen. Sie können die Parameter als positionelle Parameter oder als Schlüsselwortparameter angeben, wobei die einzelnen Angaben durch Leerzeichen oder das (mit dem Session-Parameter `ID` bestimmte) Eingabe-Begrenzungszeichen voneinander getrennt werden müssen.

	Anmerkung: Die Parameterwerte werden immer in Großbuchstaben umgesetzt (unabhängig von dem Terminalkommando `%L` oder dem Profilparameter `LC=ON`).

Beispiele für das EXECUTE-Kommando

```
EXECUTE PROG1
```

```
EXECUTE PROG1 ULIB1
```

```
PROG1
```

```
PROG1 VALUE1 VALUE2 VALUE3
```

```
PROG1 VALUE1, VALUE2, VALUE3
```

```
PROG1 PARM1=VALUE1, PARM2=VALUE2, PARM3=VALUE3
```

```
PROG1 PARM3=VALUE3 PARM1=VALUE1 VALUE2
```

19 FIN

FIN

Das Systemkommando FIN dient dazu, eine Natural-Session zu beenden. Es gilt für Online- wie für Batch-Sessions.

Eine Batch-Session wird auch beendet, sobald in den Kommando-Eingabedaten eine „End-of-File“-Bedingung entdeckt wird.

20 GLOBALS

```
GLOBALS [parameter=value ...]
```

Mit dem Systemkommando GLOBALS können Sie Natural-Session-Parameter setzen.

Syntax-Erklärung

GLOBALS	Falls Sie das GLOBALS-Kommando ohne Parameter eingeben, erhalten Sie einen Schirm, der Ihnen die gegenwärtig gültigen Parameterwerte zeigt. Dort haben Sie auch die Möglichkeit, diese Werte zu ändern.
parameter	Die einzelnen Session-Parameter-Einstellungen können in beliebiger Reihenfolge angegeben werden; sie müssen jeweils durch ein Leerzeichen voneinander getrennt werden. Falls der Platz in der Kommandoeingabezeile nicht ausreicht, um alle gewünschten Parameter anzugeben, führen Sie gegebenenfalls mehrere GLOBALS-Kommandos nacheinander aus, siehe Beispiel.

Beispiel:

```
GLOBALS DC=, ID=.
```

Liste der Parameter

Die folgende Tabelle enthält eine Liste der Session-Parameter, die Sie mit dem Systemkommando GLOBALS angeben können.

Parameter	Funktion (Kurztext)
CC	Error Processing in Batch Mode
CF	Character for Terminal Commands
CPCVERR	Code Page Conversion Error
DC	Character for Decimal Point Notation
DFOUT	Date Format for Output
DFSTACK	Date Format for Stack
DFTITLE	Output Format of Date in Standard Report Title
DO	Display Order of Output Data
DU	Dump Generation
EJ	Page Eject
FCDP	Filler Character for Dynamically Protected Input Fields
FS	Default Format/Length Setting for User-Defined Variables

Parameter	Funktion (Kurztext)
IA	Input Assign Character
ID	Input Delimiter Character
IM	Input Mode
LE	Reaction when Limit for Processing Loop Exceeded
LS	Line Size
LT	Limit for Processing Loops
MT	Maximum CPU Time
NC	Use of Natural System Commands
OPF	Overwriting of Protected Fields by Helproutines
PD	Limit of Pages for `NATPAGE`
PM	Print Mode
PS	Page Size for Natural Reports
REINP	Issue Internal `REINPUT` Statement for Invalid Data
SA	Sound Terminal Alarm
SF	Spacing Factor
SL	Source Line Length
SM	Programming in Structured Mode
THSEPCH	Thousands Separator Character
TS	Translate Output from Programs in System Libraries
WH	Wait for Record in Hold Status
ZD	Zero-Division Check
ZP	Zero Printing

Zusammenhang zwischen GLOBALS, SET GLOBALS und anderen Statements

SET GLOBALS-Statement

Das Systemkommando `GLOBALS` und das Statement `SET GLOBALS` bieten dieselben Parameter und können beide in derselben Natural-Session verwendet werden.

Die mit einem `GLOBALS`-Kommando angegebenen Parameterwerte gelten solange, bis Sie sie mit einem neuen `GLOBALS`-Kommando überschreiben, die Session beenden oder ein Logon in eine andere Library ausführen.

Andere Statements, die die Session-Parameter-Einstellungen beeinflussen

Für ein einzelnes Programm oder Teile eines Programms können Sie zum Teil Parameterwerte angeben, die von den sessionweit gültigen abweichen, und zwar mittels `LIMIT`-, `EJECT`- und `FORMAT`-Statements und mittels Formatangaben, die Sie in `INPUT`-, `DISPLAY`-, `PRINT`- und `WRITE`-Statements machen.

Informationen zu diesen Statements finden Sie in der *Statements*-Dokumentation.

21 HELP

Mit dem Systemkommando HELP rufen Sie das Natural-Hilfesystem auf. Es liefert Informationen zu Natural-Statements, -Kommandos und -Fehlermeldungen.

HELP	Zeigt das Hilfemenü an.
HELP *statement*	Zeigt Informationen zu dem betreffenden Statement an.
HELP *command*	Zeigt Informationen zu dem betreffenden Kommando an.
HELP [NAT]*nnnn*	Wenn Sie HELP und eine (bis zu vierstellige) Nummer (wahlweise mit NAT davor) eingeben, erhalten Sie die Erklärung zu der betreffenden Fehlernummer, d.h. den Langtext der Natural-Systemfehlermeldung NAT*nnnn*.
HELP NAT	Zeigt Informationen zu allen Fehlermeldungen an.
HELP USER*nnnn*	Zeigt den Langtext der library-spezifischen Fehlermeldung Nummer *nnnn* aus der aktuellen Library an.
HELP USER*nnnn* **[***library-name***]**	Zeigt den Langtext der library-spezifischen Fehlermeldung Nummer *nnnn* aus der angegebenen Library an.
HELP USER	Zeigt den Langtext der library-spezifischen Fehlermeldung Nummer *nnnn* aus der aktuellen Library an.

HELP USER [*library-name***]**	Zeigt eine Auswahlliste *aller* library-spezifischen Meldungen der aktuellen Library an.
HELP ERROR	Zeigt den Langtext des zuletzt aufgetretenen Fehlers an.

22 INPL

```
INPL [R]
```

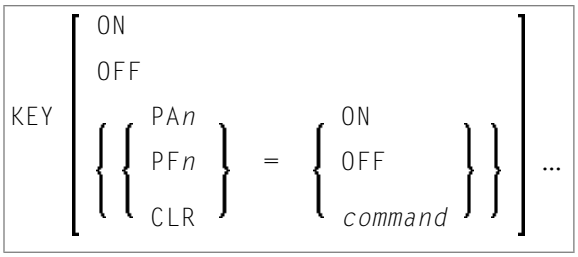
Mit dem INPL-Kommando rufen Sie die INPL-Utility auf. Diese Utility dient *nur* zum Laden von Software-AG-Installationsdatasets in die Systemdateien (wie in der INPL-Online-Hilfe und in den plattform-spezifischen Installationsschritten der Installationsdokumentation beschrieben).

Ansonsten verwenden Sie zum Laden von Objekten in die Systemdateien den Object Handler.

INPL	Wenn Sie das INPL-Kommando ohne Parameter eingeben, rufen Sie damit die INPL-Utility auf.
INPL R	Ruft die INPL-Utility-Funktion **Natural Security Recover** auf. Diese Option ist nur verfügbar, wenn Natural Security installiert ist. Damit stellen Sie den Urzustand der Zugriffsberechtigung auf die Natural Security-Library SYSSEC wieder her. Dadurch werden der User DBA, die Library SYSSEC sowie der Link zwischen beiden wieder so definiert, wie Sie nach der Installation ursprünglich definiert waren; gleichzeitig werden alle anderen Links nach SYSSEC gelöscht. Siehe auch *Inaccessible Security Profiles* im Abschnitt *Countersignatures* der *Natural Security*-Dokumentation.

Weitere Informationen siehe *INPL Utility* in der *Utilities*-Dokumentation.

23 KEY



Mit dem Systemkommando KEY können Sie bestimmten Funktionstasten auf der Tastatur Ihres Video-Terminals eine Funktion zuweisen oder eine zugewiesene Funktion ändern. Außerdem können Sie die zugewiesenen Funktionen deaktivieren und später wieder aktivieren.

Dies ist für folgende Funktionstasten möglich:

- PA1 bis PA3,
- PF1 bis PF24
- CLEAR bzw. LÖSCH

Als Funktion können Sie einer Taste folgendes zuweisen:

- ein Natural-Systemkommando,
- ein Natural-Terminalkommando,
- ein benutzerdefiniertes Kommando.

Natural führt das zugewiesene Kommando immer dann aus, wenn Sie die betreffende Taste im Kommando-Modus (NEXT-Modus) drücken.

Anmerkungen:

1. Die mit dem Systemkommando KEY zugewiesenen Funktionen sind völlig unabhängig von den Funktionen, die den Tasten mittels eines SET KEY-Statements in einem Programm zugewiesen werden.
2. Der Natural-Administrator hat außerdem die Möglichkeit, mit dem Profilparameter KEY bestimmten Tasten Funktionen zuzuweisen.
3. Dieses Kommando kann nicht im Batch-Betrieb ausgeführt werden.

Kommandos zuweisen

Wenn Sie nur das Kommando KEY (ohne Parameter) eingeben, erscheint der **Function-Key Assignments**-Schirm. Auf diesem Schirm können Sie den einzelnen Tasten Kommandos zuweisen, indem Sie sie in den Eingabefeldern eintragen.

Um einer Taste eine anderes Kommando zuzuweisen, überschreiben Sie den bestehenden Eintrag mit einem neuen Kommando.

Um eine Kommandozuweisung zu löschen, löschen Sie den Eintrag oder überschreiben ihn mit Leerzeichen.

Sie können einzelnen Tasten auch Kommandos zuweisen, indem Sie diese direkt mit dem KEY-Kommando angeben, und zwar in der Form KEY *key=command* (wobei *key* der Name der Taste und *command* der Name des Kommandos ist, das der Taste zugewiesen werden soll). Beispiel:

```
KEY PF1=CLEAR
```

Enthält das zugewiesene Kommando Leerzeichen, muß es in Apostrophen stehen. Beispiel:

```
PF13='UPDATE OFF'
```

Aktivieren/Deaktivieren aller Tasten - KEY ON/OFF

Mit dem Kommando KEY OFF/ON aktivieren/deaktivieren Sie alle zugewiesenen Kommandos.

KEY OFF	Wenn Sie eine Funktionstaste drücken, erhalten Sie von Natural eine entsprechende Meldung, dass die Taste nicht aktiv ist.
KEY ON	Mit dem Kommando KEY ON aktivieren Sie die mit KEY OFF außer Kraft gesetzten Kommandos wieder.

Sie können die Tasten auch aktivieren/deaktivieren, indem Sie oben rechts auf dem **Function-Key Assignments**-Schirm den Eintrag ON bzw. OFF im Feld **Activate Keys** überschreiben.

Anmerkung: Die CLEAR- bzw. LÖSCH-Taste kann nicht aktiviert bzw. deaktiviert werden. Falls ihr keine andere Funktion zugewiesen wird, hat sie die gleiche Funktion wie das Terminalkommando %%. Das Kommando KEY ON/OFF und das **Activate Keys**-Feld haben keine Auswirkung auf die CLEAR-Taste.

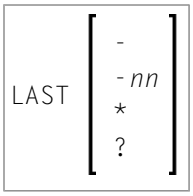
Aktivieren/Deaktivieren einzelner Tasten — KEY key=ON/OFF

Mit dem Kommando `KEY key=OFF/ON` setzen Sie das einer bestimmten Taste zugewiesene Kommando außer Kraft.

KEY *key*=OFF	Damit setzen Sie das einer bestimmten Taste (*key*) zugewiesene Kommando außer Kraft. Beispiel: `KEY PF24=OFF`
KEY *key*=ON	Damit reaktivieren Sie eine zuvor außer Kraft gesetzte Kommandozuweisung. Beispiel: `KEY PF24=ON`

Anmerkung: Das Kommando `KEY CLR=ON/OFF` ist nicht möglich (vgl. **Anmerkung** oben).

24 LAST



Mit dem Systemkommando LAST können Sie sich die zuletzt ausgeführten Systemkommandos anzeigen lassen.

Sobald das Kommando in die Kommandozeile oder NEXT-Zeile gestellt worden ist, können Sie es durch Drücken von EINGABE erneut ausführen. Sie können es auch überschreiben, bevor Sie es ausführen.

Das LAST-Kommando zeigt nur die Systemkommandos an, die Sie tatsächlich eingegeben haben; Kommandos, die Natural aufgrund eines von Ihnen eingegebenen Kommandos intern ausgeführt hat, werden von LAST nicht erfaßt.

LAST	Das zuletzt ausgeführte Kommando wird in die Kommandozeile oder NEXT-Zeile gestellt und kann ausgeführt werden.
LAST -	Das zuletzt ausgeführte Kommando wird in die Kommandozeile oder NEXT-Zeile gestellt und kann ausgeführt werden. Wenn Sie LAST - noch einmal eingeben, wird das vorletzte Kommando in die Kommandozeile oder NEXT-Zeile gestellt und kann ausgeführt werden. Durch wiederholte Eingabe von LAST - können Sie so Kommando für Kommando „zurückblättern“. **Anmerkung:** Anstatt es wiederholt von Hand einzugeben, können Sie LAST - über das Systemkommando KEY auch einer PF-Taste zuweisen.

LAST -*nn*	Natural kann sich maximal die 20 letzten Kommandos „merken“; *nn* darf also nicht größer als 20 sein. Das *nn*tletzte ausgeführte Kommando wird in die Kommandozeile oder NEXT-Zeile gestellt und kann ausgeführt werden.
LAST *	Es erscheint ein Fenster, in dem die 20 zuletzt abgesetzten Kommandos angezeigt werden. Mit PF8 und PF7 können Sie in der Liste vor und zurück blättern, wenn mehr als 10 Kommandos angezeigt werden: ■ Wenn Sie ein *einzelnes* Kommando erneut ausführen möchten, markieren Sie es mit dem Cursor und drücken Sie PF5 oder markieren Sie das Kommando mit einem Zeichen und drücken Sie ENTER. ■ Wenn Sie *mehrere* Kommandos erneut ausführen möchten, markieren Sie sie mit Zahlen in der Reihenfolge, in der sie ausgeführt werden sollen, und drücken Sie ENTER, die Kommandos werden dann wie durch die Zahlen vorgegeben in aufsteigender Reihenfolge ausgeführt.
LAST ?	Die Hilfe-Funktion für das LAST-Kommando wird aufgerufen.

25 LASTMSG

LASTMSG

Mit dem Systemkommando LASTMSG können Sie sich zusätzliche Informationen zu der zuletzt aufgetretenen Fehlersituation anzeigen lassen.

Wenn Natural eine Fehlermeldung ausgibt, kann es in manchen Fällen sein, dass es sich bei dem betreffenden Fehler nicht um den tatsächlichen Fehler handelt, sondern um einen Folgefehler eines anderen Fehlers (welcher wiederum ein Folgefehler eines anderen Fehlers sein kann usw.) Mit dem LASTMSG-Kommando können Sie in solchen Fällen den ausgegebenen Fehler bis zu dem Fehler zurückverfolgen, der die Fehlersituation ursprünglich verursacht hat.

Wenn Sie das Kommando LASTMSG eingeben, erhalten Sie — jeweils zu der zuletzt aufgetretenen Fehlersituation — die ausgegebene Fehlermeldung sowie alle vorherigen (nicht ausgegebenen) Fehlermeldungen, die zu diesem Fehler geführt haben.

Um Informationen zu dem entsprechenden Fehler anzuzeigen

- Markieren Sie eine dieser Meldungen mit dem Cursor und drücken Sie ENTER.

  Sie erhalten folgende Informationen zu dem betreffenden Fehler:

  - Fehlernummer;
  - Nummer der Zeile, in der der Fehler auftrat;
  - Name, Typ und Aufrufebene (Level) des Objekts, das den Fehler verursacht hat;
  - Name, Datenbank-ID und Dateinummer der Library, in der das Objekt enthalten ist;
  - Fehlerklasse (System = von Natural ausgegebener Fehler, User = von Benutzeranwendung ausgegebener Fehler);
  - Fehlertyp (Runtime, Syntax, Command Execution, Session Termination, Program Termination, Remote Procedure Call);

- Datum und Uhrzeit, wann der Fehler auftrat.

Anmerkung: Die Library SYSEXT enthält eine Programmierschnittstelle (API) USR2006N, über die Sie die von LASTMSG gelieferten Fehlerinformationen auch in Ihrer Natural-Anwendung erhalten können.

Natural Remote Procedure Call (RPC):

Bei einem Fehler auf dem Server werden folgende Fehlerinformationen nicht angezeigt: Datenbank-ID, Dateinummer, Datum und Uhrzeit.

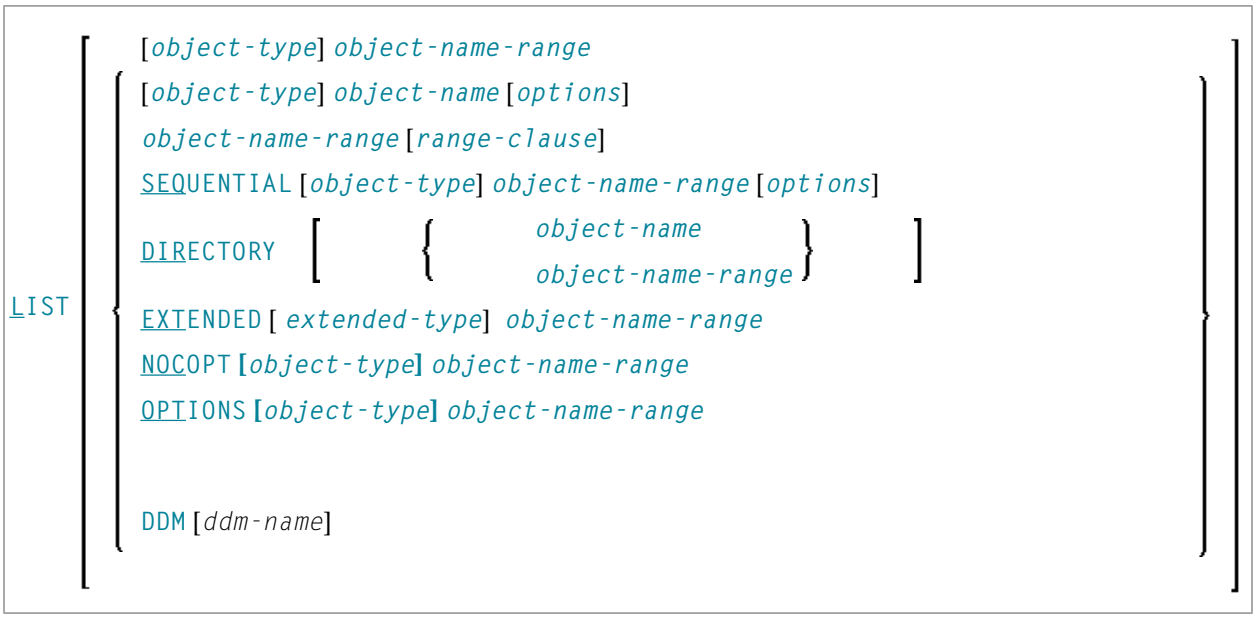
26 LIST

Mit dem Systemkommando LIST können Sie sich den Sourcecode eines einzelnen Objekts anzeigen oder mehrere in Ihrer aktuellen Library gespeichterte Objekte auflisten lassen. Die zahlreichen Möglichkeiten des LIST-Kommandos sind im Folgenden beschrieben.

Siehe auch separate Beschreibungen zu LIST XREF, LIST COUNT und LISTSQL.

Programmierschnittstellen (APIs): USR1054N, USR1055N, USR1056N, USR2018N, USR4216N. Siehe auch *SYSEXT - Natural Application Programming Interfaces* in der *Utilities*-Dokumentation.

Syntax-Übersicht

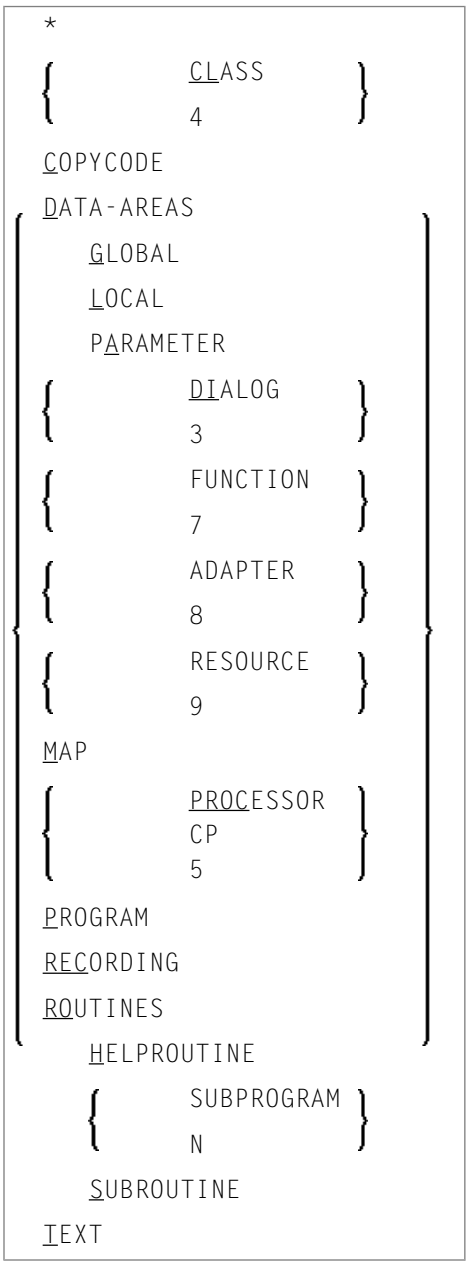


Anmerkungen:

1. Statt des Schlüsselworts DDM können Sie auch das Schlüsselwort VIEW (oder kurz V) verwenden.
2. Da das LIST-Kommando lange Zeilen mit bis zu 244 Zeichen anzeigen kann, sollten Sie die Zeilenlänge mit dem Profilparameter LS so groß wie möglich wählen. Wenn möglich, setzen Sie den Parameter LS=250.

object-type

Für *object-type* können Sie einen der nachfolgend aufgeführten Objekttypen angeben oder Stern-Notation (*) benutzen.



object-name

Für *object-name* können Sie den Namen eines Objekts (maximal 8 Zeichen; bei `LIST EXTENDED` 32 Zeichen).

object-name-range

Für *object-name-range* können Sie Stern-Notation (*) und Wildcard-Notation (?) verwenden:

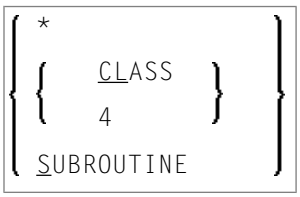
- Für eine Liste aller Objekte in der aktuellen Library geben Sie als *object-name-range* einen Stern (*) an, aber keinen *object-type*.
- Für eine Liste aller Objekte eines bestimmten Objekttyps geben Sie den gewünschten *object-type* sowie als *object-name-range* einen Stern (*) an.
- Für eine Liste eines bestimmten Bereichs von Objekten können Sie für den *object-name-range* Stern-Notation (*) und Wildcard-Notation (?) verwenden:
  - Stern-Notation ist die Möglichkeit, innerhalb des *object-name-range* einen Stern (*) anzugeben: der Stern steht für eine beliebige Zeichenkette beliebiger Länge.
  - Wildcard-Notation ist die Möglichkeit, innerhalb des *object-name-range* ein Fragezeichen (?) anzugeben: das Fragezeichen steht für ein beliebiges Zeichen.
- In einem *object-name-range* können Sie eine oder mehrere Stern- und Wildcard-Notationen miteinander kombinieren.
- Für eine vollständige Liste von Objekten ab einen bestimmten Startwert bzw. bis zu einem bestimmten Endwert können Sie die Größer-Zeichen-Notation (>) bzw. die Kleiner-Zeichen-Notation (<) verwenden.
- Die Größer-Zeichen-Notation (>) und die Kleiner-Zeichen-Notation (<) können nicht miteinander oder mit Stern-Notation (*) oder Wildcard-Notation (?) kombiniert werden. Sie können nur zum Anzeigen einer Liste von Objekten verwendet werden (siehe ***Liste von Objekten anzeigen***).

options

Für eine ausführliche Beschreibung der *options*, siehe Abschnitt ***Optionen*** .

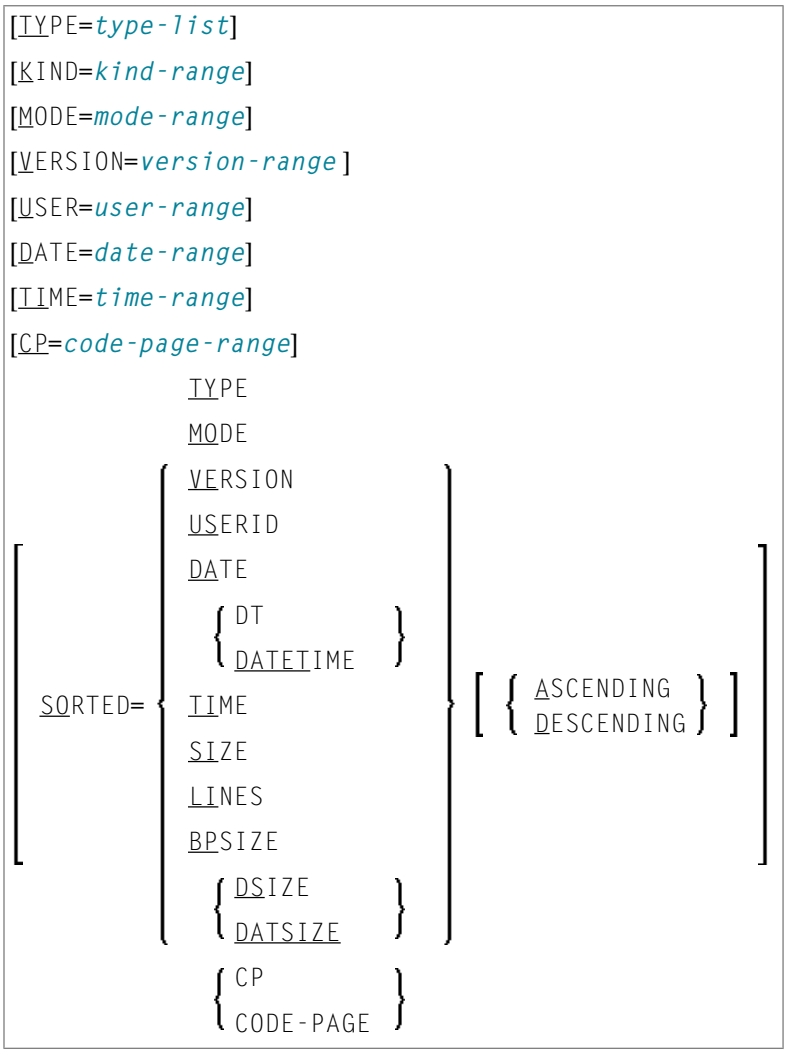
extended-type

Für *extended-type* können Sie einen der nachfolgend aufgeführten Objekttypen oder einen Stern (*) angegeben.



Für eine ausführliche Beschreibung siehe LIST EXTENDED.

range-clause



type-list	* (für alle Typen) oder eine Liste von 11 gültigen 1-Byte-Natural-Objekttyp-Zeichen (z.B. P for Program, M für Map).	
kind-range	*	Listet alle Objekte.
	S	Listet nur Objekte in Sourceform.
	C	Listet nur katalogisierte Objekte.
	S/C	Listet nur Objekte, die in Sourceform und als katalogisiertes Objekt vorhanden sind.
	S/	Listet nur Objekte auf, die nur als Sourceobjekte vorhanden sind.
	/C	Listet nur Objekte, die als katalogisiertes Objekt vorhanden sind.
	W	Listet nur in Sourceform und Objektform gespeicherte Objekte.
mode-range	*	Listet alle Objekte.
	S	Listet nur Objekte, die im Structured Mode geschrieben wurden.
	R	Listet nur Objekte, die im Report Mode geschrieben wurden.
version-range	Die Natural-Version der Natural-Objekte. Siehe auch Begriffsdefinition von Version im Glossary. Gültiges Versionsformat: *V.R.SM*, dabei ist *V* die einstellige Versionsnummer, *R* die einstellige Release-Nummer, *SM* die zweistellige System-Maintenance-Level-Nummer. Sie können einen Bereich von Versionsnummern angeben, siehe *range-notation*.	
user-range	Die User-ID des Benutzers, der ein Natural-Programmierobjekt gespeichert oder katalogisiert hat. Sie können eine einzelne User-ID oder einen Bereich von User-IDs angeben, siehe *range-notation*.	
date-range	Listet alle Objekte mit einem Speicher- oder Katalogisierungsdatum innerhalb des angegebenen Datumsbereichs. Sie können ein einzelnes Datum oder einen Datumsbereich angeben. Gültiges Datumsformat: *YYYY-MM-DD* (*YYYY*=Jahr, *MM*=Monat, *DD*=Tag) Gültige Datumsbereiche: ■ Vorangestellte Zeichen (Beispiel: 2002*) ■ Startwert (Beispiel: 2002-05>) ■ Endwert (Beispiel: 2003-02<) Zulässige spezielle Datumsangaben:	
	TODAY (+/-*nnnn*)	Alle Objekte mit aktuellem Tagesdatum.

		Im Anschluß an den Tag kann +*nnnn* oder -*nnnn* angegeben werden. Der Wert *nnnn* darf maximal 4 Zeichen lang sein. Das resultierende Datum wird berechnet aus dem aktuellen Tagesdatum plus oder minus *nnnn* Tage. Kann mit der Startwert-Option (>) oder der Endwert-Option (<) kombiniert werden, z.B. listet `TO-1>` alle Objekte, die in den letzen zwei Tage gespeichert oder katalogisiert wurden.
	`YESTERDAY`	Alle Objekte mit dem Datum des Tages vor dem aktuellen Tag.
	`MONTH`	Alle Objekte, deren Datum im Bereich des aktuellen Monats liegt.
	`YEAR`	Alle Objekte, deren Datum im Bereich des aktuellen Jahres liegt.
time-range	Listet alle Objekte mit einem Speicher- oder Katalogisierungszeitpunkt, der im angebenen Zeitbereich liegt. Geben Sie einen einzelnen Zeitwert oder einen Zeitbereich an. Gültiges Zeitformat: *HH:II:SS* (*HH* = Stunden, *II* = Minuten, *SS* = Sekunden) Gültige Zeitbereiche: ■ Vorangestellte Zeichen (Beispiel: 10:*) ■ Startwert (Beispiel: 10:30>) ■ Endwert (Beispiel: 11:20<)	
code-page-range	Sie können eine einzelne Codepage oder einen Bereich von Codepages angeben, siehe *range-notation*.	

range-notation

- Für eine Liste aller Objekte in der aktuellen Library geben Sie einen Stern (*) an.
- Für eine Liste eines bestimmten Bereichs von Objekten können Sie Stern-Notation (*) und Wildcard-Notation (?)verwenden:
  - Stern-Notation ist die Möglichkeit, einen Stern (*) anzugeben: der Stern steht für eine beliebige Zeichenkette beliebiger Länge.
  - Wildcard-Notation ist die Möglichkeit, ein Fragezeichen (?) anzugeben: das Fragezeichen steht für ein beliebiges Zeichen.
- Sie können eine oder mehrere Stern-Notationen (*) und Wildcard-Notationen (?) miteinander kombinieren.

- Für eine vollständige Liste von Objekten ab einen bestimmten Startwert bzw. bis zu einem bestimmten Endwert können Sie die Größer-Zeichen-Notation (>) bzw. die Kleiner-Zeichen-Notation (<) verwenden.
- Die Größer-Zeichen-Notation (>) und die Kleiner-Zeichen-Notation (<) können nicht miteinander oder mit Stern-Notation (*) oder Wildcard-Notation (?) kombiniert werden. Sie können nur zum Anzeigen einer Liste von Objekten verwendet werden (siehe *Liste von Objekten anzeigen*).

Inhalt des Arbeitsbereichs auflisten

LIST	Wenn Sie nur das `LIST`-Kommando (ohne Parameter) angeben, wird der Inhalt des Editor-Arbeitsbereichs aufgelistet.

Sourcecode eines einzelnen Objekts anzeigen

LIST *object-name* [*options*]	In beiden Fällen wird der Sourcecode des angegebenen Objekts gelistet. Wenn Sie einen einzelnen *object-name* mit dem LIST-Kommando eingeben, brauchen Sie keinen *object-type* anzugeben; es wird der Sourcecode des angegebenen Objekts gelistet. Wenn Sie einen *object-type* angeben, müssen Sie auch einen *object-name* angeben.
LIST *object-type object-name* [*options*]	

Sourcecode mehrerer Objekte nacheinander anzeigen

LIST SEQUENTIAL *object-name-range* [*options*]	In beiden Fällen müssen Sie Stern-Notation (*) und/oder Wildcard-Notation (?) für den *object-name-range* benutzen. Dann werden nacheinander die Sourcecodes aller Objekte angezeigt, die die angegebenen Auswahlkriterien erfüllen.
LIST SEQUENTIAL *object-type object-name-range* [*options*]	

Liste von Objekten anzeigen

LIST *object-name-range*	In beiden Fällen müssen Sie Stern-Notation (*) und/oder Wildcard-Notation (?) für den *object-name-range* benutzen. Sie erhalten eine Liste aller Objekte, die die angegebenen Auswahlkriterien erfüllen. In der Liste können Sie dann Objekte zur Anzeige auswählen, indem Sie sie mit dem Funktionscode `LI` markieren (siehe ***Eine Funktion auf einem Objekt ausführen***).
LIST *object-type object-name-range*	

Vorsortierte Liste von ausgewählten Objekten anzeigen

LIST *object-name-range*	In beiden Fällen müssen Sie Stern-Notation (*) und/oder Wildcard-Notation (?) für den *object-name-range* benutzen. Sie erhalten eine Liste aller Objekte, die die angegebenen Auswahlkriterien erfüllen. In der Liste können Sie dann Objekte zur Anzeige auswählen, indem Sie sie mit dem Funktionscode `LI` markieren (siehe ***Eine Funktion auf einem Objekt ausführen***). Mit der *range-clause* können Sie zusätzliche Auswahl- und Sortierkriterien angeben, siehe auch das **Beispiel** weiter unten.
LIST *object-name-range* *range-clause*	

Langnamen katalogisierter Subroutinen und Klassen anzeigen

LIST EXTENDED *object-name-range*	Listet die Langnamen der katalogisierten Subroutinen und Klassen. Bezüglich der Namensoptionen siehe auch *object-name-range* weiter oben.
LIST EXTENDED *extended-type object-name-range*	

NOC-Optionen katalogisierter Objekte anzeigen

LIST NOCOPT [*object-type*] *object-name-range*	Listet die katalogisierten Objekte, die mit dem Natural Optimizer Compiler (NOC) kompiliert wurden, und zeigt außerdem die zu Beginn beim CATALOG verwendeten NOC-Optionen an. Bezüglich der Namensoptionen siehe auch *object-name-range* weiter oben.

Compiler-Optionen katalogisierter Objekte anzeigen

LIST OPTIONS [*object-type*] *object-name-range*	Listet die katalogisierten Objekte auf und zeigt die beim CATALOG verwendeten Compiler-Optionen. Bezüglich der Namensoptionen siehe auch *object-name-range* weiter oben. Standardmäßig werden die endgültig verwendeten Compiler-Optionen angezeigt, das heißt, die Optionseinstellungen, die am Ende des CATALOG aktiv waren. Bei Objekten, die mit Natural Version 4.2.5 oder höher katalogisiert wurden, können auch die zu Beginn gesetzten Compiler-Optionen (das heißt, die Optionseinstellungen, die am Anfang des CATALOG aktiv waren) oder die geänderten Compiler-Optionen (das heißt, die Optionseinstellungen, die innerhalb des Sourcecode geändert wurden) angezeigt werden. Bezüglich der Bereichsfelder auf der Online-Maske siehe den enstprechenden Hilfe-Bildschirm.

Directory-Informationen anzeigen

LIST DIRECTORY	Zeigt die Directory-Informationen zu dem gerade im Editor-Arbeitsbereich befindlichen Objekt an: ■ **Sourcecode:** „Saved-on"-Datum und -Uhrzeit, Library-Name, User-ID, Programmiermodus (Reporting oder Structured Mode), TP-System, Terminal-ID, Betriebssystem, Transaktion, Natural-Version, Codepage-Informationen (falls verfügbar), Sourceprogrammgröße ■ **Objektcode:** „Cataloged-on"-Datum und -Uhrzeit, Library-Name, User-ID, Programmiermodus (Reporting oder Structured Mode), TP-System, Terminal-I/O-Transaktion, Natural-Version, Codepage-Informationen (falls verfügbar), Betriebssystem/Version, verwendete GDA, Größe der Global Data Area, Größe in DATSIZE, Größe im Buffer Pool, Größe des OPT-Code (Größe des vom Natural Optimizer Compiler erzeugten Maschinencodes),

	Anfangs-OPT-String (OPT-Profileparameterwert, der zur STOW-Zeit wirksam war), Compiler-Optionen
LIST DIRECTORY *object-name*	Dieses Kommando zeigt die Directory-Informationen zu dem betreffenden Objekt an (wie bei LIST DIRECTORY beschrieben). Wenn Sie Stern-Notation (*) oder Wildcard-Notation (?) für *object-name-range* verwenden, werden nacheinander die Directory-Informationen der entsprechenden Objekte angezeigt.
LIST DIRECTORY *object-name-range*	
LIST *object-name* **WITH DIRECTORY**	Dieses Kommando zeigt zunächst die Directory-Informationen zu dem betreffenden Objekt an (wie bei LIST DIRECTORY beschrieben) und listet anschließend den Sourcecode des Objekts.

DDMs (Views) anzeigen

LIST DDM	Dieses Kommando zeigt eines Liste aller DDMs an.
LIST DDM *ddm-name*	Wenn Sie einen einzelnen DDM-Namen angeben, wird das angegebene DDM angezeigt. Für *ddm-name* können Sie einen einzelnen DDM-Namen (bis zu 32 Zeichen lang) oder wie für *object-name-range* einen Bereich angeben, um eine Liste eines bestimmten Bereichs von DDMs anzuzeigen.

Anmerkung: Statt des Schlüsselworts DDM können Sie auch das Schlüsselwort VIEW (oder kurz V) verwenden.

Optionen

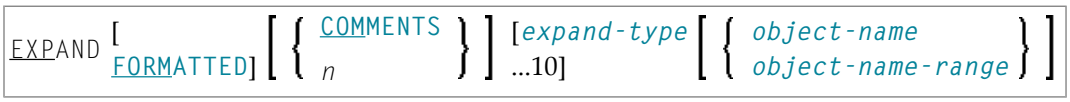
Statt *options* können Sie eine der nachfolgend aufgeführten Optionen angeben.

```
{ [[WITH] DIRECTORY] [NUMBERS OFF] [expand-option]
  formatted-option
  CONVERTED                                        }
```

DIRECTORY	Dieses Kommando zeigt zunächst die Directory-Informationen zu dem betreffenden Objekt an (wie bei ***Directory Informationen anzeigen*** beschrieben) und listet anschließend den Sourcecode des Objekts.
NUMBERS OFF	Standardmäßig wird der Sourcecode eines Objekts mit Zeilennummern gelistet. Wünschen Sie eine Anzeige ohne Zeilennummern, verwenden Sie die Option NUMBERS OFF (Vgl. *Unterkommandos für eine angezeigte Source*).

CONVERTED	Standardmäßig wird der Sourcecode in der zur Systemdatei gespeicherten Codepage angezeigt. Sie können die Option CONVERTED verwenden, um den Sourcecode in der Default-Codepage (siehe Systemvariable *CODEPAGE).

expand-option



EXPAND *object-name* **EXPAND** *object-name-range*	Die EXPAND-Option erlaubt es Ihnen, die Sourcen von Objekten, die in der gelisteten Source referenziert werden (Copycodes, Data Areas, Maps, Helproutinen, externe Subroutinen, Subprogramme, mit einem FETCH-Statement aufgerufene Programme, Fehlermeldungen), *innerhalb* der gelisteten Source anzuzeigen. Dies kann besonders im Batch-Betrieb sehr hilfreich sein. Wenn Sie beispielsweise ein Source-Programm listen, das ein INCLUDE-Statement enthält, können Sie den Sourcecode des betreffenden Copycodes innerhalb des gelisteten Source-Programms unmittelbar nach dem INCLUDE-Statement anzeigen. In den folgenden Erklärungen werden Objekte, die innerhalb einer Source gelistet werden, als „Expand-Objekte“ bezeichnet. **Unterkommandos zur EXPAND-Option** Innerhalb eines gelisteten Expand-Objekts sind nur die folgenden Unterkommandos verfügbar (die oben beschrieben sind): PRINT + - - . Siehe ***Beispiele zur Benutzung der Objekt-Liste***.
EXPAND FORMATTED	Die Option EXPAND FORMATTED ist nur für innerhalb einer Source gelistete Data Areas (mit STOW gespeichert, d.h. bei denen der Zeitstempel des Sourceobjekts und des katalogisierten Objekts identisch ist) und Maps relevant. Für Data Areas gilt: ■ Wenn Sie FORMATTED nicht angeben, ähnelt die Anzeige der Data Area der im Data-Area-Editor. ■ Wenn Sie FORMATTED angeben, ähnelt die Anzeige der Data Area einem DEFINE DATA-Statement. Dies gilt nur bei mit einem STOW-Kommando gespeicherte Data Areas (d.h. bei denen der Zeitstempel des Sourceobjekts und des katalogisierten Objekts identisch ist), siehe auch Unterkommando FORMAT. Für Maps gilt: ■ Wenn Sie FORMATTED nicht angeben, wird die *Source* der Map gelistet.

<table>
<tr><td></td><td>■ Wenn Sie FORMATTED angeben, wird das Layout der Map gelistet (d.h. die Map, wie Sie ein Benutzer zur Laufzeit sieht).</td></tr>
<tr><td>EXPAND COMMENTS

EXPAND n</td><td>Wenn Sie die Option EXPAND COMMENTS verwenden, werden nur die Kommentarzeilen am Anfang des Expand-Objekts gelistet; d.h. das Expand-Objekt wird bis zu (aber nicht einschließlich) der ersten Sourcecode-Zeile, die keine Kommentarzeile ist, gelistet.

Wenn Sie die Option EXPAND n verwenden, werden nur die ersten n Zeilen des Expand-Objekts gelistet.

Wenn Sie keine dieser beiden Optionen verwenden, wird das Expand-Objekt vollständig gelistet.</td></tr>
<tr><td>expand-type</td><td>Als expand-type geben Sie den Objekttyp des Expand-Objekts an. Sie können die folgenden expand-types angeben:
<table>
<tr><td>P</td><td>Programme</td><td rowspan="12">Falls Sie mehr als einen expand-type angeben möchten, geben Sie sie in beliebiger Reihenfolge und ohne Leerzeichen an; um Maps, Copycodes und Subroutinen innerhalb einer gelisteten Source zu listen, geben Sie beispielsweise MCS als expand-type an.</td></tr>
<tr><td>N</td><td>Subprogramme</td></tr>
<tr><td>S</td><td>Externen Subroutinen</td></tr>
<tr><td>H</td><td>Helproutinen</td></tr>
<tr><td>G</td><td>Global Data Areas</td></tr>
<tr><td>L</td><td>Local Data Areas</td></tr>
<tr><td>A</td><td>Parameter Data Areas</td></tr>
<tr><td>M</td><td>Maps</td></tr>
<tr><td>C</td><td>Copycodes</td></tr>
<tr><td>E</td><td>Fehlermeldungen</td></tr>
<tr><td>4</td><td>Klassen</td></tr>
<tr><td>*</td><td>Alle Objekttypen</td></tr>
</table></td></tr>
<tr><td>object-name

object-name-range</td><td>Als object-name oder object-name-range geben Sie den Namen bzw. die Namen der Expand-Objekte an, die innerhalb der gelisteten Source gelistet werden soll(en).

Für den object-name bzw. object-name-range eines Expand-Objekts stehen die gleichen Optionen zur Verfügung wie für den object-name bzw. object-name-range des gelisteten Hauptobjekts in der primären LIST-Kommandosyntax. Ausnahmen: die Größer-Zeichen-Notation (>) und die Kleiner-Zeichen-Notation.</td></tr>
</table>

formatted-option

FORMATTED-Option

Die FORMATTED-Option gilt für Data Areas (mit STOW gespeichert, d.h. bei denen der Zeitstempel des Sourceobjekts und des katalogisierten Objekts identisch ist) und Maps:

FORMATTED	**Mit STOW-Kommando gespeicherte Data Area:** Wenn Sie diese Option für eine Data Area angeben, wird die Data Area formatiert angezeigt, d.h. die Anzeige ähnelt einem DEFINE DATA-Statement (Vgl. Unterkommando FORMAT). Standardmäßig ist die Anzeige von Data Areas unformatiert, d.h. die Anzeige ähnelt der im Data Area Editor. Dies gilt nur für mit STOW-Kommando gespeicherte Data Area (das heißt, der Zeitstempel des Source-Objekts und des katalogisierten Objekts sind identisch). Standardmäßig ist die Anzeige von Data Areas unformatiert, das heißt, die Anzeige ist ähnlich wie im Dateneditor. Sie können die Standardeinstellung im List-Profil ändern, siehe ***Individuelles List-Profil erstellen*** und siehe auch Unterkommando FORMAT **Map:** Wenn Sie die FORMATTED-Option für eine Map angeben, wird das *Layout* der Map angezeigt, d.h. die Map, wie Sie ein Benutzer zur Laufzeit sieht.

Weitere FORMATTED-Optionen für Maps

Wenn Sie Maps mit dem LIST-Kommando anzeigen, können Sie zusätzlich zum Schlüsselwort FORMATTED weitere Optionen angeben:

['*c*'] ['*c*']	**Füllzeichen benutzen:** Sie können Füllzeichen *c* für Eingabefelder (AD=A und AD=M) und Ausgabefelder (AD=O) angeben, um diese Felder sichtbar zu machen. Sie können ein beliebiges Zeichen als Füllzeichen angeben.

	Das folgende Beispiel bewirkt, dass alle Eingabefelder mit einem Unterstrich (_) und alle Ausgabefelder mit einer Raute (#) gefüllt angezeigt werden. `LIST MAP map-name FORMATTED '_' '#'`
SETTINGS	**Map-Einstellungen:** Diese Option bewirkt, dass die **Map Settings** der Map angezeigt werden. `LIST MAP map-name FORMATTED SETTINGS`
FIELDS	**Feldübersicht:** Diese Option bewirkt, dass das **Field Summary**, d.h. die Liste der Map-Felder, angezeigt wird. `LIST MAP map-name FORMATTED FIELDS`
EXTFIELDS	**Erweiterte Feldeditier-Informationen:** Diese Option bewirkt, dass die **Extended Field Editing**-Informationen für alle Map-Felder angezeigt werden. `LIST MAP map-nameFORMATTED EXTFIELDS`

Anzeige der Verarbeitungsregeln für eine Map

Die folgenden Optionen bewirken, dass die von der Map verwendeten **Processing Rules** angezeigt werden. Diese Verarbeitungsregeln werden in der Reihenfolge der Felder, denen sie zugewiesen sind, angezeigt und je Feld in der Reihenfolge des Rangs (Rank).

RULES	**Alle Verarbeitungsregeln anzeigen:** `LIST MAP map-name FORMATTED RULES` Zeigt *alle* Verarbeitungsregeln für die angegebene Map an.
INLINERULES	**Nur die Inline-Verarbeitungsregeln anzeigen:** `LIST MAP map-name FORMATTED INLINERULES` Zeigt nur die Inline-Verarbeitungsregeln für die angegebene Map an.

FREERULES	**Nur die freien Verarbeitungsregeln anzeigen:** `LIST MAP map-name FORMATTED FREERULES` Zeigt nur die freien Verarbeitungsregeln für die angegebene Map an.
AUTORULES	**Nur die automatischen Verarbeitungsregeln anzeigen:** `LIST MAP map-name FORMATTED AUTORULES` Zeigt nur die automatischen Verarbeitungsregeln für die angegebene Map an.

Siehe auch Unterkommandos `LAYOUT` und `FORMAT` im Abschnitt ***Source-Liste***.

Objekt-Auswahlliste

Wenn Sie Stern- oder Wildcard-Notation für den *object-name* verwenden, erhalten Sie eine Liste aller Objekte, die Ihr Auswahlkriterium erfüllen. Auf der Liste können Sie dann die Objekte, die Sie gelistet, gedruckt usw. haben möchten, mit einem Funktionscode markieren oder Sie können ein Natural-Systemkommando oder ein `LIST`-Unterkommando in der Kommandozeile absetzen.

Dieser Abschnitt beschreibt die Funktionen, Unterkommandos und Funktionscodes, die in der angezeigten Liste der Objekte verfügbar sind, z.B. nachdem Sie ein `LIST *`-Kommando abgesetzt haben.

- Erklärung der Spaltenüberschriften
- In der Objekt-Auswahlliste blättern
- Neue Kriterien für die Auswahliste angeben
- Informationen in der Auswahlliste
- Intensivierte dargestellte Informationen in der Auswahlliste
- Unterkommandos für die Auswahlliste
- Eine Funktion auf einem Objekt ausführen
- Inhalt der Auswahlliste sortieren
- Beispiele zur Benutzung der Objekt-Liste

Erklärung der Spaltenüberschriften

Die Liste der Objekte enthält folgende Spalten:

Spalte	Erklärung
`Cmd`	In diese Spalte können Sie einen Code eingeben, um eine Funktion auf einem Objekt auszuführen. Siehe *Eine Funktion auf einem Objekt ausführen*.
`Name`	Name des Objekts.
`Type`	Objekttyp.
`S/C`	Zeigt an, ob das Objekt als Source (S) und/oder als katalogisiertes Objekt (C) vorhanden ist.
`SM`	Der Natural-Programmiermodus, der beim Erstellen des Objekts verwendet wurde. S = Structured Mode R = Reporting Mode
`Version`	Natural-Produktversion, die zum Erstellen bzw. Katalogisieren des Objekts verwendet wurde.
`User ID`	User-ID des Benutzers, der das Objekt erstellt bzw. katalogisiert hat.
`Date`, `Time`	Zeitpunkt (Datum, Uhrzeit), zu dem das Objekt erstellt bzw. katalogisiert wurde.

In der Objekt-Auswahlliste blättern

Wenn Ihnen eine Auswahlliste von Objekten angezeigt wird, können Sie darin wie folgt blättern:

- Um in der Liste eine Seite vor bzw. zurückzublättern, drücken Sie PF8 bzw. PF7.
- Um an den Anfang bzw. ans Ende der Liste zu blättern, drücken Sie PF6 bzw. PF9.

Neue Kriterien für die Auswahliste angeben

Wenn Ihnen eine Auswahlliste von Objekten angezeigt wird, befinden sich unmittelbar unterhalb der Spaltenüberschriften Felder, in denen die Auswahlkriterien für die derzeitige Liste angezeigt werden. Sie können die Auswahlkriterien ändern, indem Sie die Werte dieser Felder überschreiben. Wenn Sie ein Fragezeichen (?) in eins dieser Felder eingeben, erhalten Sie Informationen über die möglichen Werte für dieses Feld.

Informationen in der Auswahlliste

Falls von einem Objekt sowohl eine Source als auch ein Objektmodul existiert (wie in der Spalte **S/C** angezeigt), beziehen sich die angezeigten Informationen auf die Source, nicht das Objektmodul.

Anmerkung: Wenn die SORT-Funktion aktiv ist, können die Source und das Objektmodul getrennt angezeigt werden, d.h. wenn die Liste nach Objektdatum sortiert wird und die Source und das Objektmodul unterschiedliche Datumswerte aufweisen.

Um weitere Informationen zu Source- und katalogisierten Objekten anzuzeigen

- Drücken Sie PF11, um nach rechts zu blättern.

  Oder:

  Drücken Sie PF10, um nach links zu blättern.

Anmerkung: Standardmäßig wird die Anzahl der Source-Zeilen bei Sourceobjekten aus Durchsatzgründen nicht berechnet. Wenn Sie wünschen, dass die die Anzahl der Source-Zeilen bei Sourceobjekten angezeigt wird, können Sie entweder das Unterkommando `COUNTSOURCE ON` eingeben oder im List-Profil (siehe *Individuelles List-Profil erstellen*) den Parameter `COUNT-SOURCE-LINES` auf `Y` setzen.

Intensivierte dargestellte Informationen in der Auswahlliste

Ist eine Informationseinheit in der Liste intensiviert dargestellt, bedeutet dies, dass dort eine Abweichung zwischen der Source eines Objekts und dem entsprechenden Objektmodul besteht. Um Näheres über die Abweichung zu erfahren, können Sie das Objekt mit Funktionscode `LD` markieren, um sich die betreffende Directory-Information anzeigen zu lassen. Zur Beseitigung der Abweichung genügt es in der Regel, das Objekt neu mit `STOW` zu speichern (Funktionscode `ST`; siehe unten).

Unterkommandos für die Auswahlliste

In einer Objekt-Auswahlliste können Sie ein Natural-Systemkommando oder ein `LIST`-Unterkommando in der Kommandozeile eingeben. Gültige Unterkommandos sind:

Code	Funktion	
`CODE-PAGE` oder `CP`	`ON`	Codepage-Informationen werden zu jedem Objekt angezeigt. Dies ist der Standardwert.
	`OFF`	Codepage-Informationen werden nicht angezeigt.
`SC`	Nur die Objekte werden aufgelistet, die einen Scan-Wert enthalten (kann nur bei langen Listen angewendet werden).	
`SC OFF`	Scan-Modus ausschalten.	

Code	Funktion	
SHORT	Liste der Objekte in Kurzform anzeigen, d.h. es werden nur die Objektnamen angezeigt (kann nur bei SC OFF verwendet werden).	
LONG	Liste der Objekte in Langform (mit allen vorhandenen Feldern) anzeigen.	
PRINT	Liste der Objekte drucken.	
EXTENDED	Langnamen von Subroutinen/Klassen anzeigen; entspricht der Funktion LIST EXTENDED *.	
ALL *fx*	Den Funktionscode *fx* für alle angezeigten Objekte eingeben. *fx* = gültiger Funktionscode für ein gelistetes Objekt.	
SORT	Sort-Fenster anzeigen (siehe ***Liste der Objekte sortieren***).	
COUNTSOURCE	ON	Anzahl der Source-Zeilen für Sourceobjekte wird angezeigt.
	OFF	Anzahl der Source-Zeilen für Sourceobjekte wird nicht angezeigt.
MARK-LONG-LINES	ON	Lange Zeilen in der Auflistung eines Sourceobjekts werden in den ersten zwei Stellen mit einem L markiert. Der Standardwert kann im List-Profil angegeben werden; siehe ***Individuelles List-Profil erstellen***.
	OFF	Lange Zeilen in der Auflistung eines Sourceobjekts werden nicht markiert.
DEFINE-DATA	ON	Die aufgelistete Data-Area-Source wird standardmäßig im DEFINE DATA-Format angezeigt (wie bei LIST *dataarea* FORMATTED). Der Standardwert kann im List-Profil angegeben werden; siehe ***Individuelles List-Profil erstellen***.
	OFF	Die Anzeige einer Data Area Source erfolgt unformatiert.
LISTPROFILE	Aktuelle Werte der Parameter des List-Profils anzeigen (siehe ***Individuelles List-Profil erstellen***).	
NOCOPT	Listet die katalogisierten Objekte, die mit dem Natural Optimizer Compiler (NOC) kompiliert wurden, und zeigt außerdem die zu Beginn beim CATALOG verwendeten NOC-Optionen an; entspricht LIST NOCOPT *, siehe ***NOC-Optionen katalogisierter Objekte anzeigen***.	
OPTIONS	Listet die katalogisierten Objekte auf und zeigt die beim CATALOG zu Beginn verwendeten Compiler-Optionen; entspricht LIST OPTIONS *, siehe ***Compiler-Optionen katalogisierter Objekte anzeigen***.	
REUSE	ON	Wiederverwendungsmodus einschalten. Die zuletzt angezeigte Liste wird erneut verwendet, nachdem in der Spalte **Cmd** Kommdos ausgeführt wurden. Ausgenommen sind folgende Kommandos:

Code	Funktion	
		E ED (Edit) CA (Catalog) UC (Uncat) S ST (Stow) D DE (Delete) RE (Rename)
	OFF	Wiederverwendungsmodus ausschalten. Nach der Ausführung von Kommandos, die in der Spalte **Cmd** eingegeben wurden, wird die Liste neu erstellt.
REFRESH	Die gerade angezeigte Liste aktualisieren. Dieses Unterkommando können Sie insbesondere bei eingeschaltetem Wiederverwendungmodus benutzen.	
+	Eine Seite vor blättern.	
-	Eine Seite zurück blättern.	
++	Ans Ende der Objektliste blättern.	
--	An den Anfang der Objektliste blättern.	
?	Hilfe zur Kommandozeile aufrufen.	

Eine Funktion auf einem Objekt ausführen

Um ein Objekt von der Auswahlliste für eine bestimmte Funktion auszuwählen, markieren Sie es einfach in der linken Spalte (unter der Überschrift **Cmd**) mit dem entsprehenden Funktionscode.

Sie können mehrere Objekte in der Auswahlliste mit unterschiedlichen Funktionscodes markieren. Die Funktionen werden dann nacheinander ausgeführt.

Die folgenden Funktionscodes stehen Ihnen zur Verfügung (mögliche Abkürzungen sind unterstrichen).

Funktionscode	Funktion
?	Ein Fenster zeigt Ihnen, welche Funktionen für das markierte Objekt zur Verfügung stehen. Es werden jeweils nur die Funktionen aufgeführt, die für das betreffende Objekt auch tatsächlich möglich sind (zum Beispiel wird Ihnen für eine Subroutine nicht die Funktion RU(N) angeboten und für ein Objekt, das nur in Sourceform existiert, nicht die Funktion EX(ECUTE)). In dem Fenster können Sie für das betreffende Objekt die gewünschte Funktion auswählen.
CA	Kompilieren und speichern in Objektform (entspricht dem Systemkommando CATALOG).

Funktionscode	Funktion
DE	Löschen eines Objekts (entspricht dem Systemkommando DELETE).
DL	Herunterladen (Download) des Objekts vom Großrechner auf einen Personal Computer (nur möglich, wenn Natural Connection installiert ist).
ED	Editieren des Sourcecodes (entspricht dem Systemkommando EDIT).
EX	Ausführen des Objektmoduls (entspricht dem Systemkommando EXECUTE).
LC	Anzeigen des in die Default-Codepage *CODEPAGE umgesetzten Sourcecodes des Objekts (entspricht dem Systemkommando LIST *object-name* CONVERTED).
LD	Anzeigen der Directory-Information zum Objekt (entspricht dem Systemkommando LIST DIRECTORY *object-name*).
LE	Anzeigen des Sourcecodes des Objekts in expandierter Form (entspricht LIST *object-name* EXPAND *).
LF	Formatierte Anzeige von Data Area oder Map (entspricht dem Systemkommando LIST *object-name* FORMATTED).
LI	Anzeigen (List) des Sourcecodes.
LN	Anzeigen des Langnamens der Subroutine oder der Klasse (nur möglich, wenn ein katalogisiertes Objekt vorhanden ist) oder der Resource.
NO	Zeigt die während des Katalogisierens mit CATALOG verwendeten Natural Optimizer (NOC) Optionen (funktioniert nur, wenn ein katalogisiertes Objekt existiert).
OP	Zeigt die zu Beginn, die am Ende verwendeten und die geänderten Compiler-Optionen, die beim Katalogisieren mit CATALOG verwendet wurden (funktioniert nur, wenn ein katalogisiertes Objekt existiert). Die zu Beginn verwendeten und die geänderten Compiler-Optionen können nur bei Objekten angezeigt, werden, die mit Natural Version 4.2.5 oder höher katalogisiert wurden.
PR	Drucken (Print) des Sourcecodes.
RE	Umbenennen des Objekts (entspricht dem Systemkommando RENAME).
RU	Kompilieren und Ausführen der Source (entspricht dem Systemkommando RUN).
ST	Speichern in Source- und Objektform (entspricht dem Systemkommando STOW).
UC	Löschen des Objektmoduls (entspricht dem Systemkommando UNCATALOG).
.	Ende (Verlassen des Auswahllistenfensters).

Inhalt der Auswahlliste sortieren

Das LIST-Kommando bietet Ihnen die Möglichkeit, die Liste der angezeigten Objekte nach mehreren Sortierkriterien zu sortieren.

Anmerkung: Damit Sie diese Funktion benutzen können, muss die Größe des vom Sort-Programm benutzten Work Buffers mit dem Schlüsselwortparameter WRKSIZE des Natural-Profilparameters SORT auf einen ausreichenden Wert gesetzt werden. Die maximale Größe der Liste, die sortiert werden kann, wird durch die Größe des Work Buffers begrenzt.

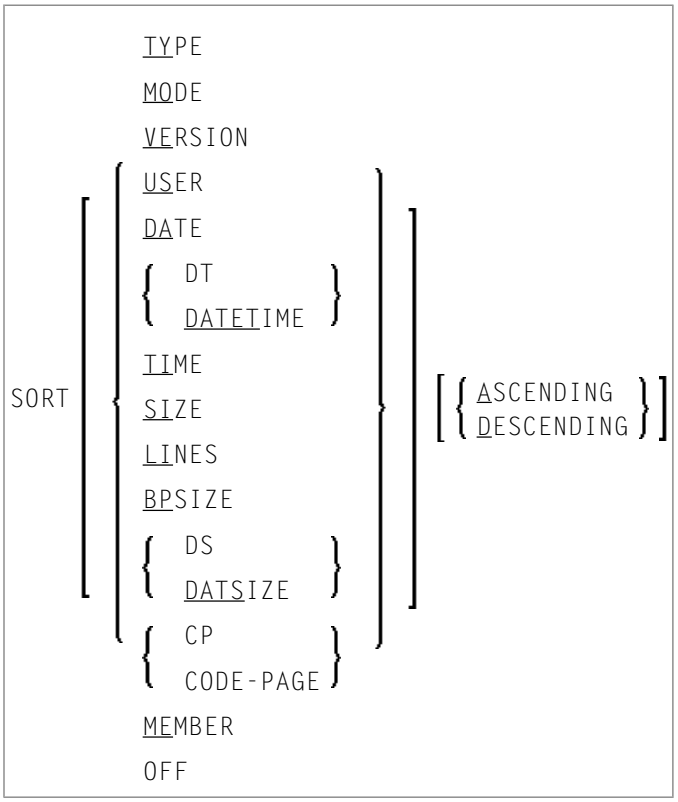
Um die Sort-Funktion aufzurufen

- Drücken Sie PF4.

  Oder:

  Setzen Sie in der Liste der Objekte ein SORT-Unterkommando ab.

Syntax des Unterkommandos SORT



Wenn Sie PF4 drücken, können Sie in einem Fenster angeben, ob Sie die Liste sotieren wollen (Y/N), ein Sort-Feld markieren und in auf- oder absteigender Reihenfolge sortieren wollen.

Sort-Feld	Schlüsselwort in Sort-Syntax
Natural object type	TYPE
Programming mode (Reporting oder Structured mode)	MODE
Version	VERSION
User ID	USER
Date	DATE
Date and time	DATETIME
Time	TIME

Sort-Feld	Schlüsselwort in Sort-Syntax
Source size	SIZE
Number of source lines	LINES
Buffer pool size	BPSIZE
DATSIZE (Größe des lokalen Daten-Buffers)	DS/DATSIZE
Code page	CP/CODE-PAGE
Member names of subroutines or classes (nur in der erweiterten Auswahlliste verfügbar)	MEMBER

Um die Sort-Funktion auszuschalten

- Geben Sie das Unterkommando SORT OFF ein.

  Oder:

  Deaktivieren Sie die Sort-Funktion in dem mit PF4 aufgerufenen Fenster.

Die sortierte Liste wird in einem Natural-Textobjekt in der Library WORKPLAN erstellt. Der Name des Objekts wird durch das LIST-Kommando erzeugt. Wenn Sie das List-Profil aufrufen (siehe *Individuelles List-Profil erstellen*), können Sie den Namen für das Textobjekt sowie die Library angeben.

Beispiele zur Benutzung der Objekt-Liste

LIST *	Listet alle Objekte jeglichen Typs in der aktuellen Library.
LIST S *	Listet alle Subroutinen in der aktuellen Library.
LIST SYS*	Listet alle Objekte (egal welchen Typs), deren Namen mit SYS anfangen.
LIST M SYS*	Listet alle Maps, deren Namen mit SYS anfangen.
LIST C *CODE	Listet alle Copycodes, deren Namen mit CODE aufhören.
LIST NAT*AL	Listet alle Objekte, deren Namen mit NAT anfangen und mit AL aufhören, ganz gleich wieviele andere Zeichen zwischen NAT und AL stehen (dies würde die Namen NATURAL und NATIONAL sowie NATAL einschließen).
LIST DOO?	Listet alle Objekte mit 4-stelligen Namen, die mit DOO anfangen (dies würde die Namen DOOR und DOOM einschließen, aber nicht DOO oder DOODLE).
LIST M NAT?AL	Listet alle Maps, deren Namen mit NAT anfangen und mit AL aufhören, und zwar mit genau einem Zeichen zwischen NAT und AL (dies würde die Namen NAT1AL und NAT2AL einschließen, aber nicht NATAL oder NATIONAL).
LIST M *1*	Listet alle Maps, deren Namen eine 1 enthalten.
LIST M F>	Listet alle Maps, und zwar ab der ersten, deren Name mit F anfängt.
LIST M MA<	Listet alle Maps, und zwar ab der ersten und bis zu der namens MA (falls vorhanden).

`LIST N?T*AL`	Listet Objekte wie zum Beispiel `NATAL`, `NATURAL`, `NAT`*`vr`*`AL` (wobei *`vr`* für die Versions- und Release-Nummer steht).
`LIST E* TYPE=PM KIND=S DATE=YEAR SORTED=DATE ASCENDING`	Erstellt eine Liste aller Sourceobjekte vom Programmen und Maps, deren Namen mit `E` anfangen und die im aktuellen Jahr gespeichert wurden. Die Liste ist in aufsteigender Reihenfolge nach dem Objektdatum sortiert.

Source-Liste

Unterkommandos für eine angezeigte Source

Wenn Ihnen der Sourcecode eines Objekts angezeigt wird, können Sie in der Kommandozeile eines der folgenden Kommandos eingeben.

Unterkommando	Funktion
`+`	Blättert ein Seite vor.
`-`	Blättert eine Seite zurück.
`++`	Blättert ans Ende der Source.
`BOTTOM`	
`--`	Blättert an den Anfang der Source.
`TOP`	
`+`*`n`*	Blättert *`n`* Zeilen vor.
`-`*`n`*	Blättert *`n`* Zeilen zurück.
`nnnn`	Blättert zu Zeile Nummer *`nnnn`*.
`CONVERTED`	Siehe `CONVERTED` bei *Options*.
`DBFNR ON`	Zeigt die `DBID` und `FNR` der Source-Library in der Kopfzeile des Sourcecodes.
`DBFNR OFF`	Zeigt die Kopfzeile des Sourcecodes ohne die `DBID` und `FNR` der Source-Library. Dies ist der Standardwert.
`EXPAND`	Siehe *`expand-option`*.
`FIELDS`	Gilt nur für Maps: Zeigt das **Field Summary**, d.h. die Liste der Felder in der Maske.
`FIND`	Zeigt nur die Sourcecode-Zeilen, in denen der angegebene Wert (*`value`*) vorkommt.
`FIND` *`value`*	

Unterkommando	Funktion
FIND ABSOLUTE *value*	Wenn Sie nur das Kommando FIND eingeben, erscheint ein Fenster, in dem Sie den gewünschten Wert eingeben sowie bestimmen können, ob die Suche absolut sein soll oder nicht. Wenn Sie FIND ohne ABSOLUTE eingeben, ist die Suche absolut, d.h. der Wert wird nur gefunden, wenn er als isoliertes Wort vorkommt. Dies ist der Standardwert. Wenn Sie FIND mit ABSOLUTE eingeben, ist die Suche absolut, d.h. der Wert wird auch gefunden, wenn er als Teil einer längeren Zeichenkette vorkommt.
FORMAT	Gilt nur für Data Areas und Maps: zeigt die ***formatierte*** Data Area bzw. Map, sowie zur Map gehörige Informationen.
LAYOUT	Gilt nur für Maps. Zeigt das Layout der Map, d.h. die Map, wie Sie ein Benutzer zur Laufzeit sieht.
NUMBERS ON	Zeigt die Source mit Sourcecode-Zeilennummern. Dies ist der Standardwert.
NUMBERS OFF	Zeigt die Source ohne Sourcecode-Zeilennummern.
PRINT	Druckt die Source.
REF REF *value* REF ABSOLUTE *value*	Zeigt die Zeilennummer der Sourcecode-Zeilen, in denen der angegebene Wert (*value*) vorkommt. Wenn Sie nur das Kommando REF eingeben, erscheint ein Fenster, indem Sie den gewünschten Wert eingeben sowie bestimmen können, ob die Suche absolut sein soll oder nicht. Wenn Sie REF ohne ABSOLUTE eingeben, dann wird der Wert wird nur gefunden, wenn er als isoliertes Wort vorkommt. Dies ist der Standardwert. Wenn Sie REF mit ABSOLUTE eingeben, ist die Suche absolut, d.h. der Wert wird auch gefunden, wenn er als Teil einer längeren Zeichenkette vorkommt.
RULES	Gilt nur für Maps. Zeigt die von der Map verwendeten Verarbeitungsregeln (Processing Rules). Die Rules werden in der Reihenfolge der Felder, denen sie zugewiesen sind, gezeigt und je Feld in der Reihenfolge des Rangs (Rank).
SCAN SCAN *value* SCAN ABSOLUTE *value*	Zeigt alle Zeilen intensiviert an, in denen der angegebene Wert (*value*) vorkommt. Die erste Zeile, in der der Wert vorkommt, wird an den Anfang der Seite geblättert. Wenn Sie nur das Kommando SCAN eingeben, erscheint ein Fenster, in dem Sie den gewünschten Wert eingeben sowie bestimmen können, ob die Suche absolut sein soll oder nicht. Wenn Sie SCAN ohne ABSOLUTE eingeben, dann wird der Wert nur gefunden, wenn er als isoliertes Wort vorkommt. Dies ist der Standardwert.

Unterkommando	Funktion
	Wenn Sie SCAN mit ABSOLUTE eingeben, ist die Suche absolut, d.h. der Wert wird auch gefunden, wenn er als Teil einer längeren Zeichenkette vorkommt.
SCAN= or SC=	Führt das letzte SCAN-Kommando erneut aus. Alternativ können Sie PF5 drücken.
SETTINGS	Gilt nur für Maps. Zeigt die Map Settings der Map.
ZOOM [*expand-type*...10] *object-name*	Die Angabe eines einzelnen Namens (*object-name*) mit dem ZOOM-Kommando hat die gleiche Auswirkung wie das Markieren des Namens mit dem Cursor in der gelisteten Source (siehe ***Cursor-sensitive Objektauswahl***). Das ausgewählte Objekt wird in einem Fenster angezeigt. Wenn Sie Stern-Notation (*) bzw. Wildcard-Notation (?) im *object-name* verwenden, werden alle ausgewählten Objekte in dem Fenster gezeigt, und zwar in der Reihenfolge, in der sie in der gelisteten Source referenziert werden. Die Angabe eines *expand-type* entspricht der *expand-option*. Für ein Objekt, das in einem über ZOOM aufgerufenen Fenster gezeigt wird, stehen die gleichen Unterkommandos zur Verfügung wie für eine normal gelistete Source (außer PRINT, EXPAND und ZOOM). Wenn aufgrund von Stern- oder Wildcard-Notation mehrere Objekte angezeigt werden, können Sie darüber hinaus mit den Kommandos NEXT und PREV (bzw. PF4 und PF5) von einem Objekt im Fenster zum nächsten bzw. zum vorigen blättern.
ZOOM [*expand-type*...10] *object-name-range*	
.	Beenden.

Anmerkung: Standardmäßig werden die DBID und FNR der Source-Library nicht in der Kopfzeile der aufgelisteten Source angezeigt. Wenn Sie wollen, dass die DBID und FNR der Source-Library angezeigt werden, können Sie entweder das Unterkommando DBFNR ON eingeben oder im List-Profil den Parameter SOURCE-LIST-WITH-DBID-FNR auf Y setzen (siehe ***Individuelles List-Profil erstellen***).

Unterkommando FORMAT

Dieses Unterkommando gilt nur für Data Areas (die mit STOW gespeichert wurden und bei denen der Zeitstempel des Sourceobjekts und des katalogisierten Objekts identisch ist) und für Maps.

Bei Data Areas entspricht dieses Unterkommando der Option FORMATTED.

Im List-Profil können Sie angeben, wie die Data Areas standardmäßig angezeigt werden sollen:

- formatiert, d.h. die Anzeige ähnelt der in einem DEFINE DATA-Statement oder
- unformatiert, d.h. die Anzeige ähnelt der im Data-Area-Editor.

In der List der Objekte können Sie das Unterkommando DEFINE-DATA ON/OFF benutzen, um den Standardwert für die Zeit zu setzen, während der das LIST-Kommando ausgeführt wird.

Wenn Data Areas standardmäßig formatiert angezeigt werden und wenn es nicht möglich ist, den Sourcecode der Data Area in das DEFINE DATA-Format umzusetzen, erscheint eine entsprechende Meldung und die Data Area wird unformatiert angezeigt.

Wenn Sie das Unterkommando FORMAT für eine Map eingeben, erscheint ein Fenster, in dem Sie eine oder mehrere zusätzliche Informationen auswählen können, die mit der Map angezeigt werden:

- **Map Settings** (entspricht dem Unterkommando SETTINGS).
- **Map Layout** (entspricht dem Unterkommando LAYOUT). Zusätzlich können Sie Füllzeichen für Eingabefelder (AD=A und AD=M) und Ausgabefelder (AD=O) angeben, um diese Felder sichtbar zu machen. Sie können ein beliebiges Zeichen als Füllzeichen angeben.
- **Field Summary** (entspricht dem Unterkommando FIELDS).
- **Processing Rules** (entspricht dem Unterkommando RULES).

Die von Ihnen gewählten Optionen werden nacheinander in der im Auswahlfenster vorhandenen Reihenfolge angezeigt.

Im FORMAT-Modus stehen dieselben Unterkommandos zum Blättern (Ausnahme: B) und die Unterkommandos FIELDS, LAYOUT, PRINT, RULES und SETTINGS zur Verfügung wie bei einer normal mit LIST angezeigten Source (siehe oben). Zusätzlich sind zu jeder Option die nachfolgend beschriebenen Unterkommandos verfügbar.

- Zusätzliche Unterkommandos für Map Layout
- Zusätzliche Unterkommandos für Field Summary List
- Zusätzliche Unterkommandos für Processing Rules

Zusätzliche Unterkommandos für Map Layout

S>*n*	Verschieben des Map Layouts um *n* Spalten nach rechts.
S<*n*	Verschieben des Map Layouts um *n* Spalten nachlinks.

Zusätzliche Unterkommandos für Field Summary List

EXTEND	Zeigt erweiterte Feld-Etitierinformationen zu allen Map-Feldern. Um die erweiterten Feld-Etitierinformationen zu einem einzelnen angezeigten Feld anzuzeigen, markieren Sie den Namen des Feldes in der Field Summary List mit dem Cursor und drücken Sie ENTER.
RULES *nn*	Zeigt die Verarbeitungsregeln, die mit dem Feld *nn* verknüpft sind (wobei *nn* für die fortlaufende Zeilennummer in der ersten Spalte der **Field Summary List** steht). Um die Verarbeitungsregeln zu einem einzelnen angezeigten Feld anzuzeigen, geben Sie ein R in der Kommandozeile ein, markieren Sie dann den Namen des Feldes in der Field Summary List mit dem Cursor und drücken Sie ENTER.

SCAN [ABSOLUTE] *value*	Wie bei ***Unterkommandos für eine angezeigte Source***.
SCAN =	

Zusätzliche Unterkommandos für Processing Rules

SCAN [ABSOLUTE] *value*	Wie bei ***Unterkommandos für eine angezeigte Source***.
SCAN =	

Cursor-sensitive Objektauswahl

Innerhalb einer gelisteten Source können Sie mit dem Cursor den Namen eines Objektes, das in dieser Source referenziert wird, markieren; die Source des so ausgewählten Objektes wird dann in einem Fenster gezeigt.

Für die in dem Fenster gezeigte Source stehen Ihnen dieselben Unterkommandos — außer `PRINT`, `EXPAND` und `ZOOM` — zur Verfügung wie für eine „normal“ gelistete Source

Individuelles List-Profil erstellen

Sie können sich ihr eigenes, individuelles Profil für das `LIST`-Kommando erstellen. Für diesen Zweck bietet Natural Ihnen das Textobjekt `LISTPROF` in der Library `SYSLIB`.

In `LISTPROF` können Sie allgemeine oder benutzerspezifische Profile mit entsprechenden Standardwerten eintragen, zum Beispiel für `COUNT-SOURCE-LINES`. Diese Standardwerte werden verwendet, wenn Sie das `LIST`-Kommando ausführen.

Um die in LISTPROF definierten Werte zu aktivieren

1. Kopieren Sie das Textobjekt `LISTPR-S` von der Library `SYSLIB` in eine beliebige Library.
2. Fügen Sie Ihre Änderungen hinzu.
3. Speichern Sie das Textobjekt `LISTPR-S` unter dem Namen `LISTPROF`.
4. Speichern Sie das Textobjekt `LISTPROF` in die Library `SYSLIB`.
5. Rufen Sie das `LIST`-Kommando auf.

Eine ausführliche Beschreibung finden Sie im Textobjekt `LISTPR-S` in der Library `SYSLIB`.

27 LISTDBRM

Anmerkungen:

1. This command is only available with Natural for DB2 and Natural for SQL/DS.
2. In case of Natural for DB2, `LISTDBRM` has to be issued from the Natural system library `SYSDB2`, which means you have to log on to `SYSDB2` first and then enter the command `LISTDBRM`.
3. In case of Natural for SQL/DS, `LISTDBRM` has to be issued from library `SYSSQL`, which means you have to log on to `SYSSQL` first and then enter the command `LISTDBRM`.

The `LISTDBRM` command is used to display either existing DBRMs (Natural for DB2) or existing packages (Natural for SQL) of Natural programs or Natural programs referencing a given DBRM.

When the command is used with Natural for DB2, the following menu is displayed:

```
10:56:20              ***** NATURAL Tools for SQL *****              2006-03-17
                                   - List DBRM -




                       Code Function

                        D   Display DBRMs of Programs
                        R   List Programs Referencing DBRM
                        ?   Help
                        .   Exit

                  Code .. _   Library .. EXAMPLE_
                              Member  .. _________
                              DBRM ..... _________
```

```
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                 Canc
```

When the command is used with Natural for SQL/DS, the following menu is displayed:

```
15:35:20                * * * LISDBRM Command * * *                2006-05-25




                     Code Function
                     ---- --------------------------------
                      D   Display DBRMs of Programs
                      R   List Programs Referencing DBRM
                      ?   Help
                      .   Exit
                     ---- --------------------------------
                Code .. _   Library .. EXAMPLE_
                            Member  .. ________
                            DBRM ..... ________



Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                 Canc
```

The following functions are available:

Code	Description
D	Displays programs that have access to DB2 (Natural for DB2) or SQL/DS (Natural for SQL/DS), and their corresponding package (DBRM). If no DBRM name is shown, the corresponding program uses dynamic SQL.
R	Lists all programs that use a given package (DBRM). If no DBRM name is specified, all programs that use dynamic SQL are listed.

The following parameters apply:

Parameter	Description
Library	Specifies the name of a Natural library. Library names beginning with `SYS` are not permitted.This parameter must be specified.
Member	Specifies the name of the Natural program (member) to be displayed. This parameter is optional and can be used to limit the output. If a value is specified followed by an asterisk (*), all members in the specified library with names beginning with this value are listed. If the **Member** field is left blank, or if an asterisk is specified only, all members in the specified library are listed.
DBRM	Specifies a valid package (DBRM) name. If left blank, programs that run dynamically are referenced. This parameter applies to function code `R` only.

Sample List DBRM Result Screen

```
11:15:45               ***** LISTDBRM Command *****                2006-03-17


    Library  Name     Type         DBRM       User ID   Date       Time
    -------- -------- -----------  --------   --------  --------   --------
    EXAMPLE  PROG1    Program      PACK1      SAG       2006-03-17 11:10:43
    EXAMPLE  PROG2    Program      PACK1      SAG       2006-03-17 11:10:48
    EXAMPLE  PROG3    Program      PACK2      SAG       2006-03-17 11:11:04
    EXAMPLE  PROG4    Program                 SAG       2006-03-17 11:11:07
```

28 LIST COUNT

Mit dem Kommando `LIST COUNT` können Sie sich die Anzahl der Natural-Objekte in Ihrer aktuellen Library anzeigen lassen.

LIST COUNT	Zeigt die Gesamtanzahl aller Objekte.
LIST COUNT *	Zeigt die Anzahl der Objekte aufgeschlüsselt nach Objekttypen.
LIST COUNT *name*<	Zeigt die Anzahl der Objekte, deren Namen kleiner/gleich *name* sind.
LIST COUNT *name*>	Zeigt die Anzahl der Objekte, deren Namen größer/gleich *name* sind.
LIST COUNT *name**	Zeigt die Anzahl der Objekte, deren Namen mit *name* anfangen.

Anmerkung: Sollten Objekte unter Objekttyp **undefined** aufgelistet sein, deutet dies darauf hin, dass die Library Objekte enthält, deren Version nicht kompatibel ist.

29 LIST XREF

```
LIST XREF
```

Das Kommando `LIST XREF` ist nur verfügbar, wenn Predict installiert ist.

Mit diesem Kommando können Sie sich alle aktiven Referenzdaten für die aktuelle Library anzeigen lassen.

Weitere Informationen siehe *List XREF for Natural* in der Predict-Dokumentation.

30 LISTSQL

```
LISTSQL [object-name]
```

Dieses Kommando ist nur mit Natural for DB2, Natural SQL Gateway und Natural for SQL/DS verfügbar.

Das Kommando `LISTSQL` listet diejenigen Natural-Statements im Sourcecode eines Natural-Objekts auf, die in Verbindung mit einem Datenbankzugriff stehen, sowie die entsprechenden SQL-Kommandos, in die sie übersetzt wurden.

Weitere Informationen zum `LISTSQL`-Kommando siehe

- *LISTSQL Command* im Teil *Natural for DB2* der *Database Management System Interfaces*-Dokumentation,
- *LISTSQL Command* im Teil *Natural SQL Gateway* der *Database Management System Interfaces*-Dokumentation,
- *LISTSQL Command* im Teil *Natural for SQL/DS* der *Database Management System Interfaces*-Dokumentation.

31 LISTSQL

This command is only available with Natural for DB2, Natural SQL Gateway, and Natural for SQL/DS. There are minor differences depending on whether the command is used with Natural for DB2, Natural SQL Gateway, or Natural for SQL/DS. These differences are marked accordingly in the following description.

```
LISTSQL [ { object-name } [ALL] ]
        [ { <sa>        }       ]
```

This command generates a list of those Natural statements in the source code of a programming object which are associated with a database access. Also, it displays the corresponding SQL commands these Natural statements have been translated into. This enables you to view the generated SQL code before executing a Natural program which accesses an SQL table.

Syntax Element	Description
object-name <sa>	If you specify a valid object name, the object to be displayed must be stored in the library to which you are currently logged on. If you do not specify an object name or if you specify <sa> (source area), LISTSQL refers to the object currently in the Natural source area. In any case, LISTSQL needs a cataloged or stowed object to perform its functionality.
ALL	If you specify the keyword ALL, the generated SQL statements of one object will be displayed in direct succession; that is, without scrolling. If you omit this keyword, the generated SQL statements contained in the specified object are listed one per page. You can use ALL in online mode and in batch mode. The output format will be the same. The functions **Error** (PF2), **Explain** (PF4) or **Parms** (PF6) are not available. When you specify ALL, you can use a question mark (?) or an asterisk (*) as wildcard character, for example: LISTSQL PGM* ALL. The special characters > and < are allowed, but only at the end of a string; this means, that, for example, ABC<DEF would be an invalid expression.

Sample LISTSQL Screen:

```
14:50:23               ***** NATURAL TOOLS FOR SQL *****            2009-12-04
Member DEM2SEL                    - LISTSQL -                   Library SYSDB243
SQL Builder Version 4.10
Natural statement  at line 0140                                   Stmt   1 / 1

  SELECT *
      INTO VIEW NAT-DEMO
      FROM NAT-DEMO

Generated SQL statement   Mode : dynamic  DBRM :                  Line   1 / 3
                                                                  Length  68
  SELECT  NAME, ADDRESS, DATEOFBIRTH, SALARY
  FROM    NAT.DEMO
  FOR FETCH ONLY




Command ===>                                        Queryno for EXPLAIN 1____


Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
           Error Exit  Expl        Parms                   Prev  Next  Canc
```

If a static DBRM has been generated, the name of this DBRM is displayed in the **DBRM** field of the **LISTSQL** screen; otherwise, the **DBRM** field remains empty. A static DBRM is only available with Natural for DB2 and Natural for SQL/DS.

The following screen-specific PF key functions are available:

PF Key (Label):	Function:
PF2 (Error)	This key executes the SQLERR command. If an error occurs during EXPLAIN, you can use this key to get information on DB2 or SQL/DS errors, respectively.
PF4 (Expl)	With this key, you can execute an EXPLAIN command for the SQL statement currently listed. The query number for the EXPLAIN command (in the field **Queryno for EXPLAIN**) is set to 1 by default, but you can overwrite this default. EXPLAIN is only supported for Natural for DB2 and Natural for SQL/DS.
PF6 (Parms)	You can use this key to display a further screen which lists all parameters from the SQLDA for the currently displayed SQL statement; see sample screen below.
PF10 (Prev), PF11 (Next)	Within the listed results, you can go from one listed SQL statement to another by pressing the corresponding key.

Sample Parameters Screen:

```
14:55:24               ***** NATURAL TOOLS FOR SQL *****           2009-12-04
Member DEM2SEL                      LISTSQL                    Library SYSDB243

        Mode : dynamic   DBRM :           Contoken :
                   (3rd/pre)
        static parms : (1st)
                       (2nd)
        SQLDA
                                    DBID : 250  FNR :   1  CMD : S1 0140 08
    Nr  Type     Length     CCSID
     1. CHAR             20       8001 0000 0014 01C4 0000 0000 0800 0000
     2. CHAR            100       8002 0000 0064 01C4 0000 0000 0800 0000
     3. CHAR             10       8003 0000 000A 01C4 0000 0000 0800 0000
     4. DECIMAL         6.2       8004 4000 0602 01E5 0000 0000 0800 0000









Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit                                                   Canc
```

In static mode, static information is also displayed, which includes the static DBRM name, the DB2 or SQL/DS consistency token, and some internal static parameters.

To navigate on the parameters screen, you can use the following PF keys, whose functions are assigned only if the information does not fit on the screen.

PF Key (Label):	Function:
PF6 (top,--), PF9 (bottom,++)	Using these keys, you can go directly to the top (--) or to the bottom (++) of the list.
PF8, PF7 (-)	Using these keys, you can scroll forwards (+) or backwards (-) by pressing the corresponding key.

Using the EXPLAIN Command with Natural for DB2

Wichtig: Before you use the DB2 `EXPLAIN` command, refer to the section *LISTSQL and Explain Functions* in the section *Special Requirements for Natural Tools for DB2* in the *Natural for DB2* documentation.

The `EXPLAIN` command provides information on the DB2 optimizer's choice of strategy for executing SQL statements. For the `EXPLAIN` command to be executed, a `PLAN_TABLE` must exist. The informa-

tion determined by the DB2 optimizer is written to this table. The corresponding explanation is read from the PLAN_TABLE and displayed via the **EXPLAIN Result** screen.

Sample **Explain Result** Screen:

```
 10:57:47               ***** NATURAL TOOLS FOR SQL *****            2009-12-03
 Queryno 1                       EXPLAIN Result                     Row  1 / 1

                      Estimated cost :   296.6  timerons

Qblock  Plan Mixop Acc. Match Index Pre-  Column- Access-
    No    No   seq type  cols only  fetch fn_eval Creator.Name
------ ----- ----- ---- ----- ----- ----- ------- ----------------------------
     1     1       R               S




       Table-                           Tslock        -- sortn --  -- sortc --
 TabNo Creator.Name                     mode   Method uq jo or gr  uq jo or gr
------ --------------------------------  ------ ------ -- -- -- --  -- -- -- --
     1 NAT.DEMO                         IS            N  N  N  N   N  N  N  N




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit  Info        More   -     +                     Canc
```

If an explanation does not fit on one screen, you can scroll backwards and forwards by pressing PF7 (-) or PF8 (+), respectively.

The value in the **Estimated cost** field is taken from SQLERRD (4) in the SQLCA; it is a rough estimate of the required resources.

With PF4 (Info), the additional information that is provided with the EXPLAINB command is displayed.

Using the EXPLAIN Command with Natural SQL Gateway

This command is not applicable.

Using the EXPLAIN Command with Natural for SQL/DS

LISTSQL enables you to use the SQL/DS command EXPLAIN, which provides information on the SQL/DS optimizer's choice of strategy for executing SQL statements.

Natural executes the EXPLAIN command for the SQL statement that is displayed on the LISTSQL screen.

The information determined by the SQL/DS optimizer is written into your `PLAN_TABLE`. Natural then reads the table and displays the contents.

Sample **EXPLAIN Result** Screen:

```
 10:22:07               ***** NATURAL TOOLS FOR SQL *****            2009-12-03
 Queryno 1                      EXPLAIN Result                      Row  1 / 1

                      Estimated cost :     3.3  timerons

            Qblockno                   Table
                  Planno Method Tabno creator     Tablename
              ---  ---   ------  ---  -------- ------------------
              1    1             1    NAT      DEMO




           Access  Access
            type   creator      Accessname      sort_new sort_comp
            ----   --------  ------------------ -------- ---------
             R                                    N          N




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit        Del          -     +                     Canc
```

32 LISTSQLB

Anmerkungen:

1. This command is only available with Natural for DB2.
2. Before you use the `LISTSQLB` command, refer to *LISTSQL and Explain Functions* in the section *Special Requirements for Natural Tools for DB2* in the *Natural for DB2* documentation.

The command `LISTSQLB` can be executed in batch mode or issued online from the Natural `NEXT` prompt.

If executed online, the following screen is invoked:

```
10:54:35              ***** NATURAL Tools for SQL *****            2006-03-17
                                  - LISTSQLB -




                         Code Function

                          X   Explain all SQL Statements
                          .   Exit

                    Code .. _   Member ... ________
                                Queryno .. 1____
```

```
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit                                                   Canc
```

By specifying a valid member name, the explanation of SQL statements can be limited to certain member(s); an asterisk (*) can be used for range specification:

- If you specify a unique member name, all SQL statements contained in this member are explained;
- If you specify a value followed by an asterisk, all SQL statements contained in all members with names beginning with the specified value are explained;
- If you specify an asterisk only (or leave the field blank), all SQL statements of all existing SQL members are explained.

A query number must be specified, so that with each issued `EXPLAIN` command, the newly created explanation is added to the appropriate query number. The default query number is `1`.

To issue the `EXPLAIN` command, enter function code `X` and specify a valid member name and query number; all SQL statements contained in the specified member(s) are explained.

If `LISTSQLB` is executed online, the following screen informs you about the processing status of the command and if any errors have occurred.

```
10:55:24              ***** NATURAL Tools for SQL *****             2006-03-17
                                 - LISTSQLB -
    Queryno : 1                              Member  Stmtno Message

    Current Object :
    Library         TEST
    Member          RTTB--IN

    Statistics :
    Members read    1
       with SQL     1
    SQL statements  7

       Member   Message




       RTTB--IN OK

Press Enter to continue
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
```

If executed in batch mode, error messages are written to a dataset referred to by DD name `CMPRINT` (logical printer 0).

33 LOGOFF

LOGOFF

Verwandtes Kommando: LOGON

Mit dem Systemkommando LOGOFF erreichen Sie, dass die Library-ID auf SYSTEM und das Adabas-Passwort auf Leerzeichen gesetzt wird. Der Source-Inhalt des Editor-Arbeitsbereichs bleibt erhalten.

Das LOGOFF-Kommando hat keinen Einfluß auf die Werte der globalen Parameter.

Informationen zur LOGOFF-Verarbeitung unter Natural Security siehe *How to End a Natural Session* im Abschnitt *Logging On* in der *Natural Security*-Dokumentation.

Anmerkung: Mit dem LOGOFF-Kommando beenden Sie nicht die Natural-Session. Um eine Session zu beenden, verwenden Sie das Systemkommando FIN oder führen ein Programm aus, das ein TERMINATE-Statement enthält.

34 LOGON

```
LOGON library-id [password]
```

Verwandtes Kommando: LOGOFF

Mit dem Systemkommando LOGON wählen Sie eine bestimmte Library. In dieser Library werden dann alle von Ihnen während der Session in Source- und/oder Objektform erstellten Objekte gespeichert (es sei denn, Sie geben bei einem SAVE-, CATALOG- oder STOW-Kommando ausdrücklich eine andere Library an).

Der Source-Inhalt des Editor-Arbeitsbereichs bleibt bei einem LOGON-Kommando unverändert.

Das Kommando LOGON bewirkt, dass alle Natural Global Data Areas und anwendungsunabhängigen Variablen (AIVs), alle mit einem SET KEY-Statement vorgenommenen Zuordnungen und alle gehalten ISN-Listen freigegeben werde. Im DDM-Buffer-Bereich enthaltene DDMs werden ebenfalls freigegeben.

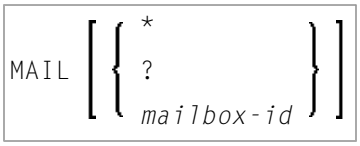
Siehe auch *Library Naming Conventions* in der *Using Natural*-Dokumentation.

<table>
<tr><td>LOGON library-id</td><td colspan="2">Eine Library-ID darf 1 bis 8 Zeichen lang sein und keine Leerzeichen enthalten. Sie darf folgende Zeichen enthalten:<table><tr><td>A - Z</td><td>Großbuchstaben</td></tr><tr><td>0 - 9</td><td>Ziffern</td></tr><tr><td>-</td><td>Bindestrich</td></tr><tr><td>_</td><td>Unterstrich</td></tr></table>Das erste Zeichen einer Library-ID muß ein Großbuchstabe sein.</td></tr>
<tr><td>LOGON library-id

password</td><td colspan="2">Das Adabas-Passwort; siehe Session Parameters im Abschnitt Library Maintenance der Natural Security-Dokumentation.</td></tr>
</table>

Unter Natural Security gelten für das LOGON-Kommando bestimmte Regeln und Einschränkungen. Diese erfahren Sie von Ihrem Administrator oder finden Sie im Abschnitt *Logging On* in der *Natural Security*-Dokumentation.

35 MAIL



Dieses Kommando ist nur verfügbar, wenn Natural Security installiert ist.

Mit dem Systemkommando `MAIL` können Sie eine Mailbox aufrufen, um den Inhalt sowie das Ablaufdatum zu ändern.

Eine Mailbox ist eine Art „Schwarzes Brett“ zur Übermittlung von Informationen, das mit Natural Security definiert wird.

MAIL	Wenn Sie das `MAIL`-Kommando ohne Parameter eingeben, erscheint ein Fenster, in dem Sie aufgefordert werden eine Mailbox-ID anzugeben.
MAIL *	Es wird eine Liste der Mailboxen angezeigt, die Sie benutzen können. Sie können aus der Liste eine Mailbox auswählen.
MAIL ?	
MAIL *mailbox-id*	Wenn Sie eine *mailbox-id* angeben (bis zu 8 Zeichen lang), wird die entsprechende Mailbox direkt aufgerufen. Die *mailbox-id*, die Sie angeben, muß in Natural Security definiert sein.

Weitere Informationen siehe *Mailboxes* in der *Natural Security*-Dokumentation.

36 MAINMENU

Mit dem Systemkommando MAINMENU können Sie die Anzeige des Natural-Hauptmenüs ein- bzw. ausschalten oder ein Programm aufrufen, das einen eigenen Menüschirm anzeigt.

Das Kommando ist in der Kommandozeile einer Remote-Entwicklungsumgebung verfügbar.

MAINMENU	Die Eingabe des MAINMENU-Kommandos ohne Parameter ist identisch mit MAINMENU ON. Dies ist der Standardwert.
MAINMENU ON	
MAINMENU OFF	Schaltet den Hauptmenü-Modus aus.
MAINMENU *user-program*	Statt des Natural-Hauptmenüs wird ein benutzerdefiniertes Programm aufgerufen, das seinerseits ein benutzerdefiniertes Menü aufruft.

Siehe auch Natural-Profilparameter MENU.

37 NOCOPT

NOCOPT

Mit dem Kommando NOCOPT können Sie verschiedene beim Start von Natural gesetzte Optionen für den Natural Optimizer Compiler anzeigen und ändern.

Weitere Informationen zu NOCOPT finden Sie im Abschnitt *Activating the Optimizer Compiler* in der *Natural Optimizer Compiler*-Dokumentation.

38 NOCSHOW

NOCSHOW

Das Kommando NOCSHOW liefert Pufferspeicher-Informationen zur Ausgabe der PGEN-Option des Natural Optimizer Compiler.

Weitere Informationen zu NOCSHOW finden Sie im Dokument *Optimizer Options*, Abschnitt *Output of the PGEN Option*, in der *Natural Optimizer Compiler*-Dokumentation.

39 NOCSTAT

NOCSTAT

Das Kommando NOCSTAT liefert Statistikdaten zu Programmen, die für die Verarbeitung durch den Natural Optimizer Compiler geeignet sind.

Weitere Informationen zu NOCSTAT siehe *NOCSTAT Command* in der *Natural Optimizer Compiler*-Dokumentation.

40 PROFILE

Dieses Kommando ist nur verfügbar, wenn Natural Security installiert ist.

```
PROFILE
```

Mit dem Systemkommando `PROFILE` können Sie sich Ihr gegenwärtig gültiges Benutzerprofil anzeigen lassen. Dieses Profil informiert Sie über die Bedingungen und Voraussetzungen, die in Ihrer aktuellen Natural-Umgebung für Sie gelten.

Weitere Informationen siehe *PROFILE Command* in der *Natural Security*-Dokumentation.

41 READ

```
READ object-name [library-id]
```

Verwandtes Kommando: EDIT

Mit dem Kommando READ können Sie ein in Sourceform gespeichertes Objekt in den Arbeitsbereich des entsprechenden Editors einlesen.

Vorsicht: Ein bereits im Editor befindliches Objekt wird dadurch überschrieben.

Siehe auch *Namenskonventionen für Objekte* in der Dokumentation *Natural benutzen*.

READ *object-name*	Der Name des gewünschten Objekts. Wenn Sie den *object-name* ohne Library-ID angeben, wird das gewünschte Objekt nur dann eingelesen, wenn es in Ihrer aktuellen Library gespeichert ist.
READ *object-name* *library-id*	Falls das gewünschte Objekt nicht in Ihrer aktuellen Library, sondern in einer anderen gespeichert ist, müssen Sie die Library-ID dieser Library angeben. Wenn Sie sowohl den *object-name* und die *library-id* angeben, liest Natural das Objekt nur dann ein, wenn es in der angegebenen Library gespeichert ist.

42 RENAME

```
RENAME [old-name [new-name [new-type]]]
```

Mit dem Kommando RENAME können Sie ein Natural-Objekt umbenennen und außerdem seinen Objekttyp ändern.

Sie können jeweils nur ein Objekt zur Zeit umbenennen. Das Objekt muß in der Library, in der Sie sich gerade befinden, gespeichert sein. Um Inkonsistenzen zu vermeiden, benennt Natural jede bestehende Form (Sourcecode, Objektmodul oder beides) des Objekts um.

Siehe auch *Namenskonventionen für Objekte* in der Dokumentation *Natural benutzen*.

RENAME	Wenn Sie das Kommando ohne Parameter absetzen, erscheint ein **Rename Object**-Fenster, in dem Sie dieselben Parameter wie in der Kommandozeile angeben können.
old-name	Als *old-name* geben Sie den derzeitigen Namen des Objekts an.
new-name	Als *new-name* geben Sie den neuen Namen an, unter dem das Objekt von nun an gespeichert sein soll.
new-type	Wenn Sie ein Objekt umbenennen, haben Sie außerdem die Möglichkeit, seinen Objekttyp zu ändern, indem Sie das entsprechende Zeichen für den *new-type* angeben. Dies ist allerdings nur bei Objekten möglich, die lediglich in Sourceform bestehen. Möliche Werte für *new-type*:

4	Klasse
5	Processor
8	Adapter
9	Resource
A	Parameter Data Area
C	Copycode
G	Global Data Area

	H	Helproutine
	L	Local Data Area
	M	Map
	N	Subprogramm
	O	Macro
	P	Programm
	S	Subroutine
	T	Text
	Y	Rule
	Z	Recording

43 RENUMBER

RENUMBER [*(n)*]

Mit dem Kommando RENUMBER erreichen Sie, dass die Sourcecodezeilen des Objekts, das sich gerade im Programm-Editor befindet, neu durchnumeriert werden.

RENUMBER	Standardmäßig, d.h. wenn Sie das Kommando ohne Parameter eingeben, wird in 10er-Schritten nummeriert.
RENUMBER (*n*)	Mit *n* (eine Zahl von 1 bis 10) können Sie angeben, um wieviel eine Zeilennummer jeweils höher als die vorherige sein soll.

44 RETURN

Mit dem Kommando RETURN können Sie zu einer bestimmten vorherigen Natural-Anwendung (oder der Ausgangsanwendung) zurückkehren.

Programmierschnittstelle (API): USR1026N. Siehe *SYSEXT - Natural Application Programming Interfaces* in der *Utilities*-Dokumentation.

RETURN	Wenn Sie RETURN ohne Parameter eingeben, wird die Kontrolle an die vorherige Anwendung übergeben. Eine vorherige Anwendung wird mit dem Systemkommando SETUP als solche definiert. Alle (mit SETUP spezifizierten) Informationen über diese vorherige Anwendung werden gelöscht. Ist keine vorherige Anwendung definiert, erfolgt ein Rücksprung zur Ausgangsanwendung. Wenn Sie RETURN eingeben und kein Rückkehrpunkt definiert ist, wird das RETURN-Kommando ignoriert. **Unter Natural Security:** Ein LOGOFF-Kommando wird ausgeführt, wenn Sie RETURN eingeben und kein Rückkehrpunkt definiert ist.
RETURN I	Mit diesem Kommando übergeben Sie die Kontrolle direkt an die Ausgangsanwendung (I = Initial Application). Alle Informationen über vorherige Anwendung (außer der Ausgangsanwendung) werden dabei gelöscht.
RETURN *nn*	Mit diesem Kommando übergeben Sie die Kontrolle an die *nn*-te vorherige Anwendung. Alle Informationen über die der *nn*-ten Anwendung nachfolgenden Anwendungen werden dabei gelöscht.
RETURN *	Dieses Kommando zeigt Ihnen eine Liste aller gegenwärtig definierten Rückkehrpunkte an. Auf der Liste können Sie dann den gewünschten Rückkehrpunkt auswählen.

Siehe Systemkommando SETUP für Beispiele und weitere Informationen.

45 ROUTINES

ROUTINES

Mit dem Kommando ROUTINES können Sie feststellen, welche katalogisierten Objekte in Ihrer aktuellen Library welche externen Subroutinen verwenden.

Sie erhalten eine Liste aller Objekte in der aktuellen Library, wobei für jedes Objekt die Namen der von ihm aufgerufenen externen Subroutinen angezeigt werden, sowie die Objektnamen der Subroutinen, in denen diese externen Subroutinen enthalten sind.

Handelt es sich bei dem Objekt selbst um eine Subroutine, Klasse oder Function, wird der Name der in ihr enthaltenen Subroutine, Klasse oder Function angezeigt.

46 RPCERR

RPCERR

Mit dem Kommando RPCERR können Sie sich die Nummer und die Meldung des zuletzt aufgetretenen Natural-Fehlers anzeigen lassen, falls er in Bezug zum RPC steht. Außerdem wird der Reason Code des Brokers und die zugehörige Meldung angezeigt. Darüber hinaus kann der Node- und Server-Name aus dem letzten Broker-Aufruf abgerufen werden.

Weitere Informationen siehe *Monitoring the Status of an RPC Session* im Abschnitt *Operating a Natural RPC Environment* in der *Natural Remote Procedure Call (RPC)*-Dokumentation.

47 RUN

RUN [REPEAT] [*program-name* [*library-id*]]

Das Kommando RUN dient dazu, ein Source-Programm zu kompilieren und auszuführen. Das auszuführende Programm kann sich entweder in der Natural-Systemdatei oder im Arbeitsbereich des Editors befinden.

Siehe auch:

Natural-Compiler in Natural *System-Architektur*
Namenskonventionen für Objekte in *Natural benutzen*

RUN	Wenn Sie den *program name* nicht angeben, dann kompiliert Natural das Programm, das sich gegenwärtig im Editor-Arbeitsbereich befindet, und führt es aus.
REPEAT	Wenn das auszuführende Programm mehrere Ausgabeschirme erzeugt und Sie möchten, dass die Schirme unmittelbar nacheinander (d.h. ohne zwischengeschaltete Eingabezeilen) ausgegeben werden, müssen Sie zusätzlich zum Schlüsselwort RUN das Schlüsselwort REPEAT angeben. Nach Beendigung des Programms wechselt Natural in den Kommando-Modus.
program-name	Der Name des Programms, das kompiliert und ausgeführt werden soll. Wenn Sie den *program-name* ohne Library-ID angeben, wird das Programm in den Arbeitsbereich gelesen (wobei ein bereits im Arbeitsbereich befindliches Objekt überschrieben wird), kompiliert und ausgeführt, vorausgesetzt es ist in Ihrer aktuellen Library gespeichert. Wenn es nicht unter der aktuellen Library-ID gespeichert ist, erscheint eine Fehlermeldung.
library-id	Die Library in der sich das gewünschte Programm befindet. Falls Sie ein Programm kompilieren und ausführen wollen, das nicht in Ihrer aktuellen Library gespeichert ist, müssen Sie zusätzlich zum Programmnamen die Library-ID dieser Library angeben.

	Wenn Sie sowohl den *program-name* und die *library-id* angeben, dann wird das angegebene Programm nur dann kompiliert und ausgeführt, wenn es unter der angegebenen Library-ID gespeichert ist. Wenn es nicht unter der aktuellen Library-ID gespeichert ist, erscheint eine Fehlermeldung. Eine Library-ID, die mit `SYS` beginnt, darf nicht angegeben werden (Ausnahme: `SYSTEM`). Wenn Natural Security aktiv ist, können Sie keine Library-ID angeben, d.h. Sie können nur Programme aus ihrer aktuellen Library kompilieren/ausführen.

48 SAVE

```
SAVE [object-name [library-id]]
```

Verwandte Kommandos: STOW | CATALOG

Das Kommando SAVE dient dazu, ein Source-Objekt in der Natural-Systemdatei zu speichern. Der Inhalt des Editor-Arbeitsbereichs wird dadurch nicht beeinflußt.

Siehe auch:

Namenskonventionen für Objekte in *Natural benutzen*
Natural-Compiler in Natural *System-Architektur* als Hintergrundinformation

Vorsicht: Wird das SAVE-Kommando zurückgewiesen, bedeutet dies, dass der Parameter RECAT auf ON gesetzt ist und das Programm, das Sie in Sourceform speichern wollen, bereits in kompilierter Form als Objektmodul gespeichert ist. In diesem Falle geben Sie das Systemkommando STOW ein, mit dem Sie das Programm gleichzeitig in Source- und Objektform speichern und so Diskrepanzen zwischen beiden Formen vermeiden.

SAVE	Wenn Sie das Kommando ohne *object-name* benutzen, wird das im Arbeitsbereich des Editors befindliche Objekt in der aktuellen Library gespeichert. Vorhandener Source-Code wird ersetzt.
SAVE *object-name*	Ein neues Objekt wird erstellt. Als *object-name* geben Sie den Namen an, unter dem das neue Objekt gespeichert werden soll. Falls ein Objekt dieses Namens schon vorhanden ist, wird das Kommando zurückgewiesen.
SAVE *object-name* *library-id*	Wenn Sie keine Library-ID angeben, wird das Programm in der Library, in der Sie gerade arbeiten, gespeichert. Wollen Sie es in einer anderen Library speichern, müssen Sie die Library-ID dieser Library angeben. Es ist allerdings nur möglich, ein Programm in einer anderen Library zu speichern, falls dort noch kein Objekt gleichen Namens gespeichert ist. Wenn Sie ein Objekt unter einem anderen Namen oder ein neu erstelltes Objekt speichern, wird das Objekt standardmäßig in der aktuellen Library gespeichert. Wollen

	Sie es in einer anderen Library speichern, müssen Sie nach dem *object-name* die *library-id* dieser Library angeben. Es wird ein neues Objekt angelegt. Falls ein Objekt dieses Namens schon vorhanden ist, wird das Kommando zurückgewiesen.

49 SCAN

SCAN

Mit dem Kommando SCAN können Sie den Sourcecode von Objekten nach einer bestimmten Zeichenkette absuchen; darüber hinaus besteht die Möglichkeit, die gesuchte Zeichenkette durch eine andere Zeichenkette zu ersetzen.

Sie können ein oder mehrere oder alle Objekte einer Library absuchen; hierbei können Sie bestimmte Objekte und Objekttypen auswählen.

Wichtig: Da das SCAN-Kommando den Sourcecode im Arbeitsbereich beeinflußt, sollten Sie vor der Verwendung des SCAN-Kommandos den Inhalt des Arbeitsbereichs gegebenenfalls mit einem SAVE- oder STOW-Kommando speichern.

Menü-Optionen

Wenn Sie das Kommando SCAN eingeben, wird das SCAN-Menü angezeigt, das Ihnen folgende Funktionen bietet:

Feld	Eingabe	
Code	T	**Statistics** Diese Funktion liefert Ihnen folgende Informationen: ■ die Anzahl der Objekte, die durchsucht wurden; ■ die Anzahl der Objekte, in denen der gesuchte Wert gefunden wurde; ■ die Anzahl der Sourcecode-Zeilen, in denen der gesuchte Wert gefunden wurde.
	L	**List of Objects Containing Scan Value** Diese Funktion liefert Ihnen eine Liste aller Objekte, in denen der gesuchte Wert gefunden wurde. Von der Liste können Sie dann einzelne Objekte zur weiteren Verarbeitung auswählen. Bei Bedarf können Sie die Zeilen mit den entsprechenden ***SCAN-Unterkommandos*** verändern. Mit dem SCAN-Editor können Sie alle Objekttypen, außer Maps und Data Areas und gesperrte Objekte (siehe *Locking of Source Objects*), verändern, und zwar im Full-Screen-Modus oder zeilenorientiert. Dadurch haben Sie die Möglichkeit, alle Zeilen zu editieren, nicht nur die mit dem SCAN-Kommando gefundenen. Sobald Sie ein Objekt fertig editiert haben, sollten Sie es speichern, bevor Sie vom normalen Editieren zur Fortsetzung der

Feld	Eingabe	
		SCAN-Operation zurückkehren. Bevor die SCAN-Verarbeitung einen anderen Editor aufruft, erhalten Sie eine Aufforderung, die bis dato mit dem SCAN-Editor gemachten Änderungen zu bestätigen.
	S	**Object Lines with Scan Value** Diese Funktion zeigt Ihnen nacheinander die Sourcecode-Zeilen an, in denen der gesuchte Wert gefunden wurde. Bei Bedarf können Sie die Zeilen mit den entsprechenden ***SCAN-Unterkommandos*** verändern. Mit dem SCAN-Editor können Sie alle Objekttypen, außer Maps und Data Areas und gesperrte Objekte (siehe *Locking of Source Objects*), verändern, und zwar im Full-Screen-Modus oder zeilenorientiert. Dadurch haben Sie die Möglichkeit, alle Zeilen zu editieren, nicht nur die mit dem SCAN-Kommando gefundenen. Sobald Sie ein Objekt fertig editiert haben, sollten Sie es speichern, bevor Sie vom normalen Editieren zur Fortsetzung der SCAN-Operation zurückkehren. Bevor die SCAN-Verarbeitung einen anderen Editor aufruft, erhalten Sie eine Aufforderung, die bis dato mit dem SCAN-Editor gemachten Änderungen zu bestätigen.
Scan value	Der Wert, nach dem gesucht werden soll. **Anmerkung:** Bevor Sie das SCAN-Kommando verwenden, achten Sie bitte darauf, dass die automatische Umsetzung von Klein- in Großbuchstaben je nach Bedarf ein- bzw. ausgeschaltet ist. Benutzen Sie dazu das Terminal-Kommando %L.	
Replace value	Der Wert, durch den der gesuchte Wert ersetzt werden soll. Bei Maps, Data Areas, Recordings, Dialogen und gesperrten Objekten (siehe *Locking of Source Objects*) können Sie den gesuchten Wert nicht durch einen anderen ersetzen.	
Library	Die ID der Library, in der gesucht werden soll; standardmäßig ist dies Ihre aktuelle Library. Wenn Sie als Library SYSTEM angeben, wird die Library in der FUSER-Datei abgesucht. Wenn Sie den Namen einer anderen Library, die mit SYS anfängt, angeben, wird die Library in der FNAT-Datei abgesucht.	
Object name	Das zu durchsuchende Objekt.	
	Leerzeichen	Alle Objekte der Library.
	*	
	object-name>	Alle Objekte, deren Namen größer oder gleich *object-name* sind.
	object-name<	Alle Objekte, deren Namen kleiner oder gleich *object-name* sind.
	Sie können den Bereich der zu durchsuchenden Objekte auch einschränken, indem Sie Stern-Notation (*) und/oder Wildcard-Notation (?) für den Objektnamen verwenden, und zwar in der gleichen Weise wie beim Systemkommando LIST beschrieben. Siehe auch *Namenskonventionen für Objekte* in der Dokumentation *Using Natural*.	

Feld	Eingabe	
Object type(s)	Sie können die Suche auf bestimmte Objekttypen beschränken. Eine Auswahlliste der möglichen Objekttypen erhalten Sie, wenn Sie ein Fragezeichen (?) in dieses Feld eingeben. Wenn Sie dieses Feld leer lassen oder einen Stern (*) eingeben, ist die Suche nicht auf bestimmte Objekttypen beschränkt.	
Absolute scan	Y	Es wird eine „absolute“ Suche durchgeführt, d.h. der zu suchende Wert wird in jeder möglichen Form gesucht, also auch als Teil einer längeren Zeichenkette.
	N	Standardmäßig ist die Suche nicht „absolut“. **Anmerkung:** Bei Data Areas erfolgt, unabhängig von der Eingabe in diesem Feld, immer eine „absolute“ Suche.
Selection list	Y	Liefert Ihnen gemäß Ihren Angaben von Name, Typ(en) für Code `T` oder `S` (siehe oben) zunächst eine Liste von Objekten, von denen Sie einzelne (durch Markieren mit einem beliebigen Zeichen) zur `SCAN`-Verarbeitung auswählen können.
	N	Es wird keine Auswahlliste angezeigt. Dies ist die Standardeinstellung.
Trace	Y	Die Trace-Funktion ist aktiviert.
	N	Die Trace-Funktion ist nicht aktiviert. Dies ist die Standardeinstellung.

SCAN Subcommands

In der (den) Kommandozeile(n) des Ergebnis-Schirms können Sie folgende Unterkommandos verwenden:

Kommando	Funktion
Leerzeichen	Fortsetzen der normalen `SCAN`-Verarbeitung.
`Q`	Beenden der `SCAN`-Verarbeitung.
.	
`E`	Objekt editieren (im Full-Screen-Editor).
`EDT`	Objekt editieren (zeilenorientiert).
`LIST`	Anzeigen des Objekts, so wie es sich gegenwärtig im Editor-Arbeitsbereich befindet.
`LET`	Ignorieren aller Zeilenveränderungen, die erfolgt sind, seit das letztemal EINGABE gedrückt wurde.
`I`	Ignorieren des gegenwärtig abgesuchten Objekts, etwaige Änderungen des Objekts werden nicht gespeichert, die `SCAN`-Verarbeitung fährt mit dem nächsten Objekt fort.

Kommando	Funktion
.D	(= Delete) Löschen einer Zeile. Neben der Zeile wird durch ein D angezeigt, dass sie gelöscht wurde.
.L	Ignorieren aller Veränderungen, die erfolgt sind, seit das letzte Mal EINGABE gedrückt wurde. Alle vorher mit .D gelöschten Zeilen werden ebenfalls wieder eingefügt.

Editier-Regeln

- Die Zeilenlänge eines Source-Objekts auf dem Ergebnis-Schirm darf 72 Stellen nicht überschreiten. Zeilen mit mehr als 72 Zeichen werden mit einem L markiert und können nicht geändert werden.
- Wenn die **Replace value**-Option verwendet und/oder ein Objekt auf dem Ergebnis-Schirm verändert wird, wird das Objekt automatisch gesichert, es sei denn, Sie geben eines der Unterkommandos I, Q oder einen Punkt (.) ein, bevor Sie das nächste Objekt mit SCAN bearbeiten.
- Zeilen, die PASSW, PASSWORD=, CIPHER= oder CIPH= enthalten, werden bei der SCAN-Verarbeitung ignoriert.

SCAN-Schlüsselwörter

Sie können die SCAN-Funktionen auch direkt aufrufen (im Online- und im Batch-Betrieb), und zwar durch die Angabe von Schlüsselwörtern:

Schlüsselwort	Erklärung
FUNC	Functionscode
SVAL	Zu suchender Wert
LIB	Library
RVAL	Einzusetzender Wert
OBJ	Objektname
TYPE	Objekttyp
ABSOL	„Absolute“ Suche

Vorsicht: Um unerwünschte Ergebnisse beim Suchen/Ersetzen zu vermeiden, sollten Sie in Direktkommandos keine Suchwerte angeben, die Leerzeichen enthalten. Im Online-Betrieb ist die Suche nach einem Wert, der Leerzeichen enthält, nur aus dem **SCAN-Menü** möglich.

Beispiele für SCAN-Kommandos mit Schlüsselwörtern:

```
SCAN FUNC=S,SVAL=value,LIB=SYSTEM,OBJ=PGMO*,TYPE=S
```

```
SCAN FUNC=S,SVAL=value,RVAL=value,OBJ=PGM1
```

SCAN im Batch-Betrieb

Das SCAN-Kommando verarbeitet nur eine Funktion pro Aufruf, um die möglichen nachteiligen Auswirkungen, die durch die Eingabe ungültiger Daten entstehen können, so gering wie möglich zu halten. Sie können entweder Schlüsselwörter (siehe **oben**) oder positionelle Parameter verwenden; letztere werden in folgender Reihenfolge eingegeben:

Positionelle Parameter werden wie folgt verarbeitet:

```
SCAN func, scan-value, replace-value, library, object-name, object-type, absolute
```

Die möglichen Werte für positionelle Parameter sind dieselben wie unter ***Menü-Optionen*** beschrieben.

Wichtig: Um nach einem Wert zu suchen, der Zeichen in Kleinschreibung oder eingebettete Leerzeichen enthält, dürfen Sie den *scan-value* nicht zusammen mit dem SCAN-Kommando in derselben Zeile des Batch-Jobs angegeben, sondern müssen in einer separaten Datenzeile eintragen und dabei die Daten entsprechend der Darstellung auf dem Online-Schirm anordnen, siehe ***Menü-Optionen***.

Beispiel für einen Suchen-/Ersetzen-Wert mit eingebetteten Leerzeichen:

```
SCAN S,MOVE LEFT,MOVE RIGHT,SYSTEM,PGMO*,N,*,N,N
```

SCAN unter Natural Security

Wenn Natural Security aktiv ist, können Sie SCAN nur verwenden, wenn Sie die Systemkommandos LIST, EDT, EDIT und READ in Ihrer aktuellen Library verwenden dürfen. Falls Sie die **Replace value**-Option verwenden wollen oder falls die Source änderbar sein soll, muß außerdem das Systemkommando SAVE zugelassen sein.

Unter Natural Security kann es sein, dass Sie SCAN in manchen Libraries nicht verwenden dürfen.

Wenn in der Library nur Structered Mode zugelassen ist, können Sie im Reporting Mode geschriebene Objekte durchsuchen, aber nicht verändern.

50 SCRATCH

Dieses Kommando wird nur aus Kompatibilitätsgründen unterstützt. Es empfiehlt sich, statt des SCRATCH-Kommandos das DELETE-Kommando zu verwenden, da DELETE mehr Flexibilität sowie Schutz gegen versehentliches Löschen bietet.

51 SETUP

```
SETUP [application-name] [command-name] [I]
```

Mit dem Systemkommando SETUP können Sie Rückkehrpunkte setzen, zu denen Sie dann später mit dem Systemkommando RETURN zurückkehren können. Das erlaubt es Ihnen, während einer Natural-Session problemlos von einer Anwendung in eine andere zu gelangen.

Programmierschnittstelle (API): USR1026N. Siehe *SYSEXT - Natural Application Programming Interfaces* in der *Utilities*-Dokumentation.

Syntax-Erklärung

Dieser Abschnitt beschreibt die Kommando-Syntax und die Parameter, die Sie mit dem SETUP-Kommando absetzen können. Wenn Sie einen Parameter auslassen wollen, müssen Sie das Eingabe-Begrenzungszeichen benutzen, um den Anfang des nächsten Parameters oder der nächsten Parameter zu markieren.

SETUP	Falls Sie SETUP ohne Parameter eingeben, erhalten Sie ein Menü, auf dem Sie die entsprechenden Informationen über einen Rückkehrpunkt eingeben können.
application-name	Der Name der Anwendung, an die die Kontrolle übergeben werden soll. Der Name darf bis zu 8 Stellen lang sein (Format/Länge A8). Falls Sie keinen *application-name* angeben, wird kein LOGON-Kommando ausgeführt. Dadurch haben Sie die Möglichkeit, mehrere Rückkehrpunkte innerhalb einer Anwendung zu setzen. Falls Sie als *application-name* einen Stern (*) eingeben, wird der Inhalt der Natural-Systemvariablen *LIBRARY-ID (zum Zeitpunkt des SETUP-Kommandos) verwendet, um damit ein LOGON-Kommando auszuführen, sobald das RETURN-Kommando benutzt wird.
command-name	Der Name des Kommandos, das ausgeführt werden soll, sobald die Kontrolle an die Anwendung übergeben wird. Der Name darf bis zu 60 Stellen lang sein (Format/Länge A60). Wenn Sie kein Kommando angeben, wird nach dem LOGON-Kommando kein weiteres ausgeführt. Dies empfiehlt sich, wenn zu einer Anwendung zurückgekehrt wird, für die in Natural Security eine Startup-Transaktion definiert ist. Falls Sie als Kommando einen Stern (*) eingeben, wird der Inhalt der Natural-Systemvariablen *STARTUP (zum Zeitpunkt des SETUP-Kommandos) verwendet und als Kommando ausgeführt, sobald das RETURN-Kommando benutzt wird.
I	Wenn Sie ein I eingeben, werden alle mit vorherigen SETUP-Kommandos definierten Rückkehrpunkte gelöscht und die mit diesem SETUP I-Kommando angegebene Anwendung als neue Ausgangsanwendung definiert.

	In einer Nicht-Security-Umgebung gilt: Wenn Sie aus der Library SYSTEM heraus in eine andere Library wechseln und es ist noch kein Rückkehrpunkt definiert, dann wird diese andere Library automatisch als Ausgangsanwendung (Initial Return Point) definiert.

Beispiel für SETUP/RETURN

1. Der Benutzer beginnt eine Natural-Session und gelangt in die als Standardanwendung definierte Anwendung APPL1.

   Auf Stufe 1 wird APPL1 als Rückkehrpunkt definiert.
2. Der Benutzer begibt sich mit dem Kommando LOGON APPL2 in eine andere Anwendung.
3. Der Benutzer führt ein Programm aus, das zwei Kommandos ausführt:

   SETUP *,MENU (Definieren eines Rückkehrpunktes)

   LOGON APPL3 (Wechsel zu einer anderen Anwendung)

   Auf Stufe 2 wird APPL2, STARTUP MENU als Rückkehrpunkt definiert.
4. Der Benutzer wechselt mit dem Kommando LOGON APPL4 in eine andere Anwendung.
5. Der Benutzer drückt die PF-Taste, die mit dem Kommando RETURN belegt ist. Natural führt folgende Kommandos aus:

   ```
   LOGON APPL2

   MENU
   ```

   Der Benutzer kehrt zu APPL2 zurück; der auf Stufe 2 gesetzte Rückkehrpunkt wird gelöscht.
6. Der Benutzer führt ein Programm aus, das zwei Kommandos ausführt:

   ```
   SETUP *,MENU

   LOGON APPL5
   ```

   Auf Stufe 2 wird APPL2, STARTUP MENU als Rückkehrpunkt definiert.
7. Der Benutzer führt ein Programm aus, das zwei Kommandos ausführt:

   ```
   SETUP *,MENU

   LOGON APPL6
   ```

   Auf Stufe 3 wird APPL5, STARTUP MENU als Rückkehrpunkt definiert.
8. Der Benutzer führt ein Programm aus, das zwei Kommandos ausführt:

```
SETUP *,MENU

LOGON APPL7
```

Auf Stufe 4 wird `APPL6, STARTUP MENU` als Rückkehrpunkt definiert.

9. Der Benutzer führt ein Programm aus, das zwei Kommandos ausführt:

```
SETUP *,MENU

LOGON APPL8
```

Auf Stufe 5 wird `APPL7, STARTUP MENU` als Rückkehrpunkt definiert.

10. Der Benutzer führt ein Programm aus, das zwei Kommandos ausführt:

```
SETUP *,MENU

LOGON APPL9
```

Auf Stufe 6 wird `APPL8, STARTUP MENU` als Rückkehrpunkt definiert.

11. Der Benutzer begibt sich mit dem Kommando `RETURN 2` zwei Stufen zurück.

Natural kehrt zu `APPL7` (Stufe 5) zurück, weil das die vorletzte Session war (alle Informationen zu `APPL8` sind jetzt verloren). Stufe 6 (`APPL8`) wird gelöscht. Der Rückkkehrpunkt der Stufe 5 (`APPL7`) wird aktiviert und die Stufe gelöscht.

12. Der Benutzer begibt sich mit dem Kommando `RETURN` eine Stufe zurück.

Der Rückkkehrpunkt der Stufe 4 (`APPL6`) wird aktiviert, der Rückkehrpunkt der Stufe wird gelöscht. Natural kehrt zu `APPL6` zurück, weil dies die Session vor `APPL7` war.

13. Der Benutzer begibt sich mit dem Kommando `RETURN` eine Stufe zurück.

Der Rückkkehrpunkt der Stufe 3 (`APPL5`) wird aktiviert, der Rückkehrpunkt der Stufe wird gelöscht. Natural kehrt zu `APPL5` zurück, weil dies die Session vor `APPL6` war.

14. Der Benutzer setzt das Kommando `RETURN I` ab.

Der Rückkkehrpunkt der Stufe 2 (`APPL2`) wird gelöscht, der Rückkkehrpunkt der Stufe 1 (`APPL1`) wird aktiviert.

52 SQLDIAG

Anmerkung: This command is only available with Natural for DB2.

The SQLDIAG command provides diagnostic information about the last SQL statement (other than a GET DIAGNOSTICS statement) that was executed. This diagnostic information is gathered as the previous SQL statement is executed. Some of the information available through the GET DIAGNOSTICS statement is also available in the SQLCA.

For detailed information about the returned diagonstics information see the IBM DB2 documentation of the GET DIAGOSTICS statement.

Fields, which are prefixed with a a plus sign (+), may contain more data than displayed on the screen. You can display the full contents either when you position the cursor on the field (description or data) and press Enter, or when you enter the abreviation of the field (which are the capital letters of the description) prefoxed by the '+' sign in the command line. For example,. +SN shows a window with the full value of the SERVER_NAME.

The SQLDIAG command can be issued either from the Natural NEXT prompt or from within a Natural program (by using the FETCH statement).

Sample **SQLDIAG Diagnostic Information** Screens:

```
11:03:12           *** SQLDIAG Diagnostic Information ***           2006-04-15
                          - Statement Information -



DB2_Last_Row .....................           0
DB2_Number_Parameter_Markers .....           0
DB2_Number_Result_Sets ...........           0
DB2_Return_Status ................           0
DB2_SQL_Attr_Cursor_Hold .........     _Rowset ..     _Scrollable ...
```

```
                                          _Type ..     _Sensitivity ..
 DB2_Number_Rows ..................          0
 Row_Count ........................          0

 More .............................

 Number ...........................          1



Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Error Exit  Updat                                     Next  Canc
```

```
 11:09:49           *** SQLDIAG Diagnostic Information ***           2006-04-15
                          - Condition Information 1 -

+Server_Name ................. DAEFDB28
+CUrsor_Name .................
 DB2_Error_Code1 .............        -500   DB2_Error_Code2 ...           0
          _Code3 .............           0             _Code4 ...          -1
 DB2_Internal_Error_Pointer ..        -500  +DB2_Sqlerrd1(-6) ..        -500
 DB2_Module_Detecting_Error .. DSNXOTL
+DB2_Ordinal_Token_1 ......... HGK.DEMO
 DB2_Row_Number ..............           0
 DB2_Line_Number .............           0
 DB2_Returned_SQLCode ........        -204
 DB2_Reason_Code .............           0
 Returned_SQLState ...........       42704
 DB2_Message_ID ............... DSN00204E
 Message_Octet_Length ........           0
+Message_Text ................. HGK.DEMO IS AN UNDEFINED NAME


Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Error Exit  Updat                               Prev  Next  Canc
```

```
 11:14:41           *** SQLDIAG Diagnostic Information ***           2006-04-15
                           - Connection Information -



 DB2_Authentication_Type ..
 DB2_Authentication_ID .... GGS

 DB2_Connection_State .....           0
 DB2_Connection_Status ....           0
 DB2_Encryption_Type ......
```

```
 DB2_Product_ID ........... DSN08010
 DB2_Server_Class_Name .... QDB2 for DB2 UDB for z/OS









Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Error Exit  Updat                               Prev        Canc
```

53 SQLERR

SQLERR

Anmerkungen:

1. This command is only available with Natural for DB2, Natural SQL Gateway, and Natural for SQL/DS.
2. There are minor differences depending on whether the command is used with Natural for DB2, Natural SQL Gateway, or Natural for SQL/DS. These differences are marked accordingly in the following description.

The SQLERR command is used to obtain diagnostic information about the most recent SQL error.

When an SQL error occurs, Natural issues an appropriate error message. When you enter the SQLERR command, the following information on the most recent SQL error is displayed:

- the Natural error message number;
- the corresponding reason code (if applicable);
- for Natural for DB2: the variables SQLSTATE and SQLCODE returned by DB2
- for Natural SQL Gateway: the SQL code returned by the ConnecX SQL engine or the SQL database system;
- for Natural for SQL/DS: the variables SQLSTATE and SQLCODE returned by SQL/DS;
- the corresponding error message.

The SQLERR command can be issued either from the Natural NEXT prompt or from within a Natural program (by using the FETCH statement).

Sample SQLERR Diagnostic Information Screen (Natural for DB2)

```
               ***** SQLERR Diagnostic Information *****
---------------------- NATURAL SQL Interface Codes -------------------------
Return Code: 3700    Reason Code: 0      SQLSTATE : 52003   SQL code : -206
--------------------------------- SQLCA-------------------------------------
SQLERRP (DB2 Sub routine where error occurred)              :  DSNXOGP
SQLERRD (DB2 Internal State)
        RDS Return Code                                    :         700
        DBSS Return Code                                   :           0
        Number of Rows Processed                           :           0
        Estimated Cost                                     :        11.2
        Syntax error on PREPARE or EXECUTE IMMEDIATE       :           0
        Buffer Manager ERROR Code                          :           0
SQLWARN (Warning Flags)
        Data truncated
        Null Values ignored (AVG,SUM,MAX,MIN)              :
        No. of columns greater than no. of host variables  :
        UPDATE/DELETE without WHERE clause                 :
        SQL Statement not valid in DB2                     :
         Adjustment to DATE/TIMESTAMP Variable made         :
DB2 Error Message :
DSNT4081 SQLCODE = -206, ERROR: THE OBJECT TABLE OR VIEW OF THE INSERT,
              DELETE, OR UPDATE STATEMENT IS ALSO IDENTIFIED IN A FROM CLAUSE
```

Sample SQLERR Diagnostic Information Screen (Natural SQL Gateway)

```
                 *** SQLERR Diagnostic Information ***
---------------------- Natural SQL Interface Codes -------------------------
Return Code: 3700    Reason Code: 0                 ZZN01   SQL code: -4017
--------------------------------- SQLCA -------------------------------------
SQLERRD (Additional Information)
        Number of Rows Processed                           :           0

SQLWARN (Warning Flags)
        Data truncated                                     :

        No. of columns greater than No. of host variables  :
CNX Error Message :
4017(E): SERVER ERROR: ODBC:(HY000) NATIVE:(0) : Ambiguous table reference: (D
EMO) ? PERS_ID , NAME , ADDRESS , DATEOFBIRTH , SALARY FROM << Syntax Error >>
  NSB.DEMO ?.
```

```
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
            Error Exit  Expl        Parms  -     +          Prev  Next  Canc
```

Sample SQLERR Diagnostic Information Screen (Natural for SQL/DS)

```
                  *** SQLERR Diagnostic Information ***
---------------------- NATURAL SQL Interface Codes -------------------------
Return Code: 3700              Reason Code: 0               SQL code : -204
-------------------------------- SQLCA ---------------------------------------
SQLERRP (Adabas SQL Subroutine where error occurred)       :  ARIXOCA
SQLERRD (Adabas SQL Internal State)
        RDS Return Code                                    :            100
        DBSS Return Code                                   :              0
        Number of Rows Processed                           :              0
        Estimated Cost                                     :              1.0
        Syntax error on PREPARE or EXECUTE IMMEDIATE       :              0
        Buffer Manager ERROR Code                          :              0
SQLWARN (Warning Flags)
        Data truncated
        Null Values ignored(AVG,SUM,MAX,MIN)               :
        No. of columns greater than no. of host variables  :
        UPDATE/DELETE without WHERE clause                 :
        SQL statement causes a performance degradation     :
        Adjustment to DATE/TIMESTAMP Variable made         :
SQL/DS Error Message :
SAG.SYSTABLES  not found in system catalog
```

54 STOW

```
STOW [object-name [library-id]]
```

Verwandte Kommandos: SAVE | CATALOG

Das Kommando STOW dient dazu, ein Objekt gleichzeitig in Sourceform und Objektform in der Natural-Systemdatei zu kompilieren und zu speichern. Es hat die gleiche Wirkung wie ein CATALOG-Kommando mit anschließend abgesetztem SAVE-Kommando.

See also:

Natural-Compiler in Natural *System-Architektur*
Namenskonventionen für Objekte in *Natural benutzen*

STOW	Wenn Sie das Kommando ohne *object-name* benutzen, wird das im Arbeitsbereich des Editors vorhandene Sourceobjekt und der erzeugte Code unter demselben Namen in der aktuellen Library gespeichert. Vorhandener Source- und Objektcode werden ersetzt.
STOW *object-name*	Verwenden Sie diese Kommandosyntax, um ein neues Objekt (Source und generierter Code) namens *object-name* in der aktuellen Library zu speichern. Falls das Objekt schon vorhanden ist, wird das Kommando zurückgewiesen.
STOW *object-name library-id*	Wenn Sie sowohl *object-name* als auch *library-id* angeben, wird ein neues Objekt angelegt und unter diesem Namen in der angegebenen Library gespeichert. Falls das Objekt in Sourceform oder katalogisierter Form vorhanden ist, wird das Kommando zurückgewiesen.

Anmerkung: Falls im Parametermodul eine ungültige Systemdatei FDIC angegeben ist, erscheint eine entsprechende Natural-Fehlermeldung, wenn das STOW-Kommando abgesetzt wird.

55 STRUCT

```
STRUCT
```

Das Kommando `STRUCT` erfüllt zwei Aufgaben:

- Sie können es dazu verwenden, die Sourcecode-Zeilen eines Programms entsprechend der Programmstruktur einzurücken.
- Diverse zusätzliche Anzeigen verdeutlichen die Programmstruktur und erlauben es Ihnen so, etwaige Inkonsistenzen in der Struktur aufzuspüren.

Da das `STRUCT`-Kommando die Natural-Sourcen unabhängig davon verarbeitet, ob sie tatsächlich katalogisert werden können oder nicht, wird die Source nicht auf syntaktische Korrektheit analysiert. Obwohl das `STRUCT`-Kommando sorgfältig strukturierte Source-Zeilen liefert, können Zeilen vorhanden sein, die unklar sind und die nicht wie erwartet strukturiert werden.

Folgende Arten von Statements sind vom `STRUCT`-Kommando betroffen:

- Verarbeitungsschleifen (`READ`, `FIND`, `FOR` usw.),
- Statement-Blöcke mit Bedingungen (`AT BREAK`, `IF`, `DECIDE FOR` usw.),
- `DO/DOEND`-Statement-Blöcke,
- `DEFINE DATA`-Statement-Blöcke,
- interne Subroutinen.

Wenn Sie das `STRUCT`-Kommando eingeben, erhalten Sie das `STRUCT`-Menü, von dem Sie folgende Funktionen auswählen können:

Generate Structured Source into Work Area

Mit dieser Funktion können Sie ein Source-Programm einrücken, so dass die Einrückung der Sourcecode-Zeilen die Struktur des Programms widerspiegelt.

Diese Funktion entspricht der des Editorkommandos `STRUCT`.

Bei der Einrückung wird die Länge der Sourcecode-Zeilen berücksichtigt; d.h. eine eingerückte Zeile wird nicht über den rechten Rand hinausgeschoben; selbst wenn eine „korrekte“ Einrückung dies erfordern würde, wird eine Zeile nicht über den Rand hinausgeschoben, sondern nur soweit nach rechts wie möglich.

Die **Generate**-Funktion bietet Ihnen folgende Optionen:

Feld	Erklärung	
Source Name	In dieses Feld geben Sie den Namen der Source ein, die Sie einrücken möchten. Die betreffende Source wird dann von der Systemdatei in den Editor-Arbeitsbereich gelesen und strukturell eingerückt. Wenn sie keinen Namen angeben, wird das gerade im Editor-Arbeitsbereich befindliche Objekt eingerückt. Wenn der Arbeitsbereich leer ist, müssen Sie einen Namen angeben.	
Shift setting	In diesem Feld können Sie angeben, um wieviel Stellen (1 bis 9) die Zeilen jeweils eingerückt werden sollen. Standardmäßig wird um 2 Stellen eingerückt.	
Align Comments	Y	Jede Kommentarzeile wird soweit eingerückt wie die darüberstehende Statement–Zeile; Ausnahme: Kommentarzeilen, die am Anfang einer Zeile anfangen, werden nicht eingerückt.
	N	Kommentarzeilen werden nicht eingerückt.
	L	Kommentarzeilen werden linksbündig ausgerichtet.
Display Messages	Y	Sie erhalten eine Meldung, dass das strukturierte Programm in den Arbeitsbereich generiert wurde, sowie eine Liste der Sourcecode-Zeilen, die nicht „korrekt“ eingerückt (siehe oben) werden konnten.
	N	Es wird keine derartige Meldung ausgegeben.
Return to STRUCT	Y	Nach Ausführen der **Generate**-Funktion gelangen Sie zum `STRUCT`-Menü zurück.
	N	Nach Ausführen der **Generate**-Funktion gelangen Sie zu dem Schirm zurück, auf dem Sie das `STRUCT`-Kommando eingegeben hatten.

Anmerkung: Ein im Reporting Mode geschriebenes Programm wird anders eingerückt als ein im Structured Mode geschriebenes.

Teilweise Einrückung

Mit den Spezial-Statements `/*STRUCT OFF` und `/*STRUCT ON` können Sie bestimmte Abschnitte Ihres Source-Programms von der Einrückung ausschließen. Die beiden Statements müssen jeweils am Anfang einer Sourcecode-Zeile stehen. Wenn Sie die **Generate**-Funktion ausführen, werden die Zeilen zwischen diesen beiden Statements nicht davon betroffen; sie bleiben, wie sie waren.

Beispiel für strukturelle Einrückung

Programm, bevor es eingerückt wird:

```
DEFINE DATA LOCAL
1 EMPL VIEW OF EMPLOYEES
2 PERSONNEL-ID
2 FULL-NAME
3 FIRST-NAME
3 NAME
1 VEHI VIEW OF VEHICLES
2 PERSONNEL-ID
2 MAKE
END-DEFINE
FIND EMPL WITH NAME = 'ADKINSON'
IF NO RECORDS FOUND
WRITE 'NO RECORD FOUND'
END-NOREC
FIND (1) VEHI WITH PERSONNEL-ID = EMPL.PERSONNEL-ID
DISPLAY EMPL.PERSONNEL-ID FULL-NAME MAKE
END-FIND
END-FIND
END
```

Dasselbe Programm, nachdem die Funktion **Generate Structured Source** angewandt wurde:

```
DEFINE DATA LOCAL
1 EMPL VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 FULL-NAME
    3 FIRST-NAME
    3 NAME
1 VEHI VIEW OF VEHICLES
  2 PERSONNEL-ID
  2 MAKE
END-DEFINE
FIND EMPL WITH NAME = 'ADKINSON'
  IF NO RECORDS FOUND
    WRITE 'NO RECORD FOUND'
  END-NOREC
  FIND (1) VEHI WITH PERSONNEL-ID = EMPL.PERSONNEL-ID
    DISPLAY EMPL.PERSONNEL-ID FULL-NAME MAKE
  END-FIND
END-FIND
END
```

Display Structure of Source

Mit dieser Funktion können Sie den Sourcecode eines Objekts zusammen mit verschiedenen Informationen, die die Struktur des Objekts verdeutlichen, anzeigen.

Die **Display**-Funktion bietet Ihnen die folgenden Optionen:

Feld	Erklärung	
Source Name	In dieses Feld geben Sie den Namen der Source ein, die Sie anzeigen möchten. Die betreffende Source wird dann von der Systemdatei in den Editor-Arbeitsbereich gelesen und angezeigt. Wenn sie keinen Namen angeben, wird das gerade im Editor-Arbeitsbereich befindliche Objekt angezeigt. Wenn der Arbeitsbereich leer ist, müssen Sie einen Namen angeben.	
Display Compressed	Y	Sourcecode-Zeilen auf derselben strukturellen Ebene werden nicht angezeigt. Es werden nur die Zeilen gezeigt, die eine Änderung in der Strukturtabelle am rechten Bildschirmrand bewirken. Anhand der Lücke in der Abfolge der Zeilennummern können Sie sehen, wieviele Zeilen zwischen zwei gezeigten Zeilen nicht gezeigt werden.
	N	Alle Sourcecode-Zeilen werden angezeigt.
Return to STRUCT	Y	Nach Ausführen der Display-Funktion gelangen Sie zum `STRUCT`-Menü zurück.
	N	Nach Ausführen der Display–Funktion gelangen Sie zu dem Schirm zurück, auf dem Sie das `STRUCT`-Kommando eingegeben hatten.

Folgende Informationen werden angezeigt:

Line Numbers	Bei jedem Statement, das einen Statement-Block beendet, wird links neben dem Sourcecode die Sourcecode-Zeilennummer des Statements, das den betreffenden Block initiiert, angezeigt.
Structure Table	Rechts neben dem Sourcecode wird eine Strukturtabelle angezeigt, die Aufschluß über offene Statement-Blöcke gibt. Für jeden offenen Block wird ein einzelner Buchstabe angezeigt. Die verschiedenen Buchstaben beziehen sich auf verschiedene Arten von Statements (für eine Erklärung der einzelnen Buchstaben drücken Sie PF1). Etwaige Inkonsistenzen in der Struktur des Sourcecodes werden durch eine entsprechende Meldung in der Strukturtabelle angezeigt.

Beispiel für Anzeige mit Struktur-Informationen:

```
14:17:47 - Structured Source ABC in Library XYZ -                    2003-02-04
  0010      DEFINE DATA LOCAL                                      *0
  0020      1 EMPL VIEW OF EMPLOYEES                               *0
  0030        2 PERSONNEL-ID                                       *0
  0040        2 FULL-NAME                                          *0
  0050          3 FIRST-NAME                                       *0
  0060          3 NAME                                             *0
  0070      1 VEHI VIEW OF VEHICLES                                *0
  0080        2 PERSONNEL-ID                                       *0
  0090        2 MAKE                                               *0
  0100 0010 END-DEFINE                                             *0
  0110      FIND EMPL WITH NAME = 'ADKINSON'                       *F
  0120        IF NO RECORDS FOUND                                  *FJ
  0130          WRITE 'NO RECORD FOUND'                            *FJ
  0140 0120   END-NOREC                                            *FJ
  0150        FIND (1) VEHI WITH PERSONNEL-ID = EMPL.PERSONNEL-I   *FF
  0160          DISPLAY EMPL.PERSONNEL-ID FULL-NAME MAKE           *FF
  0170 0150   END-FIND                                             *FF
  0180 0110 END-FIND                                               *F
  0190      END                                                    *
  PF1=Help, PF2=Menu, PF3=Exit, PF6=Top, PF12=Cancel.
```

Der aktuelle Inhalt des Arbeitsbereichs wird von der angezeigten Source nicht beeinflußt.

Print Structure of Source

Mit dieser Funktion können Sie den Sourcecode eines Objekts zusammen mit seinen Struktur-Informationen ausdrucken.

Die **Print**-Funktion entspricht der Funktion **Display Structure of Source**, nur dass die Ausgabe nicht auf dem Bildschirm angezeigt, sondern auf einem Drucker ausgedruckt wird.

Mit der **Print**-Funktion haben Sie die gleichen Optionen wie bei der **Display**-Funktion.

Write Structure of Source into Work Area

Mit dieser Funktion können Sie eine Source von der Systemdatei lesen und zusammen mit ihren Struktur-Informationen sowie mehreren Zeilen (mit Zeilennummer `0000`) am Anfang der Source, die die Struktur-Informationen erklären, in den Editor-Arbeitsbereich laden.

Mit der **Write**-Funktion haben Sie die gleichen Optionen wie bei der **Display**-Funktion, mit dem Unterschied, dass Sie einen Source-Namen angeben *müssen*.

Die Source und ihre Struktur-Informationen werden als Text in den Arbeitsbereich geladen und können mit dem Systemkommando `EDIT` editiert werden.

56 SYSADA

SYSADA

Mit dem Kommando SYSADA rufen Sie die ADACALL-Utility auf, die sich in der Library SYSADA befindet.

Mitteils der ADACALL-Utility können Sie von einem Großrechner-Natural aus Adabas-Direktkommandos (Native Commands) direkt an eine Adabas-Datenbank absetzen.

Die ADACALL-Utility kann für Übungszwecke oder zum Testen/Analysieren verschiedenster Probleme oder Szenarien verwendet werden.

Weitere Informationen siehe *ADACALL - Issuing Adabas Direct Calls* in der *Utilities*-Dokumentation.

57 SYSAPI

SYSAPI

Mit diesem Kommando rufen Sie die SYSAPI-Utility auf.

Diese Utility dient zum Auffinden von Programmierschnittstellen (APIs), die von Natural-Add-on-Produkten, z.B. Entire Output Management (NOM), zur Verfügung gestellt werden.

Zu jedem API liefert die Utility SYSAPI ein oder mehrere Beispielprogramme, die eine Funktionsbeschreibung des API enthalten und zum Testen der Auswirkung des API verwendet werden können.

Weitere Informationen siehe *SYSAPI - APIs of Natural Add-on Products* in der *Utilities*-Dokumentation.

58 SYSBPM

SYSBPM

Mit diesem Kommando rufen Sie die SYSBPM-Utility auf.

Die SYSBPM-Utility liefert statistische Informationen über den aktuellen Status des Natural Buffer Pool, über den Buffer Pool Cache und über die Objekte, die sich gerade im Buffer Pool und im Buffer Pool Cache befinden.

Außerdem bietet SYSBPM Administrationsfunktionen.

Weitere Informationen siehe *SYSBPM Utility - Buffer Pool Management* in der *Utilities*-Dokumentation.

59 SYSCP

SYSCP

Mit diesem Kommando rufen Sie die SYSCP-Utility auf.

Die SYSCP-Utility kann benutzt werden, um Informationen über Codepages zu erhalten und um die Codepage-Zuordnung für eine Natural-Source zu prüfen oder zu ändern.

Weitere Informationen siehe *SYSCP Utility - Code Page Administration* in der *Utilities*-Dokumentation.

60 SYSDB2

SYSDB2

Mit diesem Kommando rufen Sie die Natural Tools for DB2 auf, wenn Natural for DB2 installiert ist.

Weitere Informationen siehe *Using Natural Tools for DB2* im *Natural for DB2*-Teil der *Database Management System Interfaces*-Dokumentation.

61 SYSDDM

SYSDDM

Mit diesem Kommando rufen Sie die SYSDDM-Utility auf. Diese bietet Funktionen, die zum Erstellen und zum Pflegen von Natural Data Definition Modules (DDMs) benötigt werden.

Weitere Informationen siehe *SYSDDM Utility* in der *Editors*-Dokumentation.

Hinweis bezüglich Natural Single Point of Development:

In einer Remote-Entwicklungsumgebung steht dieses Kommando nicht in der Kommandozeile von Natural Studio zur Verfügung, weil DDMs in der Tree View unter dem Node DDM aufgelistet werden und alle Funktionen der SYSDDM-Utility über das Kontextmenü oder die Menüleiste verfügbar sind.

62 SYSEDT

SYSEDT

Mit diesem Kommando rufen Sie die SYSEDT-Utility for Editor Buffer Pool Services auf. Die SYSEDT-Utility ist nur für Natural-Administratoren gedacht. Sie dient zum

- Anzeigen von Parametern und Laufzeitinformationen über den Editor Buffer Pool,
- Ändern von Parametern,
- Löschen von logischen Arbeitsdateien (Work Files) und Wiederherstellungsdateien (Recovery Files).

Weitere Informationen siehe *SYSEDT Utility - Editor Buffer Pool Services* in der *Utilities*-Dokumentation.

63 SYSERR

SYSERR

Mit diesem Kommando rufen Sie die SYSERR-Utility auf.

Mit der SYSERR-Utility können Sie Ihre eigenen anwendungsspezifischen Meldungen schreiben.

- Sie können mit der SYSERR-Utility Fehler- oder Informationsmeldungen von Ihrem Natural-Code trennen und sie getrennt verwalten.
- Sie können Meldungen vereinheitlichen und Meldungsbereiche für verschiedene Meldungsarten festlegen sowie Meldungen in andere Sprachen übersetzen und einen Langtext zu einer Meldung hinzufügen.
- Mit der SYSERR-Utility können Sie darüber hinaus die Texte vorhandener Natural-Systemmeldungen ändern, obwohl dies nicht empfohlen wird, weil diese Änderungen bei neuen Natural-Releases verloren gehen.

Weitere Informationen siehe *SYSERR Utility* in der *Utilities*-Dokumentation.

64 SYSEXT

SYSEXT

Mit diesem Kommando rufen Sie die SYSEXT-Anwendung auf.

Sie enthält verschiedene Natural-Programmierschnittstellen (API). Zu jedem API steht folgendes zur Verfügung:

- ein Subprogramm (in Objektform),
- ein Beispielprogramm (in Sourceform) für den Aufruf des Subprogramms,
- ein Text-Member, das die Funktion des API beschreibt.

Weitere Informationen siehe *SYSEXT - Natural Application Programming Interfaces* in der *Utilities*-Dokumentation.

65 SYSEXV

SYSEXV

Mit diesem Kommando rufen Sie die SYSEXV-Utility mit Beispielen der neuen Merkmale der aktuellen Natural-Versionen auf.

Weitere Informationen siehe *SYSEXV Utility* in der *Utilities*-Dokumentation.

66 SYSFILE

```
SYSFILE [ { WORKFILE } ]
        [ { PRINTER  } ]
```

Mit diesem Kommando können Sie die `SYSFILE`-Funktion der `SYSTP`-Utility aufrufen. Diese Funktion liefert Informationen zu den verfügbaren Arbeitsdateien und Druckdateien (Work und Print Files).

Programmierschnittstelle (API): `USR1007N`. Siehe *SYSEXT - Natural Application Programming Interfaces* in der *Utilities*-Dokumentation.

SYSFILE	Wenn Sie nur `SYSFILE` eingeben, werden Arbeitsdatei- und Druckdatei-Zuweisungen nacheinander angezeigt.
SYSFILE WORKFILE	Nur die Arbeitsdatei-Zuweisungen werden angezeigt.
SYSFILE PRINTER	Nur die Druckdatei-Zuweisungen werden angezeigt.

Weitere Informationen siehe *Natural Print/Work Files - SYSFILE* im Abschnitt *General SYSTP Functions* in der *Utilities*-Dokumentation sowie die plattformspezifischen Informationen zu Arbeitsdateien und Druckdateien in der *Operations*-Dokumentation.

67 SYSMAIN

SYSMAIN

Mit diesem Kommando rufen Sie die SYSMAIN-Utility auf.

Mit dieser Utility können Sie Natural-Objekte kopieren, übertragen und löschen. Die SYSMAIN-Utility dient außerdem dazu, innerhalb des Natural-Systems Objekte mittels der Import-Funktion von einer Umgebung in eine andere Umgebung zu übertragen.

Weitere Informationen siehe *SYSMAIN Utility - Object Maintenance* in der *Utilities*-Dokumentation.

68 SYSNCP

SYSNCP

Mit diesem Kommando rufen Sie die SYSNCP-Utility auf.

Weitere Informationen siehe *SYSNCP Utility* in der *Utilities*-Dokumentation.

69 SYSOBJH

SYSOBJH

Mit diesem Kommando rufen Sie den Object Handler auf. Mit dem Object Handler können Sie Natural- und Nicht-Natural-Objekte zwecks Verteilung in Natural-Umgebungen handhaben.

Weitere Informationen siehe *Object Handler* in der *Utilities*-Dokumentation.

70 SYSPARM

SYSPARM

Mit diesem Kommando rufen Sie die SYSPARM-Utility auf. Sie können diese Utility benutzen, um Zeichenketten mit Natural-Profilparametern zu erstellen und zu pflegen, die als Natural-Profile gespeichert werden.

Diese Natural-Profile können dann zu Beginn einer Natural-Session mit dem Profilparameter PROFILE aufgerufen werden.

Das SYSPARM-Kommando bietet verschiedene Parameter, die hauptsächlich im Batch-Betrieb verwendet werden (siehe *Direct Commands and Batch Processing* in der Beschreibung der SYSPARM-Utility).

Weitere Informationen siehe *SYSPARM Utility* in der *Utilities*-Dokumentation.

71 SYSPROD

SYSPROD

Mit dem Kommando SYSPROD können Sie feststellen, welche Produkte in Ihrer Natural-Umgebung installiert sind: d.h. Natural selbst, sowie Produkte, die mit bzw. unter Natural laufen.

Programmierschnittstellen (APIs): USR0050N, USR2031N. Siehe *SYSEXT - Natural Application Programming Interfaces* in der *Utilities*-Dokumentation.

Wenn Sie das Kommando eingeben, erscheint ein Fenster, in dem zu jedem installierten Produkt die folgenden Informationen angezeigt werden:

- Produktname
- Produktversion
- System Maintenance (SM) Level
- Installationszeitpunkt (Datum und Uhrzeit)

Zu einigen der aufgeführten Produkte können Sie weitere Informationen erhalten, indem Sie sie mit einem Kommando markieren. Durch Eingabe eines Fragezeichens (?) in der **Cmd**-Spalte des SYSPROD-Fensters erhalten Sie eine Liste der verfügbaren Kommandos.

Zeilenkommando	Beschreibung
EX	Weitere Produktinformationen anzeigen.
HI	Historie der Produktinformationen anzeigen.
SC	Unterbestandteile des Produkts anzeigen.

Anmerkung: Bei manchen Produkten sind keine Zeilenkommandos zulässig.

SYSPROD-Kommadoschnittstelle (Batch-Betrieb)

Für den Batch-Betrieb oder eine unformatierte Online-Ausgabe können Sie `SYSPROD` mit zusätzlichen Kommandozeilenparametern aufrufen:

72 SYSPROF

SYSPROF

Mit dem Kommando SYSPROF können Sie sich die gegenwärtigen Definitionen der Natural-Systemdateien anzeigen lassen.

Zu jeder Systemdatei werden folgende Informationen angezeigt.

- Dateiname
- Datenbank-ID
- Dateinummer
- Datenbanktyp

Programmierschnittstellen (APIs): USR0010N, USR2013N, USR3013N. Siehe auch *SYSEXT - Natural Application Programming Interfaces* in der *Utilities*-Dokumentation.

73 SYSRPC

SYSRPC

Mit diesem Kommando rufen Sie die SYSRPC-Utility auf.

Die SYSRPC-Utility bietet Funktionen zum Verwalten von Natural Remote Procedure Calls.

Weitere Informationen siehe *SYSRPC Utility* in der *Utilities*-Dokumentation.

Informationen, wie Sie die Funktionen der SYSRPC-Utility benutzen können, um ein Kommunikationsnetzwerk zwischen Server- und Client-Systemen zu erstellen, finden Sie in der *Natural Remote Procedure Call (RPC)*-Dokumentation.

74 SYSTP

SYSTP

Mit diesem Kommando rufen Sie die SYSTP-Utility auf, mit der Sie verschiedene TP-Monitor-spezifische Charakteristika von Natural überwachen und steuern können.

Weitere Informationen siehe *SYSTP Utility* in der *Utilities*-Dokumentation.

75 TECH

TECH

Mit diesem Kommando können Sie sich technische und andere Informationen über Ihre Natural-Session anzeigen lassen.

Folgende Informationen werden angezeigt:

- User-ID
- Library-ID
- Natural-Version und -SM-Level (siehe auch *Version* im *Glossary*)
- Startup-Transaktion
- Natural Security-Indikator
- Betriebssystemname und -version
- Maschinenklasse
- Hardware
- TP-Monitor (auf Großrechnern, bei Windows (*TPSYS) nur in einer Remote-Entwicklungsumgebung)
- Gerätetyp
- Terminal-ID (auf Großrechnern, bei Windows nur in einer Remote-Entwicklungsumgebung)
- Codepage
- Locale
- zuletzt ausgeführtes Kommando
- Informationen zum zuletzt aufgetretenen Fehler
- Namen, Datenbank-IDs und Dateinummern aller gerade aktiven Steplib-Libraries

- Namen, Typen und Ebenen aller gerade aktiven Programmierobjekte und aller Objekte auf höheren Ebenen sowie die Zeilennummern der Statements, die untergeordnete Programmierobjekte (nur bei Großrechnern, UNIX und OpenVMS) aufrufen.

Anmerkungen:

1. Nur bei Anwendungen mit zeichenorientierter Benutzeroberfläche: Um diese Informationen von einem beliebigen Punkt in einer Anwendung aus aufzurufen, können Sie das Terminalkommando `%<TECH` verwenden.
2. Dieses Kommando steht auch in einer Remote-Session zur Verfügung. Alle Informationen können auch im Batch-Betrieb gelesen werden.

Programmierschnittstelle (API): `USR2026N`. Siehe auch *SYSEXT - Natural Application Programming Interfaces* in der *Utilities*-Dokumentation.

76 TEST

Mit dem Kommando TEST rufen Sie die Debugging-Funktion auf.

TEST	Wenn Sie nur TEST eingeben, erhalten Sie das Hauptmenü der Debugging–Funktion. Mit TEST ON/OFF aktivieren/deaktivieren Sie den Test-Modus der Debugging-Funktion.
TEST ON	Aktiviert den Test-Modus der Debugging-Funktion.
TEST OFF	Deaktiviert den Test-Modus der Debugging-Funktion.
debugger-commands	Die Direktkommandos zum Ausführen von Debuggin-Funktionen sind im Abschnitt *Command Summary and Syntax* in der *Debugger*-Dokumentation beschrieben.

Um die Debugging-Funktion von einem beliebigen Punkt in einer Anwendung aus aufzurufen, können Sie das Terminalkommando %<TEST verwenden.

Weitere Informationen finden Sie in der *Debugger*-Dokumentation.

Die *Utilities*-Dokumentation enthält außerdem Informationen über weitere Natural-Utilities, die Sie fzum Testen und Überwachen im Online-Modus verwenden können.

Anmerkung zu Natural Single Point of Development:

Dieses Kommando ist nur auf Großrechnern verfügbar. Falls ein Benutzer ein Programm namens TEST geschrieben hat, dann führt Natural in einer lokalen Windows-, UNIX - oder OpenVMS-Umgebung dieses Programm aus. Besteht eine aktive Verbindung zu einem Development-Server auf einem Großrechner, wird die Großrechner-Utility TEST aufgerufen, wenn dieses Kommando abgesetzt wird.

77 TEST DBLOG

```
TEST DBLOG [parameters]
```

Mit diesem Kommando rufen Sie die DBLOG-Utility auf, mit der Sie Datenbankaufrufe protokollieren können.

TEST DBLOG	Aktiviert bzw. deaktiviert die DBLOG-Utility.
parameters	Die Parameter, die Sie beim TEST DBLOG-Kommando benutzen können, sind im Abschnitt *TEST DBLOG Command* der*Utilities*-Dokumentation beschrieben.

Weitere Informationen siehe *DBLOG Utility - Logging Database Calls* in der *Utilities*-Dokumentation.

Die *Utilities*-Dokumentation enthält Informationen über weitere Natural-Utilities, die zum Testen und Überwachen verwendet werden können.

Anmerkung zu Natural Single Point of Development:

Besteht eine aktive Verbindung zu einem Development-Server auf einem Großrechner, wird die Großrechner-Utility DBLOG aufgerufen, wenn dieses Kommando unter Natural for Windows abgesetzt wird.

78 UNCATALOG

Dieses Kommando wird nur aus Kompatibilitätsgründen unterstützt. Es empfiehlt sich, statt des `UNCATALOG`-Kommandos das `DELETE`-Kommando zu verwenden, da `DELETE` mehr Flexibilität sowie Schutz gegen versehentliches Löschen bietet.

79 UNLOCK

Mit dem UNLOCK-Kommando können Sie Natural-Sourceobjekte lokal in einer Natural-Großrechnerumgebung entsperren.

Das UNLOCK-Kommando dient dazu, Sourceobjekte anzuzeigen, die gesperrt sind, und diese zu entsperren. Es wird empfohlen, dass dieses Kommando nur vom Natural-Administrator benutzt wird. Der Administrator kann jedoch die Benutzung dieses Kommandos für jedes Benutzerprofil in Natural Security ermöglichen.

Anmerkungen:

1. Als Voraussetzung für die Benutzung des UNLOCK-Systemkommandos muss der Profilparameter SLOCK auf PRE gesetzt sein.
2. Wenn die Anzahl der gesperrten Objekte groß ist, kann es passieren, dass die angezeigte Liste nicht sortiert ist. Zur Abhilfe können Sie die Größe des vom Sort-Programm benutzten Work-Buffer erhöhen; siehe Schlüsselwortparameter WRKSIZE des Profilparameters SORT.

Weitere Informationen siehe *Locking of Source Objects* in der *Editors*-Dokumentation und Profilparameter SLOCK in der *Parameter Reference*-Dokumentation.

Siehe auch *Namenskonventionen für Objekte* in der Dokumentation *Natural benutzen*.

Natural-Objekte entsperren

Wenn Sie das Systemkommando UNLOCK ohne Parameter benutzen, erscheint eine Maske, auf der Sie die Parameter eingeben können.

```
UNLOCK
```

Die folgende Übersicht zeigt die Kommandosyntax zum Entsperren von Natural-Objekten:

```
UNLOCK [NATURAL] [OBJECT] object-name
       [TYPE object-type]
       [LIBRARY library-name]
       [DBID dbid] [FNR fnr]
       [PASSWORD password] [CIPHER cipher]
       [USER locked-by]
       [DATE locked-on [locked-on2]]
```

UNLOCK-Parameter-Beschreibungen

Sie müssen in jedem Fall den *object-name* angeben. Falls Sie einen der übrigen Parameter nicht angeben, wird der Standardwert verwendet.

Parameter	Format/Länge	Standardwert	Beschreibung
object-name	A33	*	Der Name des zu entsperrenden Objekts. Sie können Stern-Notation (*) oder das Größerzeichen (>) verwenden.
object-type	A1	*	**Natural-Objekttypen:** Anstelle von *object-type* können Sie einen der unten aufgeführten Objekttypcodes oder Stern-Notation (*) benutzen.

P	Programm
4	Klasse
N	Subprogramm
S	Subroutine
7	Function
8	Adapter
C	Copycode
H	Helproutine
T	Text
M	Map
L	Local Data Area
G	Global Data Area
A	Parameter Data Area
V	DDM (View)

Anmerkung: Das Sperren können Sie auch lokal auf einem Großrechner-Server einschalten, der auf Natural für Großrechner Version 4.2 oder höher basiert. In diesem Fall gelten folgende Einschränkungen: Der *application-name* kann nicht als Auswahlkriterium verwendet werden. Bei *dbid* und *fnr* werden die aktuellen Systemdateien FNAT and FUSER durchsucht, wenn Sie Stern-Notation (*) benutzen.

Parameterverarbeitung und Anzeige der gefundenen Objekte

Das Objekt sofort entsperrt, wenn die angegebenen Parameter gültig sind und ein vollständiger Objektname angegeben wurde und wenn das entsprechende Objekt gefunden und es vom aktuellen Benutzer gesperrt wurde. Es erscheint eine entsprechende Meldung.

Dies gilt unter der Bedingung, dass der Objektname direkt und ohne die Benutzung von Stern-Notation (*) angegeben wird und der aktuelle Benutzer seine eigenen gesperrten Objekte zu entsperren versucht.

Falls einer der angegebenen Parameter ungültig ist oder falls keine gesperrten Objekte gefunden werden, erscheint der Unlock-Dialog mit einer Fehlermeldung.

In den folgenden Fällen werden die gefundenen gesperrten Objekte angezeigt und können mittels dem Zeilenkommando `U` entsperrt werden (siehe unten):

- Wenn Sie Stern-Notation (*) oder (wo zutreffend) das Größerzeichen (>) benutzt haben.
- Wenn Sie keinen spezifischen Objektnamen angegeben haben.

Unlock-Liste

Funktionstasten

Die Unlock-Liste bietet folgende Funktionstasten:

PF1	Help	Hilfe aufrufen.
PF3	Exit	Rückkehr zur Unlock-Liste.
PF6	--	Zum Anfang der Liste springen.
PF7	-	Eine Seite zurück blättern.
PF8	+	Eine Seite vor blättern.
PF9	++	Zum Ende der Liste springen.
PF10	<	Nach links blättern zum ersten Teil der Informationen (Typ, Library, Datenbank-ID, Dateinummer).
PF11	>	Nach rechts blättern zum zweiten Teil der Informationen (gesperrt durch, gesperrt am).
PF12	Cancel	Anulliert das `UNLOCK`-Kommando.

Zeilenkommando

U	In der Spalte **Cmd** der Unlock-Liste können Sie das Kommando `U` in einer einzelnen oder in mehreren Zeilen eingeben, um das entsprechende Objekt oder mehrere Objekte zu entsperren. Die erfolgreiche Entsperrung wird durch die Meldung **unlocked** in der Spalte **Message** angezeigt.

Batch-Verarbeitung

Falls kein Fehler aufgetreten ist, werden alle gesperrten Objekte entsperrt, und es erscheint eine entsprechende Meldung.

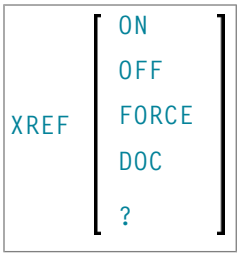
80 UPDATE

Mit dem Kommando UPDATE können Sie verhindern (bzw. ermöglichen), dass ein auszuführendes Programm Datenänderungen auf der Datenbank durchführt.

UPDATE ON	Damit ermöglichen Sie Datenbankänderungen. Dieses Kommando wird ignoriert, falls der Natural-Administrator bereits bei der Installation von Natural die Möglichkeit, Datenbankänderungen vorzunehmen, ausgeschaltet hat.
UPDATE OFF	Damit verhindern Sie, dass die Statements UPDATE, STORE oder DELETE, die normalweise eine Datenänderung bewirken würden, ausgeführt werden. Ein Programm, das diese Statements enthält, wird ganz normal ausgeführt, aber Datenbankänderungen werden nicht durchgeführt. Stattdessen wird für jede nicht ausgeführte Datenbankänderung eine entsprechende Meldung ausgegeben.

Wenn das Systemkommando CHECK in Verbindung mit UPDATE OFF benutzt wird, erscheint eine Fehlermeldung. Das Systemkommando UPDATE hat keinen Einfluss auf andere Natural-Systemkommandos.

81 XREF



Dieses Kommando ist nur verfügbar, wenn Predict installiert ist.

Mit dem Kommando XREF können Sie die Verwendung der Predict-Funktion **Active Cross-References** steuern.

Mit Hilfe der aktiven Referenzen wird im Predict Data Dictionary automatisch dokumentiert, welche Objekte von einem Programm bzw. einer Data Area referenziert werden. Bei diesen Objekten kann es sich handeln um: Programme, Subprogramme, Subroutinen, Helproutinen, Maps, Data Areas, Datenbanksichten (Views), Datenbankfelder, Benutzervariablen, Verarbeitungsregeln (Processing Rules), Fehlernummern, Arbeitsdateien, Drucker, Klassen und gehaltene ISN-Listen.

Die automatische Dokumentation erfolgt, wenn ein Programm bzw. eine Data Area katalogisiert, d.h. in Objektform gespeichert wird.

Mit der XREF-Option des Systemkommandos LIST können Sie sich die dokumentierten Informationen anzeigen lassen. Weitere Informationen zu aktiven Referenzen finden Sie in der Predict-Dokumentation.

Das XREF-Kommando bietet Ihnen folgende Möglichkeiten:

XREF	Wenn Sie das XREF-Kommando ohne Parameter eingeben, erhalten Sie ein Menü, von dem Sie die gewünschte Option auswählen können.
XREF ON	Dieses Kommando schaltet die Dokumentation von aktiven Referenzen ein. Predict dokumentiert die betreffenden Informationen für jedes Natural-Programm bzw. jede Data Area, welche(s) katalogisiert wird.
XREF OFF	Dieses Kommando schaltet die Dokumentation von aktiven Referenzen aus. Es erfolgt keine Dokumentation in Predict. Bereits bestehende Referenzdaten des betreffenden Objekts werden gelöscht.
XREF FORCE	Das Objekt kann nur katalogisiert werden, wenn dafür ein entsprechender Predict-Eintrag vorhanden ist. Beim Katalogisieren werden die betreffenden Referenzdaten in Predict gespeichert. Wenn kein Predict-Eintrag vorhanden ist, kann das Objekt nicht katalogisiert werden.
XREF DOC	Das Objekt kann nur katalogisiert werden, wenn dafür ein entsprechender Predict-Eintrag vorhanden ist. Allerdings werden beim Katalogisieren keine Referenzdaten in Predict gespeichert, und bereits bestehende Referenzdaten des betreffenden Objekts werden gelöscht. Wenn kein Predict-Eintrag vorhanden ist, kann das Objekt nicht katalogisiert werden.
XREF ?	Ruft die Help-Funktion des XREF-Kommandos auf.

XREF unter Natural Security

Unter Natural Security wird die Dokumentation von aktiven Referenzen über die Security-Profile der einzelnen Libraries gesteuert. Je nach Security-Profil kann es sein, dass Sie das XREF-Kommando nur eingeschränkt benutzen können.

Stichwortverzeichnis

S