Natural für Windows

Systemkommandos

Version 6.3.8 für Windows

Februar 2010

Natural

Dieses Dokument gilt für Natural ab Version 6.3.8 für Windows.

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.

Anmerkung: Beschreibungen von Systemkommandos, die nur im Zusammenhang mit SQL-Datenbanken verwendet werden, sind nur in englischer Sprache verfügbar.

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

Die Funktionen, die mit Hilfe von Systemkommandos aufgerufen werden können, sind als Menüeinträge verfügbar. Außerdem können Sie Systemkommandos in der Kommandozeile eingeben. Siehe *Kommandos in der Kommandozeile absetzen* in der Dokumentation *Natural Studio benutzen*.

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).
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.
MAP	Mit diesem Kommando können Sie eine Verbindung zu einem Server herstellen.
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.
SYSPROF	Mit diesem Kommando können Sie sich die gegenwärtigen Definitionen der Natural-Systemdateien anzeigen lassen.
UNMAP	Mit diesem Kommando können Sie die Verbindung zu einem Server beenden.

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 beenden Sie das zur Zeit aktive Objekt und öffnen ein neues Editor-Fenster ohne Inhalt und ohne Namen. Der Editor-Typ ist derselbe wie für das zur Zeit aktive Objekt.
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.
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.
SYSWIZDW	Mit diesem Kommando können Sie den Dialog Wizard aufrufen, mit dem Sie Dialoge für spezielle Zecke erstellen können. Diese Dialoge können zur Anpassung an die gewünschten Anforderungen mehrere Layouts haben.

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

NaturalX-spezifische Kommandos

Kommando	Funktion
REGISTER	Mit diesem Kommando können Sie Natural-Klassen registrieren. Die Registrierung erfolgt für die Server-ID, unter der Natural gestartet wurde.
UNREGISTER	Mit diesem Kommando können Sie die Registrierung von Natural-Klassen rückgängig machen.

Sonstige Kommandos

Kommando	Funktion
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.
SYSEXT	Mit diesem Kommando rufen Sie die SYSEXT-Anwendung auf. Sie enthält verschiedene Natural-Programmierschnittstellen (API).
SYSEXV	Mit diesem Kommando rufen Sie die SYSEXV-Anwendung mit Beispielen der neuen Merkmale der aktuellen Natural-Version auf.

Kommando	Funktion
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).
SYSWIZDB	Mit diesem Kommando können Sie den Dialog Wizard aufrufen, mit dem Sie Dialoge für spezielle Zwecke erstellen können. Diese Dialoge können zur Anpassung an die gewünschten Anforderungen mehrere Layouts haben.
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 CATALL

Mit diesem Systemkommando können Sie alle Objekte oder ausgewählte Objekte in der aktuellen Library katalogisieren, überprüfen, speichern oder gleichzeitig in Sourceform und Objektform in der Natural-Systemdatei kompilieren und speichern.

CATALL im interaktiven Modus

```
CATALL
```

Wenn Sie dieses Kommando eingeben, erscheint das Dialogfeld **Catalog Objects in Library**. In diesem Dialogfeld geben Sie an, welche Objekttypen bearbeitet werden sollen. Die Objekte werden in derselben Reihenfolge bearbeitet wie sie im Dialogfeld aufgelistet sind. Zusätzlich können Sie wählen, welche Aktion durchgeführt und welche Objekte bearbeitet werden sollen.

Siehe auch *Objekte in einer Library katalogisieren* in der Dokumentation *Natural Studio benutzen*.

In dem Dialogfeld können Sie die folgenden Angaben machen:

Starting from	Geben Sie einen Stern (*) ein, wenn Sie in der aktuellen Library alle Objekte der markierten Typen bearbeiten möchten. Wenn Sie die Anzahl der Objekte auf einen bestimmten Bereich einschränken möchten, können Sie für den Namen die Stern-Notation benutzen.
Apply action only to existing modules	Wenn Sie diese Option markieren, werden nur die Objekte erneut katalogisiert, für die bereits Objektmodule in der aktuellen Library existieren; Objekte, die nur in Source-Form vorliegen, werden nicht katalogisiert.
Apply action to all sources	Wenn Sie diese Option markieren, werden `alle` ausgewählten Objekte katalogisiert.
Action	Sie können eine der folgenden Aktionen auswählen, um sie auf den ausgewählten Objekten auszuführen: ■ **Catalog** ■ **Check** ■ **Save** ■ **Stow** Diese Aktionen entsprechen den gleichlautenden Systemkommandos. **Anmerkung:** Unter Natural Security können bestimmte Aktion nicht erlaubt sein.
Renumber source lines	Standardmäßig werden mit den Aktionen **Save** und **Stow** die Sourcecode-Zeilen von Source-Objekten auch neu numeriert. Wenn Sie keine automatische Neunumierung der Zeilen wünschen, deaktivieren Sie diese Kontrollkästchen.

Object types	Standardmäßig gilt CATALL für Objekte jeglichen Objekttyps in der aktuellen Library (alle Objekte sind aktiviert). Wenn Sie bestimmte Objekttypen vom CATALL ausnehmen möchten, deaktivieren Sie die entsprechende Option. Zusätzlich sind Befehlsschaltflächen vorhanden, mit denen Sie alle Option wählen oder alle Kontrollkästchen deaktivieren können. **Anmerkung:** Wenn Sie in einer Mainframe Remote-Entwicklungsumgebung unter SPoD arbeiten, können Sie die Optionen **DDMs** und **Generate new map source** nicht benutzen. Diese Optionen sind dann grau dargestellt.
Generate new map source	Maps, die mit früheren Natural-Versionen erstellt wurden, sind nicht unbedingt kompatibel mit Natural Version 3.1 und höher. Markieren Sie diese Option, damit die Maps während des Katalogisierens konvertiert werden. Diese Option konvertiert die Maps und speichert sie in Source- *und* Objektform.

CATALL im Batch-Modus

```
CATALL object-name [ RECAT ] [TYPES types] [ CATALOG ] [options ...]
                   [ ALL   ]               [ CHECK   ]
                                           [ SAVE    ]
                                           [ STOW    ]
```

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

object-name	Der Name des zu katalogisierenden Objekts. Geben Sie einen Stern (*) ein, wenn Sie in der aktuellen Library alle Objekte der markierten Typen katalogisieren möchten. Wenn Sie die Anzahl der Objekte auf einen bestimmten Bereich einschränken möchten, können Sie für den Namen die Stern-Notation benutzen.
RECAT / ALL	Entspricht den Optionen **Apply action only to existing modules** bzw. **Apply action to all sources** im Dialogfeld **Catalog Objects in Library**. RECAT ist der Standardwert.
TYPES *types*	Entspricht den Objekttypen, die im Dialogfeld **Catalog Objects in Library** markiert werden. Mögliche *types* sind: G Global Data Area A Parameter Data Area L Local Data Area D DDM

	S Subroutine N Subprogramm H Helproutine M Map P Programm 3 Dialog 4 Klasse 7 Funktion 8 Adapter * Alle Typen (gilt standardmäßig) Die *types* müssen als eine Zeichenkette angegeben werden (z.B. LAG for Local, Parameter und Global Data Areas. Standardmäßig gilt CATALL für alle Objektarten in der aktuellen Library.	
CATALOG / CHECK / SAVE / STOW	Entspricht den gleichlautenden Aktionen im Dialog **Catalog Objects in Library**. CATALOG ist der Standardwert.	
options	NOREN	Keine automatische Neunumerierung der Sourcecode-Zeilen von Source-Objekten.

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

6 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:

Objekte katalogisieren in *Natural Studio benutzen*
Namenskonventionen für Objekte in der Dokumentation *Natural Studio 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 Library unter dem Namen des Source-Objekts gespeichert, das zuletzt in den Arbeitsbereich gelesen wurde (z.B. mit einem EDIT-, READ- oder RUN-Kommando).
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 in der Parameterdatei eine ungültige Systemdatei `FDIC` angegeben ist, erscheint eine entsprechende Natural-Fehlermeldung, wenn das `CATALOG`-Kommando abgesetzt wird.

7 CHECK

CHECK

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

Siehe auch:

Objekte prüfen in *Natural Studio benutzen*
Namenskonventionen für Objekte in der Dokumentation *Natural Studio benutzen*

Die Syntaxprüfung wird abgebrochen, sobald ein Syntaxfehler entdeckt wird, und die fehlerhafte Sourcecode-Zeile wird angezeigt. Sie können dann entweder die Zeile korrigieren (woraufhin die Prüfung fortgesetzt wird) oder EINGABE drücken, ohne die angezeigte Zeile zu korrigieren. Die Prüfung wird angehalten, und der Editor wird aufgerufen.

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

8 CLEAR

CLEAR

Mit diesem Kommando beenden Sie das zur Zeit aktive Objekt und öffnen ein neues Editor-Fenster ohne Inhalt und ohne Namen. Der Editor-Typ ist derselbe wie für das zur Zeit aktive Objekt.

Falls das zur Zeit aktive Objekt seit dem letzten Speichern geändert wurde, werden Sie aufgefordert, die Änderungen zu speichern.

Siehe auch *Editor-Fenster löschen* in der Dokumentation *Natural Studio benutzen*.

9 COMPOPT

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

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

Die Standardwerte für die einzelnen Optionen werden mit den entsprechenden Profilparametern in der Natural-Parameterdatei gesetzt.

Syntax-Erklärung

COMPOPT	Wenn Sie nur das Kommando COMPOPT eingeben, wird ein Dialogfeld angezeigt, in dem Sie die unten beschriebenen Optionen ein- bzw. ausschalten können. Die dort vorhandenen Schlüsselwörter werden nachfolgend beschrieben. Siehe auch *Compiler Options* in der Dokumentation *Natural Studio benutzen*.
COMPOPT *option*=*value*	Anstatt eine Option im Dialogfeld zu setzen, können Sie sie auch direkt mit dem COMPOPT-Systemkommando angeben. **Beispiel:** `COMPOPT DBSHORT=ON`

Compiler-Optionen

Folgende Compiler-Optionen stehen zur Verfügung. Informationen über Zweck und mögliche Einstellungen sind in den Beschreibungen der gleichnamigen Natural-Profilparameter enthalten.

KCHECK | PCHECK | DBSHORT | PSIGNF | TQMARK | THSEP | GFID | MASKCME

Compiler-Parameter angeben

Compiler-Parameter können Sie auf verschiedenen Ebenen angeben:

1. **Als Standardeinstellungen**

   Die Standardeinstellungen der einzelnen Compiler-Parameter werden mit dem **Compiler Options**-Dialog in der Configuration Utility vorgenommen. Diese Einstellungen werden in der Natural-Parameterdatei NATPARM gespeichert.

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

   Zu Beginn einer Natural-Session können Sie die Standardeinstellungen überschreiben, indem Sie die entsprechenden Profilparameter setzen.

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

   Während einer aktiven Natural-Session haben Sie zwei Mölichkeiten, die Compiler-Parameterwerte mit dem Systemkommando `COMPOPT` zu ändern: Entweder direkt mittels Kommandozuordnung (`COMPOPT option=value`) oder indem Sie das Kommando `COMPOPT` ohne Optionen absetzen, woraufhin das Dialogfeld **Compiler Options** erscheint. Bei einem `LOGON` auf eine andere Library, werden die Standardeinstellungen (siehe Punkt 1) wieder hergestellt. Beispiel:

```
 OPTIONS KCHECK=ON
DEFINE DATA LOCAL 1
#A (25) 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.

COMPOPT in einer Remote Mainframe-Umgebung

Die folgenden Themen gelten nur, wenn das `COMPOPT`-Kommando in einer Remote Mainframe-Umgebung abgesetzt wird.

Compiler-Schlüsselwortparameter angeben (Remote Mainframe-Umgebung)

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 Kommandozuordnung (`COMPOPT` *`option=value`*) oder indem Sie das Kommando `COMPOPT` ohne Optionen absetzen, woraufhin der Schirm **Compiler 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 (Remote Mainframe-Umgebung)

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 Objekt aufrufenden 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, Functions, Subprogramme, externe Subroutinen, 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 Parameterprüfung. 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 Profilparameter THSEP und 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.

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 (Remote Mainframe-Umgebung)

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

10 DEBUG

```
DEBUG object-name
```

Dieses Kommando ruft den Natural Debugger auf. Mit diesem Kommando geben Sie als *object-name* den Namen des zu verarbeitenden Objekts an.

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

Anmerkung: Dieses Kommando kann nicht in Batch-Verarbeitung ausgeführt werden.

11 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 Studio benutzen*.

Siehe auch *Editoren aufrufen* in der Dokumentation *Natural Studio benutzen*.

Syntax 1

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

object-type

Folgende Objekttypen können editiert werden:

```
( { CLASS  } )
( { 4      } )
  COPYCODE
( { DIALOG } )
( { 3      } )
  GLOBAL
  HELPROUTINE
  LOCAL
  MAP
  PARAMETER
  PROGRAM
( { SUBPROGRAM } )
( { N          } )
  SUBROUTINE
  TEXT
  VIEW
  7 (für Function)
```

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 ein Dialog, wird der Dialog Editor aufgerufen.
- Ist das zu editierende Objekt eine Klasse, wird der Class Builder aufgerufen.
- `EDIT VIEW` funktioniert nur in der aktuellen Library und nur wenn ein *`object-name`* angegeben wird. Falls das anzuzeigende Objekt ein DDM ist, wird der DDM Editor aufgerufen.
- Alle anderen Objektypen (`PROGRAM`, `SUBPROGRAM`, `SUBROUTINE`, `7` (für Function), `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.

Anmerkung: Bei DDMs beträgt die maximale Länge des Objektnamens 32 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.

Anmerkung: Falls der Arbeitsbereich des Editors nicht leer ist und ein Objekt enthält, das in einer Editor-Session geöffnet wurde, wird das entsprechende Editor-Fenster geöffnet und ist eingabefähig. Alle zwischenzeitlich im Arbeitsbereich des Editors gemachten Änderungen (z.B. durch laufende Natural-Programme) werden nicht angezeigt.

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.

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

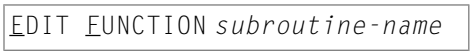
Syntax 2

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.

Anmerkung: Bitte beachten Sie, dass das in dieser Syntax verwendete Schlüsselwort FUNCTION nicht mit dem oben aufgelisteten Natural-**Objekttyp** 7 (für Function) identisch ist. Siehe Beschreibung des Objekttyps Function im *Leitfaden zur Programmierung*.

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
```

12 EXECUTE

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

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

Siehe auch *Objekte mit Execute ausführen* in der Dokumentation *Natural Studio benutzen*.

Syntax-Erklärung

EXECUTE	Das Schlüsselwort `EXECUTE` ist nicht erforderlich; es genügt, *program-name*, d.h. den Namen des auszuführenden Programms oder Dialogs anzugeben.
REPEAT	Wenn das auszuführende Programm oder der auszugebende Dialog 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 oder Dialogs, das bzw. den 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 oder der Dialogs kann nur ausgeführt werden, wenn es bzw. er 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: Wenn einer der übergebenen Parameter Leerzeichen oder eine Zeichenkette enthält, die Leerzeichen enthält, erfolgt die Übergabe nur dann, wenn Sie direkt nach dem Programmnamen ein Eingabe-Delimiterzeichen gesetzt haben.

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
```

```
PROG1,ab cd ef,gh,de fg,ab
```

13 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.

14 GLOBALS

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

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

Anmerkung: Im Batch-Betrieb ist dieses Kommando nur dann ausführbar, wenn Parameter angegeben werden, z.B. kann GLOBALS SM=ON im Batch-Betrieb ausgeführt werden.

Syntax-Erklärung

GLOBALS	Falls Sie das GLOBALS-Kommando ohne Parameter eingeben, erhalten Sie ein Fenster, das Ihnen die gegenwärtig gültigen Parameterwerte zeigt. Dort haben Sie auch die Möglichkeit, diese Werte zu ändern. Ausführliche Informationen zu diesem Fenster finden Sie im Abschnitt *Session-Parameter benutzen* im Dokument *Natural Studio benutzen*.
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. **Anmerkung:** Manche Session-Parameter können nicht über die Kommandozeile, sondern nur in dem oben genannten Fenster geändert werden.

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)
CF	Character for Terminal Commands
COMPR	Set RPC Buffer Compression
CPCVERR	Code Page Conversion Error
DBSHORT	Interpretation of Database Short Names
DC	Character for Decimal Point Notation
DFOUT	Date Format for Output
DFSTACK	Date Format for Stack

Parameter	Funktion (Kurztext)
DFTITLE	Output Format of Date in Standard Report Title
DU	Dump Generation
EJ	Page Eject
ENDIAN	Endian Mode for Compiled Objects
FCDP	Filler Character for Dynamically Protected Input Fields
FS	Default Format/Length Setting for User-Defined Variables
GFID	Global Format IDs
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
ML	Position of Message Line
NC	Use of Natural System Commands
OPF	Overwriting of Protected Fields by Helproutines
PM	Print Mode
PS	Page Size for Natural Reports
REINP	Issue Internal `REINPUT` Statement for Invalid Data
SA	Sound Terminal Alarm
SF	Spacing Factor
SM	Programming in Structured Mode
SYMGEN	Generate Symbol Table
THSEPCH	Thousands Separator Character
TIMEOUT	Wait Time for RPC Server Response
TRYALT	Try Alternative Server Address
WH	Wait for Record in Hold Status
XREF	Creation of XRef Data for Natural
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.

15 HELP

```
{ HELP } [ [NAT]nnnn   ]
{ ?    } [ USER[nnnn] ]
```

Mit dem Systemkommando `HELP` rufen Sie die Online-Hilfe zu Fehlermeldungen auf.

Anmerkung: Dieses Kommando kann nicht im Batch-Betrieb ausgeführt werden.

HELP	Zeigt das Hilfemenü 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 USER*nnnn*	Zeigt den Langtext der library-spezifischen Fehlermeldung Nummer *nnnn* aus der aktuellen Library an.

Siehe auch *Hilfe benutzen* in der Dokumentation *Natural Studio benutzen*.

16 INPL

INPL [R]

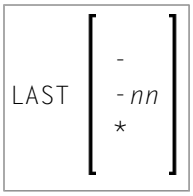
Mit dem INPL-Kommando rufen Sie die INPL-Utility auf. Diese Utility dient *nur* zum Laden von Software-AG-Installationsdatasets in die Systemdateien (wie in der Online-Hilfe und in den plattform-spezifischen Installationsschritten der Installationsdokumentation beschrieben).

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 *Tools and Utilities*-Dokumentation.

17 LAST



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

Sobald das Kommando in ein Dialogfeld 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 ein Dialogfeld gestellt und kann ausgeführt werden.
LAST -	Das zuletzt ausgeführte Kommando wird in ein Dialogfeld gestellt und kann ausgeführt werden. Wenn Sie LAST - noch einmal eingeben, wird das vorletzte Kommando in ein Dialogfeld gestellt und kann ausgeführt werden. Durch wiederholte Eingabe von LAST - können Sie so Kommando für Kommando „zurückblättern“.
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 ein Dialogfeld gestellt und kann ausgeführt werden.
LAST *	Es erscheint ein Dialogfeld, in der die 20 zuletzt abgesetzten Kommandos angezeigt werden.

	■ Wenn Sie die Kommandos erneut ausführen möchten, wählen Sie die gewünschten Kommandos aus und benutzen Sie die Schaltfläche **Copy** um die Kommandos in das Listenfeld **Selected Commands** zu kopieren. ■ Die in dem Listenfeld ausgewählten Kommandos können Sie vor dem Ausführen ändern.

18 LASTMSG

LASTMSG

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

Anmerkung: Dieses Kommando ist auch in einer Remote-Session verfügbar. Alle Informationen können auch im Batch-Betrieb gelesen werden.

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.

Siehe auch *Last Message (Letzte Meldung)* in der Dokumentation *Natural Studio benutzen*.

Um Informationen zu dem entsprechenden Fehler anzuzeigen

- Wählen Sie eine dieser Meldungen aus und benutzen Sie die Schaltfläche **Details** oder klicken Sie eine Meldung doppelt an.

  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.

19 LIST

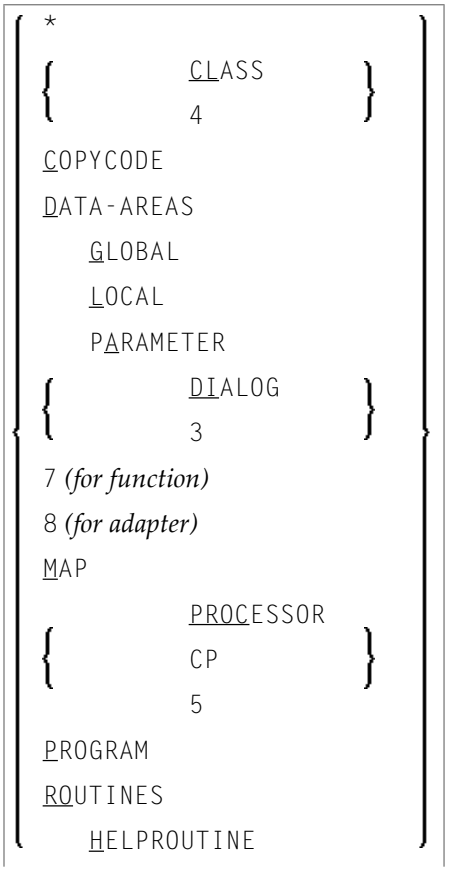
Mit dem Systemkommando `LIST` können Sie sich den Sourcecode eines einzelnen Objekts oder eines oder mehrere in Ihrer aktuellen Library gespeichterte Objekte auflisten lasten.

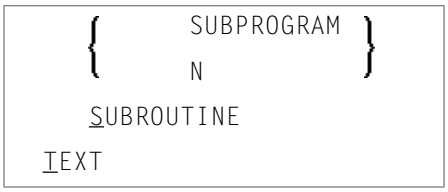
Anmerkung: Dieses Kommando kann nicht im Batch-Betrieb ausgeführt werden.

Siehe auch *Objekte auflisten* in der Dokumentation *Natural Studio benutzen* und die Beschreibungen der Systemkommandos `LIST XREF` und `LIST COUNT`.

Syntax-Übersicht

object-type





object-name

Für den *object-name* können Sie den Namen eines Objekts (maximal 8 Zeichen lang) angeben. Sie können auch Stern-Notation (*) benutzen, siehe **Beispiele**.

Eine einzelne Source anzeigen

LIST	Wenn Sie nur das Schlüsselwort LIST ohne Parameter eingeben, wird der Sourcecode des gerade ausgewählten Objekts gelistet.
LIST *object-name*	Wenn Sie einen einzelnen *object-name* mit dem LIST-Kommando eingeben, brauchen Sie keinen *object-type* anzugeben. Wenn Sie einen *object-type* angeben, müssen Sie auch einen *object-name* angeben. In beiden Fällen wird der Sourcecode des angegebenen Objekts gelistet.
LIST *object-type object-name*	

Eine Liste von Objekten anzeigen

LIST *object-name*	Wenn Sie sich alle Objekte, außer DDMs, in der aktuellen Library anzeigen lassen wollen, geben Sie einen Stern für den *object-name* an, aber keinen *object-type*. Wenn Sie sich alle Objekte eines bestimmten Typs anzeigen lassen wollen, geben Sie den gewünschten *object-type* und eine Stern (*) für den *object-name* an. Wenn Sie sich einen bestimmten Bereich von Objekten anzeigen lassen wollen, können Sie für den *object-name* Stern-Notation (*) und/oder Wildcard-Notation (?) benutzen.
LIST *object-type object-name*	

Beispiele

- Alle Objekte außer DDMs in der aktuellen Library, unabhängig vom Objekttyp, auflisten:

```
LIST *
```

- Alle Subroutinen in der aktuellen Library auflisten:

```
LIST S *
```

- Alle Objekte (jeden Typs) auflisten, deren Namen mit `SYS` anfangen:

```
LIST SYS*
```

- Alle Maps auflisten, deren Name mit `SYS` anfängt:

```
LIST M SYS*
```

- Die Directory-Informationen des Objekts `PRG01` in der aktuellen Library anzeigen:

```
LIST DIR PRG01
```

- Alle Objekte wie `NATAL`, `NATURAL`, `NAT`*`vr`*`AL` auflisten (dabei steht *`vr`* für die betreffende Versions- und Release-Nummer):

```
LIST N?T*AL
```

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), Natural-Version, Codepage-Informationen (falls verfügbar), Betriebssystem/Version, Größe, Codierung ■ **Objektcode:** „Cataloged-on"-Datum und -Uhrzeit, Library-Name, User-ID, Programmiermodus (Reporting oder Structured Mode), Natural-Version, Codepage-Informationen (falls verfügbar), Betriebssystem/Version, Endian-Modus Die Directory-Informationen zu der gespeicherten Source können nicht immer exakt sein, weil der Sourcecode mit Nicht-Natural-Editoren bearbeitet werden kann, die keiner Kontrolle durch Natural unterliegen.
LIST DIRECTORY *`object-name`*	Zeigt die Directory-Informationen zu dem angegebenen Objekt an. Wenn Sie Stern-Notation (*) für *`object-name`* verwenden, werden nacheinander die Directory-Informationen der entsprechenden Objekte angezeigt.

Anmerkung: Die angezeigten Codepage-Information zeigt nur die ersten 32 Zeichen der Codepage.

Views anzeigen

LIST VIEW	Zeigt eine Liste aller Views (DDMs).
LIST VIEW *view-name*	Wenn Sie einen einzelnen View-Namen angeben, wird die angegebene View angezeigt. Für den *view-name* können Sie Stern-Notation (*) oder einen bestimmten Bereich von Views angeben (zum Beispiel: A*).

Datei-Informationen von Resource-Objekten anzeigen

LIST RESOURCE *name*	Zeigt die Datei-Informationen zu dem angegebenen Resource-Objekt. Für *name* können Sie Stern-Notation (*) benutzen.

Beispiel: Die Datei-Informationen aller Resource-Objekte anzeigen, die mit einem `W` anfangen:

```
LIST RESOURCE W*
```

Datei-Informationen von Error Message Containers anzeigen

LIST ERROR *name*	Zeigt die Datei-Informationen zu dem angegebenen Error Message Container `N`*nn*`APMSL.MSG` anzeigen (dabei steht *nn* für den Sprachcode). Für *name* können Sie nur Stern-Notation (*) benutzen.

20 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.

21 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.

22 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.

23 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.

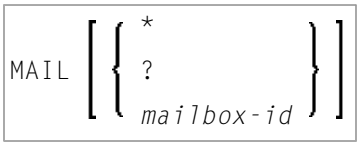
Das Kommando LOGON bewirkt, dass alle Natural Global Data Areas und anwendungsunabhängigen Variablen (AIVs), alle mit einem SET KEY-Statement vorgenommenen Zuordnungen und alle gehalten ISN-Listen freigegeben werde. Im DDM-Buffer-Bereich enthaltene DDMs werden ebenfalls freigegeben.

<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:</td></tr>
<tr><td></td><td>A - Z</td><td>Großbuchstaben</td></tr>
<tr><td></td><td>0 - 9</td><td>Ziffern</td></tr>
<tr><td></td><td>-</td><td>Bindestrich</td></tr>
<tr><td></td><td>_</td><td>Unterstrich</td></tr>
<tr><td></td><td colspan="2">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.

24 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.

25 MAP

Mit dem Systemkommando MAP können Sie die im Folgenden beschriebenen Funktionen über die Kommandozeile ausführen.

Verwandtes Kommando: UNMAP.

Eine Verbindung zu einer Natural Development Server-Umgebung herstellen

Die folgende MAP-Kommandosyntax gilt, wenn Sie über die Kommandozeile eine Verbindung zu einem Natural Development Server herstellen wollen:

```
MAP ENVIRONMENT=environment-name server-name port-name [userid [password ['parm=value;...']]]
```

Dieses Kommando hat dieselbe Wirkung wie der Dialog, der im Abschnitt *Entwicklungs-Server per Mapping zuordnen* in der *Remote-Entwicklung mit SPoD*-Dokumentation beschrieben ist.

environment-name	Alias-Name für die Verbindung. Wenn Sie den *environment-name* nicht angeben, wird ein Alias-Name in der Form *server(port)* erzeugt. Wenn der Name der Umgebung Leezeichen enthält, müssen Sie ihn in Hochkommas angeben ('... ...').
server-name	Der Name des Natural-Development-Servers auf dem Großrechner, dem UNIX-Server oder dem OpenVMS-Server.
port-name	Die TCP/IP-Port-Nummer des Development-Servers.
userid	Die User-ID für den Zugang zum Development-Server. Wenn Sie einen Stern (*) als *userid* angeben, wird die User-ID der Client Session verwendet.
password	Wenn Natural Security auf dem Development-Server installiert ist, müssen Sie das erforderliche Passwort angeben. Wenn Sie einen Stern (*) als *password* angeben, wird eine leere Passwort-Zeichenkette zum Development-Server gesendet.
parm	Wenn für Ihren Development-Server dynamische Parameter erforderlich sind, müssen Sie die Session-Parameter in Hochkommas angeben ('...').

Um das Mapping einer Session auf einem Natural-Development-Server zu beenden, können Sie das Systemkommando UNMAP verwenden.

Eine Verbindung zu einer Natural-Anwendung herstellen

Die folgende MAP-Kommandosyntax gilt, wenn Sie über die Kommandozeile eine Verbindung zu einer Natural-Anwendung herstellen wollen:

```
MAP APPLICATION=application-name[userid[password]
```

Dieses Kommando hat dieselbe Wirkung wie der Dialog, der im Abschnitt *Anwendungen per Mapping zuordnen und Zuordnung per Unmapping lösen* in der *Remote-Entwicklung mit SPoD*-Dokumentation beschrieben ist. Eine Erläuterung des Begriffs „Natural Anwendung“ finden Sie in der *Natural Single Point of Development*-Dokumentation.

application-name	Der Name der Natural-Anwendung.
userid	Die User-ID für den Zugang zur Anwendung. Wenn Sie einen Stern (*) als *userid* angeben, wird die User-ID der Client Session verwendet.
password	Wenn Natural Security auf dem Development-Server installiert ist, müssen Sie das erforderliche Passwort angeben. Wenn Sie einen Stern (*) als *password* angeben, wird eine leere Passwort-Zeichenkette zum Development-Server gesendet.

Um das Mapping einer Natural Anwendung auf einem Natural-Development-Server zu beenden, können Sie das Systemkommando UNMAP oder den Dialog verwenden, der im Abschnitt *Anwendungen per Mapping zuordnen und Zuordnung per Unmapping lösen* beschrieben ist.

26 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.

27 PURGE

PURGE [*object-name* ...]

Es empfiehlt sich, statt des PURGE-Kommandos das DELETE-Kommando zu verwenden, da DELETE mehr Flexibilität sowie Schutz gegen versehentliches Löschen bietet.

Das PURGE-Kommando dient dazu, ein oder mehrere Programmierobjekte in Sourceform aus der Natural-Systemdatei zu löschen.

Anmerkung: Wenn der Natural-Profilparameter RECAT auf ON gesetzt ist, wird das PURGE-Kommando für ein Programmierobjekte in Sourceform, zu dem das entsprechende katalogisierte Objekt existiert, zurückgewiesen.

PURGE	Wenn Sie nur PURGE (ohne *object-name*) eingeben, erhalten Sie eine Liste aller in der aktuellen Library in Sourceform gespeicherten Programmierobjekte. Auf der Liste markieren Sie die Objekte, die Sie löschen möchten.
PURGE *object-name*	Als *object-name* geben Sie den Namen des Objektes an, das Sie löschen möchten. Sie können auch mehrere Namen angeben. Sie können nur Objekte löschen, die in Ihrer aktuellen Library gespeichert sind. Um nur einen bestimmten Bereich von Objekten aufgelistet zu bekommen, können Sie den *object-name* mit Stern-Notation (*) angeben.

28 READ

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

Verwandtes Kommando: EDIT

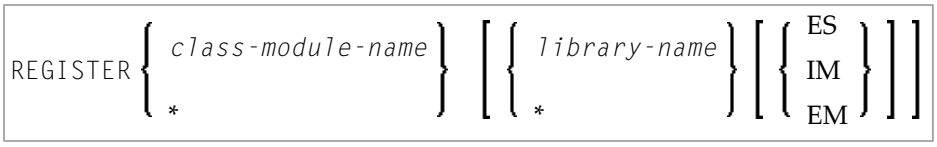
Mit dem Kommando READ können Sie ein in Sourceform gespeichertes Objekt in den Arbeitsbereich des entsprechenden Editors einlesen.

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

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

29 REGISTER



Verwandtes Kommando: UNREGISTER.

Mit dem Kommando REGISTER können Sie Natural-Klassen registrieren. Die Registrierung erfolgt für die Server-ID, unter der Natural gestartet wurde.

Weitere Informationen siehe *The REGISTER Command* im *Administrating NaturalX Applications*-Teil der *Operations*-Dokumentation.

30 RENAME

Dieses Kommando ist in einer Remote-Entwicklungsumgebung nicht über die Kommandozeile verfügbar.

```
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 Studio 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*:

3	Dialog
4	Klasse
5	Processor
7	Function

	8	Adapter
	9	Resource
	A	Parameter Data Area
	C	Copycode
	G	Global Data Area
	H	Helproutine
	L	Local Data Area
	M	Map
	N	Subprogramm
	P	Programm
	S	Subroutine
	T	Text
	Y	Rule
	Z	Recording

Siehe auch *Objekte umbenennen* in der Dokumentation *Natural Studio benutzen*.

31 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.

32 RETURN

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

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.

33 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.

34 RUN

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

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

Siehe auch:

Objekte mit Run ausführen in *Natural Studio benutzen*
Namenskonventionen für Objekte in der Dokumentation *Natural Studio benutzen*

RUN	Wenn Sie den *program name* nicht angeben, dann kompiliert Natural das Programm oder den Dialog, der sich gegenwärtig im Editor-Arbeitsbereich befindet, und führt es bzw. ihn aus.
REPEAT	Wenn das auszuführende Programm oder der auszuführende Dialog 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 oder des Dialogs wechselt Natural in den Kommando-Modus.
program-name	Der Name des Programms oder Dialogs, das bzw. der kompiliert und ausgeführt werden soll. Wenn Sie den *program-name* ohne Library-ID angeben, wird das Programm bzw. der Dialog in den Arbeitsbereich gelesen (wobei ein bereits im Arbeitsbereich befindliches Objekt überschrieben wird), kompiliert und ausgeführt, vorausgesetzt es bzw. er ist in Ihrer aktuellen Library gespeichert. Wenn es bzw. er nicht unter der aktuellen Library-ID gespeichert ist, erscheint eine Fehlermeldung.
library-id	Die Library in der sich das gewünschte Programm oder der Dialog 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 oder der Dialog nur dann kompiliert und ausgeführt, wenn es bzw. er unter der angegebenen Library-ID gespeichert ist. Wenn es bzw. er 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.

35 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.

See also: .

Objekte speichern in *Natural Studio benutzen*
Namenskonventionen für Objekte in *Natural Studio benutzen*

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 Library, aus der das Objekt in den Editor-Arbeitsbereich gelesen wurde (z.B. mit EDIT oder READ) 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.

36 SCAN

SCAN

Mit dem SCAN-Kommando rufen Sie ein Dialogfeld auf, das Sie zum Suchen nach Natural-Objekten und Zeichenketten benutzen können. Weitere Informationen zu diesem Dialogfeld siehe *Objekte in einer Library finden* in der Dokumentation *Natural Studio benutzen*.

Anmerkung: Dieses Kommando kann nicht im Batch-Betrieb ausgeführt werden.

37 SCRATCH

Mit diesem Kommando können Sie ein oder mehrere Objekte sowohl in Source- als auch in Objektform löschen. Der Inhalt des Arbeitsbereichs wird dadurch nicht beeinflußt.

SCRATCH **SCRATCH ***	Wenn Sie das SCRATCH-Kommando ohne *object-name* oder ohne *object-name*, aber mit einem Stern (*) eingeben, wird eine Liste aller Objekte oder aller ausgewählten Objekte in der aktuellen Library angezeigt. In der Liste können Sie dann die zu löschenden Objekte markieren.
SCRATCH *object-name*	Als *object-name* geben Sie den Namen des zu löschenden Objekts an. Sie können nur Objekte löschen, die in der aktuellen Library gespeichert sind. Um alle Objekte zu löschen, deren Namen mit bestimmten Zeichen beginnen, können Sie Stern-Notation (*) für *object-name* verwenden.

Anmerkung: Falls in der Parameterdatei eine ungültige Systemdatei FDIC angegeben ist, erscheint eine entsprechende Natural-Fehlermeldung, wenn das SCRATCH-Kommando abgesetzt wird.

38 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.

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.

39 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: .

Stowing Objects in *Natural Studio benutzen*
Namenskonventionen für Objekte in *Natural Studio benutzen*

STOW	Wenn Sie das Kommando ohne *object-name* benutzen, wird das Sourceobjekt im Arbeitsbereich des Editors und der erzeugte Code in der Library unter dem Namen des zuletzt in den Arbeitsbereich (z.B. mit EDIT or READ) eingelesenen Objekts gespeichert.
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 in der Parameterdatei eine ungültige Systemdatei FDIC angegeben ist, erscheint eine entsprechende Natural-Fehlermeldung, wenn das STOW-Kommando abgesetzt wird.

40 STRUCT

```
STRUCT [(n)]
```

Sie können das STRUCT-Kommando verwenden, um die Sourcecode-Zeilen des aktuellen Programmierobjekts im Arbeitsbereich des Editors entsprechend der Programmstruktur einzurücken.

STRUCT	Standardmäßig (d.h., wenn *n* nicht angegeben wird) wird um 2 Stellen eingerückt.
STRUCT (*n*)	Den Parameter "(*n*)" können Sie benutzen, um die Anzahl der zum Einrücken verwendeten Leerzeichen anzugeben. Mögliche Werte: 1 - 9. Beispiel: `STRUCT (5)`

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.

Indentation of Source Code Lines (Einrückung von Sourcecode-Zeilen)

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

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 das STRUCT-Kommando 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 es eingerückt 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
```

41 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 *Tools and Utilities*-Dokumentation.

42 SYSCP

SYSCP

Mit diesem Kommando rufen Sie die SYSCP-Utility auf.

Die SYSCP-Utility kann benutzt werden, um Informationen über Codepages zu erhalten.

Weitere Informationen siehe *SYSCP Utility - Code Page Information* in der *Tools and Utilities*-Dokumentation.

43 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 *Tools and Utilities*-Dokumentation.

44 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 *Tools and Utilities*-Dokumentation.

45 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 *Tools and Utilities*-Dokumentation.

46 SYSFILE

```
SYSFILE
```

Mit diesem Kommando können Sie Informationen zu den verfügbaren Arbeitsdateien und Druckdateien (Work und Print Files) anzeigen. Sie erhalten Informationen über

- Reports
- logische Geräte
- definierte physische Geräte
- definierte Druckerprofile
- definierte Work Files

Siehe auch *Work and Print Files (Arbeits- und Druckdateien)* in der Dokumentation *Using Natural Studio*.

Weitere Informationen zu Arbeitsdateien und Druckdateien finden Sie in folgenden Dokumenten:

- *Printer Profiles* in der *Configuration Utility*-Dokumentation
- *Device/Report Assignments* in der *Configuration Utility*-Dokumentation
- *Work Files* in der *Operations*-Dokumentation

SYSFILE in einer Remote Mainframe-Umgebung

Sie erhalten folgende Informationen zu Arbeitsdateien und Druckdateien (Work und Print Files):

- Zuweisungsart
- Datensatzformat
- logische Datensatzlänge
- Blockgröße
- Status
- dynamische Parameterangabe

47 SYSINST

SYSINST

Mit diesem Kommando rufen Sie den Natural Installer auf. Diese Utility unterstützt Sie bei der Installation von Natural-Add-on-Produkten, z.B. Natural Security (NSC).

Weitere Informationen siehe *Installer* in der *Tools and Utilities*-Dokumentation.

48 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* in der *Tools and Utilities*-Dokumentation.

Anmerkung: Dieses Kommando kann nicht im Batch-Betrieb ausgeführt werden.

49 SYSMN

SYSMN

Mit diesem Kommando rufen Sie das Mainframe Navigation-Plug-in auf.

Weitere Informationen finden Sie in der Mainframe Navigation-Dokumentation.

50 SYSNCP

SYSNCP

Mit diesem Kommando rufen Sie die SYSNCP-Utility auf.

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

51 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 *Tools and Utilities*-Dokumentation.

52 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.

Wenn Sie das Kommando eingeben, werden in einem Dialog zu jedem der installierten Produkte z.B. folgende Informationen angezeigt:

- Produktname
- Produktversion (siehe auch *Version* im *Glossary*)
- Installationszeitpunkt (Datum und Uhrzeit)
- Produktkennung (ID)

Siehe auch *Product Information (Produktinformationen)* in der Dokumentation *Natural Studio benutzen*.

53 SYSPROF

SYSPROF

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

Zu jeder Systemdatei werden folgende Informationen (auf der Seite **System Files**)angezeigt.

- Dateiname
- Datenbank-ID
- Dateinummer
- Datenbanktyp

Darüber hinaus können Sie sich zu jeder angegebenen Kombination von Datenbank-ID und Dateinummer die folgenden Informationen anzeigen lassen:

- den Pfad im Dateisystem (auf der Seite **Files in File System**)
- die logische Dateinummer, falls zugewiesen (auf der Seite **All Files**)

Siehe auch *System Files (Systemdateien)* in der Dokumentation *Natural Studio benutzen*.

54 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 *Tools and 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.

55 SYSWIZDB

SYSWIZDB

Mit diesem Kommando können Sie innerhalb von Natural Studio den Data Browser aufrufen, mit dem Sie Dateistrukturen zeigen, drucken und speichern können.

Weitere Informationen siehe *Data Browser* in the *Tools and Utilities*-Dokumentation.

56 SYSWIZDW

```
SYSWIZDW
```

Mit diesem Kommando können Sie den Dialog Wizard aufrufen, mit dem Sie Dialoge für spezielle Zwecke erstellen können. Diese Dialoge können zur Anpassung an die gewünschten Anforderungen mehrere Layouts haben.

Weitere Informationen siehe *Dialog Wizard* im Abschnitt *Dialog Editor* der *Editors*-Dokumentation.

57 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.

Siehe auch *Technical Information (Technische Informationen)* in der Dokumentation *Natural Studio benutzen*.

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. Zusätzlich werden folgende Informationen angezeigt: Namen, Typen und Ebenen aller gerade aktiven Programmierobjekte und aller Objekte auf höheren Ebenen
2. Dieses Kommando steht auch in einer Remote-Session zur Verfügung. Alle Informationen können auch im Batch-Betrieb gelesen werden.

58 UNCATALOG

UNCATALOG [*object-name* ...]

Das UNCATALOG-Kommando dient dazu, ein oder mehrere Programmierobjekte in Objektform aus der Natural-Systemdatei zu löschen.

Um Inkonsistenzen zu vermeiden, empfiehlt es sich, das Menükommando **Delete** zu verwenden, um sowohl den Sourcecode als auch das Objektmodul eines Objekts zu löschen. Siehe *Objekte löschen* in der Dokumentation *Natural Studio benutzen*.

Sie können nur die Objekte löschen, die in der Library gespeichert sind, für die Sie sich angemeldet haben. Der Inhalt des Arbeitsbereichs wird durch das UNCATALOG-Kommando nicht verändert,

UNCAT **UNCAT ***	Wenn Sie nur UNCATALOG (ohne *object-name*) oder UNCATALOG mit Stern-Notation (*) eingeben, erhalten Sie eine Liste aller in der aktuellen Library in Objektform gespeicherten Programmierobjekte. Auf der Liste markieren Sie die Objekte, die Sie löschen möchten.
UNCAT *object name*	Als *object-name* geben Sie den Namen des Objektes an, das Sie löschen möchten. Sie können nur Objekte löschen, die in Ihrer aktuellen Library gespeichert sind. Sie können auch mehrere Namen angeben, dann müssen die *object-names* durch ein oder mehrere Leerzeichen (bzw. das zur Zeit definierte Deliminiter-Zeichen) getrennt werden. Um alle Objekte zu löschen, deren Namen mit einer bestimmten Zeichenkette beginnen, können Sie für *object-name* Stern-Notation (*) verwenden. Dann erscheint eine Liste mit allen ausgewählten Objekten, in der Sie die zu löschenden Objekte markieren können.

Anmerkung: Falls in der Parameterdatei eine ungültige Systemdatei FDIC angegeben ist, erscheint eine entsprechende Natural-Fehlermeldung, wenn das UNCATALOG-Kommando abgesetzt wird.

59 UNLOCK

Mit dem UNLOCK-Kommando können Sie folgende Objekttypen entsperren:

- Natural-Sourceobjekte in einer Remote-Development-Umgebung.
- Dokumentationsobjekte in der lokalen Entwicklungsumgebung, vorausgesetzt Predict Version 4.4 oder höher ist in der Windows-Umgebung installiert.
- Dokumentationsobjekte in der lokalen Entwicklungsumgebung, vorausgesetzt das Object Description-Plug-in ist installiert (siehe separate Object Description-Dokumentation).

Das UNLOCK-Kommando dient dazu, Sourceobjekte oder Dokumentationsobjekte 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.

Weitere Informationen siehe *Objekte manuell entsperren* in der *Remote-Entwicklung mit SPoD*-Dokumentation.

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

Natural-Objekte entsperren

Wenn Sie das Systemkommando UNLOCK ohne Parameter benutzen, erscheint ein Dialog, in dem 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]
       [APPLICATION application-name]
       [USER locked-by]
       [DATE locked-on [locked-on2]]
```

Dokumentationsobjekte entsperren

Die folgende Übersicht zeigt die Kommandosyntax zum Entsperren von Dokumentationsobjekten:

```
UNLOCK DOCUMENT [OBJECT] object-name
       [TYPE object-type]
       [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
3	Dialog
M	Map
L	Local Data Area
G	Global Data Area

Parameter	Format/Länge	Standardwert	Beschreibung
			A: Parameter Data Area V: DDM (View) X: Applikation
	A2	*	**Dokumentationsobjekttypen:** Benutzerdefinierte Kurzbeschreibungen für Dokumentationsobjekttypen oder Stern-Notation (*).
library-name	A8	*	Name der Library, in der das gesperrte Objekt gespeichert ist. Sie können Stern-Notation (*) benutzen.
dbid	N5	aktuelle Datenbank-ID	Datenbank-ID der angegebenen Library. Sie können Stern-Notation (*) benutzen oder die ID im Format N5 angeben. Bei Mainframe-Servern mit Parameter SLOCK=PRE gilt Folgendes: Wenn Sie Stern-Notation (*) benutzen, werden nur die aktuellen Systemdateien FNAT und FUSER abgesucht.
fnr	N5	aktuelle Dateinummer	Dateinummer der angegebenen Library. Sie können Stern-Notation (*) benutzen oder die Nummer im Format N5 angeben. Bei Mainframe-Servern mit Parameter SLOCK=PRE gilt Folgendes: Wenn Sie Stern-Notation (*) benutzen, werden nur die aktuellen Systemdateien FNAT und FUSER abgesucht.
password	A8	leer	Falls verwendet, das Passwort für die angegebene Systemdatei (*dbid* und *fnr*). Sie brauchen kein Passwort anzugeben, wenn Sie die *dbid* und *fnr* der aktuellen Datei FNAT oder FUSER benutzen. Dieser Parameter steht nur in einer Großrechner-Remote-Development-Umgebung zur Verfügung und wenn der Profilparameter SLOCK=PRE in der Großrechnerumgebung gesetzt ist.
cipher	A8	leer	Falls verwendet, der Chiffrierschlüssel für die angegebene Systemdatei (*dbid* und *fnr*). Sie brauchen keinen Chiffrierschlüssel anzugeben, wenn Sie die *dbid* und *fnr* der aktuellen Datei FNAT oder FUSER benutzen. Dieser Parameter steht nur in einer Großrechner-Remote-Development-Umgebung zur

Parameter	Format/Länge	Standardwert	Beschreibung
			Verfügung und wenn der Profilparameter SLOCK=PRE in der Großrechnerumgebung gesetzt ist.
application-name	A32	leer	Falls verwendet, der Name der Anwendung, zu der das gesperrte Objekt gehört. Wenn Sie ein Leerzeichen angeben, werden alle gesperrten Objekte in einem Ergebnisfenster angezeigt - unabhängig davon, ob sie zu einer Anwendung gelinkt sind oder nicht. Sie können sie dann in diesem Fenster manuell entsperren.
locked-by	A8	aktuelle Benutzer-ID	Die ID des Benutzers, der die Sperrung des Objekts veranlasst hat. Sie können Stern-Notation (*) verwenden. Wenn Natural Security verwendet wird, kann der Parameter nur geändert werden, wenn im Natural Security-Benutzerprofil der Security-Unlock-Flag auf "F" (Forced Unlock) gesetzt ist.
locked-on	A10	leer	Die beiden Datumsparameter sind vorhanden, um die verschiedenen Datumsformate abzudecken: 2005-09-28 (Datumsformat gemäß DTFORM-Profilparameter) 2005-09-28 11:27:20 Heute Heute + *nnnn* Heute - *nnnn* Gestern
locked-on2	A8		

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 in einem Ergebnisfenster angezeigt und können manuell entsperrt werden:

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

Falls der Objekttyp bei einem Dokumentationsobjekt nicht eindeutig ist, können Sie in der verdeckt angezeigten Spalte neben dem Objekttyp nach den internen Objekttypen nachsehen.

Weitere Informationen zum Ergebnisfenster siehe *Objekte manuell entsperren* in der *Remote-Entwicklung mit SPoD*-Dokumentation.

60 UNMAP

Mit dem UNMAP-Kommando können Sie über die Natural-Kommandozeile die im Folgenden beschriebenen Funktionen ausführen:

Wenn Sie sich auf einem Natural Development Server oder in einer Natural-Anwendung per Mapping anmelden wollen, können Sie das Systemkommando MAP oder den Dialog benutzen, der im Abschnitt *Anwendungen per Mapping zuordnen und Zuordnung per Unmapping lösen* in der *Remote-Entwicklung mit SPoD*-Dokumentation beschrieben wird.

Unmap-Operation für die gerade aktive Umgebung/Anwendung ausführen

Die folgende Kommandosyntax gilt, wenn Sie eine Unmap-Operation für die gerade aktive Natural Development Server-Umgebung oder eine Natural-Anwendung ausführen wollen:

```
UNMAP
```

Unmap-Operation für eine Natural Development Server-Umgebung ausführen

Die folgende Kommandosyntax gilt, wenn Sie eine Unmap-Operation für eine Natural Development Server-Umgebung ausführen wollen:

```
UNMAP ENVIRONMENT=environment-name
```

Dabei ist *environment-name* der Alias-Name der Verbindung. Wenn der Name der Umgebung Leerzeichen enthält, müssen Sie ihn in Hochkommas angeben ('...').

Unmap-Operation für eine Natural-Anwendung ausführen

Die folgende Kommandosyntax gilt, wenn Sie eine Unmap-Operation für eine Natural-Anwendung ausführen wollen:

```
UNMAP APPLICATION=application-name
```

Dabei ist *application-name* der Name der betreffenden Anwendung.

61 UNREGISTER

```
UNREGISTER { class-module-name } [ { library-name } [server-id] ]
           { *                 }   { *            }
```

Verwandtes Kommando: REGISTER

Mit diesem Kommando können Sie die Registrierung von Natural-Klassen rückgängig machen.

Weitere Informationen siehe *The UNREGISTER Command* im Teil *Administrating NaturalX Applications* der *Operations*-Dokumentation.

Anmerkung: Unter Natural Security kann dieses Kommando nur für eine einzelne Library ausgeführt werden. Das heißt, dass Sie entweder den Library-Namen weglassen oder eine spezifische Library benutzen müssen. Sie können keine Stern-Notation (*) verwenden.

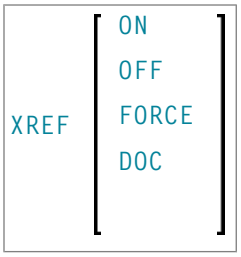
62 UPDATE

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.

63 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 einen Dialog, 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 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.