PARSE XML

PARSE XML *operand1* [**INTO** [**PATH** *operand2*] [**NAME** *operand3*] [**VALUE** *operand4*]]
[[**NORMALIZE**] **NAMESPACE** *operand5* **PREFIX** *operand6*]
statement...
END-PARSE *(structured mode only)*
[**LOOP**] *(reporting mode only)*

Dieses Kapitel behandelt folgende Themen:

- Funktion
- Syntax-Beschreibung
- Beispiele

Eine Erläuterung der in dem Syntax-Diagramm verwendeten Symbole entnehmen Sie dem Abschnitt *Syntax-Symbole*.

Gehört zur Funktionsgruppe: *Internet und XML*

Funktion

Das Statement PARSE XML ermöglicht es Ihnen, XML-Dokumente aus einem Natural-Programm zu parsen. Als Voraussetzung für die Benutzung dieses Statements muss die Library ICU installiert sein. Siehe auch *Statements für Internet- und XML-Zugang* im *Leitfaden zur Programmierung*.

Es empfiehlt sich, dynamische Variablen zu benutzen, wenn Sie das Statement PARSE XML verwenden. Der Grund dafür ist, dass es unmöglich ist, die Länge einer statischen Variablen zu ermitteln. Der Einsatz von statischen Variablen könnte wiederum zum Abschneiden des Wertes führen, der in die Variable geschrieben werden soll.

Informationen bezüglich Unicode-Support finden Sie im Abschnitt PARSE XML in der *Unicode and Code Page Support*-Dokumentation.

Markierungen

Im Folgenden finden Sie Bezeichner, die in Pfad-Zeichenketten zur Darstellung der verschiedenen Datentypen in einem XML-Dokument (auf ASCII-basierten Systemen) benutzt werden:

Markierung	XML-Daten	Position in der Pfad-Zeichenkette
?	Verarbeitungsanweisung (außer bei <?XML...?>)	Ende
!	Kommentar	Ende
C	CDATA-Abschnitt	Ende
@	Attribut (auf Großrechnern: § oder @, je nach Code Page und Terminal Emulation)	vor dem Attribut-Namen
/	Abschließender Tag und/oder Trennzeichen für Parent-Namen in einem Pfad	Ende oder zwischen Parent-Namen
$	Geparste Daten-Zeichendatenkette	Ende

Wenn Sie diese zusätzlichen Markierungen im Pfad-String benutzen, ist es leichter, die verschiedenen Elemente des XML-Dokuments im Ausgabedokument zu identifizieren.

Global Namespace

Zur Angabe des Global Namespace verwenden Sie einen Doppelpunkt (:) als Präfix und eine leere URI.

Zugehörige Systemvariablen

Die folgenden Natural-Systemvariablen werden für jedes abgesetzte PARSE-Statement automatisch erzeugt:

- *PARSE-TYPE
- *PARSE-LEVEL
- *PARSE-ROW
- *PARSE-COL
- *PARSE-NAMESPACE-URI

Durch Angabe der Notation (*r*) hinter *PARSE-TYPE, *PARSE-LEVEL, *PARSE-ROW, *PARSE-COL und *PARSE-NAMESPACE-URI können Sie den Statement-Label bzw. die Sourcecode-Zeilennummer des Statements angeben, in dem bzw. in der die PARSE-Anweisung abgesetzt wurde. Wenn (*r*) nicht angegeben wird, stellt die betreffende Systemvariable die Systemvariable der XML-Daten dar, die gerade in der zur Zeit aktiven PARSE-Verarbeitungsschleife abgearbeitet werden.

Weitere Informationen über diese Systemvariablen finden Sie in der *Systemvariablen*-Dokumentation.

Syntax-Beschreibung

Operanden-Definitionstabelle:

Operand	Mögliche Struktur					Mögliche Formate												Referenzierung erlaubt	Dynam. Definition
operand1	C	S				A	U	B										ja	nein
operand2		S				A	U	B										ja	ja
operand3		S				A	U	B										ja	ja
operand4		S				A	U	B										ja	ja
operand5		S	A			A	U	B										ja	ja
operand6		S	A			A	U	B										ja	ja

Syntax-Element-Beschreibung:

operand1	*operand1* stellt das betreffende XML-Dokument dar. Das XML-Dokument kann nicht geändert werden, während es vom Parser abgearbeitet wird. Wenn Sie versuchen, während des Parse-Vorgangs das XML-Dokument zu ändern (indem Sie es zum Beispiel überschreiben), wird eine Fehlermeldung angezeigt.
operand2	*operand2* stellt den Pfad der Daten im XML-Dokument dar. Der Pfad (`PATH`) enthält den Namen des identifizierten XML-Teils, die Namen aller Parents (übergeordneten Elemente), sowie den Typ des XML-Teils. **Anmerkung:** Die mit dem Pfad angegebenen Informationen erleichtern den Aufbau einer Baumstruktur. Siehe auch *Beispiel 1 – Benutzung von operand2*.
operand3	*operand3* stellt den Namen (`NAME`) eines Datenelements im XML-Dokument dar. Wenn `NAME` keinen Wert hat, dann wird die damit verbundene dynamische Variable auf `*LENGTH()=0` gesetzt, welche eine mit einem Leerzeichen aufgefüllte statische Variable ist. Siehe auch *Beispiel 2 – Benutzung von operand3*.
operand4	*operand4* stellt den Wert (`VALUE`) eines Datenelements im XML-Dokument dar. Ist kein Wert vorhanden, wird eine gegebene dynamische Variable auf `*LENGTH()=0` gesetzt, welche eine mit einem Leerzeichen aufgefüllte statische Variable ist. Siehe auch *Beispiel 3 – Benutzung von operand4*.

operand5* und *operand6 **NORMALIZE NAMESPACE** **PREFIX**	Der eindeutige Namen gewährleistende Namespace-URI oder Uniform Resource Identifier (*operand5*) und das Namespace-PREFIX (*operand6*) werden während der Laufzeit kopiert. Deshalb beeinflusst eine Änderung der Arrays für Namespace-Zuordnungen in der PARSE-Schleife nicht den Parser.

	operand5 und *operand6* sind eindimensionale Arrays mit einer gleichen Anzahl von Ausprägungen.
	
	Namespace-Normalisierung ist eine Funktion des PARSE-Statements. Mit XML können Namespaces für die Element-Namen definiert werden:.
	
	`<myns:myentity xmlns:myns="http://myuri" />`
	
	Die Namespace-Definition besteht aus zwei Teilen:
	
	● einem Namespace-PREFIX (`myns` in diesem Fall) und
	
	● ein URI (`myuri`) zur Definition des Namespace.
	
	Der Namespace-PREFIX ist Teil des Elementnamens. Dies bedeutet, dass für das PARSE-Statement, vor allem für *operand2*, die generierten PATH-Strings vom Namespace-PREFIX abhängig sind. Wenn der Pfad in einem Natural-Programm zur Angabe bestimmter Tags benutzt wird, dann funktioniert das nicht, wenn ein XML-Dokument den korrekten Namespace (URI), jedoch einen anderen PREFIX verwendet..
	
	Bei der Namespace-Normalisierung können alle Namespace-Präfixe auf Standardwerte gesetzt werden, die in der Namespace-Klausel definiert wurden. Der erste Eintrag wird dann benutzt, wenn ein URI mehr als einmal angegeben wird. Wenn mehr als ein Präfix im XML-Dokument benutzt wird, dann wird nur das erste für die Ausgabe berücksichtigt. Der Rest wird ignoriert.
	
	Die NAMESPACE-Klausel enthält Paare von Namespace-URIs und -Präfixe. Zum Beispiel:
	
	`uri(1) := 'http://namespaces.softwareag.com/natural/demo'`
	`pre(1) := 'nat:'`
	
	Wenn der Namespace in einem XML-Dokument definiert wird, prüft der Parser, ob dieser Namespace (URI) in der Normalisierungs-Tabelle vorhanden ist. Das Präfix der Normalisierungs-Tabelle wird für alle Ausgabedaten vom PARSE-Statement anstatt des im XML-Dokument definierten Namespace benutzt.
	
	Siehe auch:
	
	● *Beispiel 4 – Benutzung von operand5 und operand6*
	
	● *Beispiel 5 – Benutzung von operand5 und operand6 mit Namespace-Normalisierung*

	Zusätzliche Informationen zu PREFIX: Außerdem gilt Folgendes für die Präfix-Definition: ● Die Präfix-Definition beim Array für die Namespace-Normalisierung muss stets mit einem Doppelpunkt (:) enden, da dies die Zeichenkette ist, die ersetzt wird. ● Ein Präfix oder ein URI kann nur einmal bei einem Array für eine Namespace-Normalisierung vorkommen. ● Enthält ein Präfix oder der Namespace-URI nachfolgende Leerzeichen (z.B. wenn eine statische Variable verwendet wird), werden die nachfolgenden Leerzeichen entfernt, bevor der externe Parser aufgerufen wird. ● Wenn die Präfix-Definition bei der Namespace-Normalisierung nur einen Doppelpunkt (:) enthält, dann wird das Namespace-Präfix gelöscht.

Beispiele

- Beispiel 1 — Benutzung von operand2
- Beispiel 2 — Benutzung von operand3
- Beispiel 3 — Benutzung von operand4
- Beispiel 4 — Benutzung von operand5 und operand6
- Beispiel 5 — Benutzung von operand5 und operand6 mit Namespace-Normalisierung

Beispiel 1 — Benutzung von *operand2*

Der folgende XML-Code

```
myxml := '<?xml version="1.0"  ?>'-
         '<employee personnel-id="30016315" >'-
         '<full-name>'-
         '<!--this is just a comment-->'-
         '<first-name>RICHARD</first-name>'-
         '<name>FORDHAM</name>'-
         '</full-name>'-
         '</employee>'
```

wird durch folgenden Natural-Code verarbeitet:

```
PARSE XML myxml INTO PATH mypath
  PRINT mypath
END-PARSE
```

und erzeugt die folgende Ausgabe:

```
employee
employee/@personnel-id
employee/full-name
employee/full-name/!
employee/full-name/first-name
employee/full-name/first-name/$
employee/full-name/first-name//
employee/full-name/name
employee/full-name/name/$
employee/full-name/name//
employee/full-name//
employee//
```

Beispiel 2 — Benutzung von *operand3*

Der folgende XML-Code

```
myxml := '<?xml version="1.0"  ?>'-
         '<employee personnel-id="30016315" >'-
         '<full-name>'-
         '<!--this is just a comment-->'-
         '<first-name>RICHARD</first-name>'-
         '<name>FORDHAM</name>'-
         '</full-name>'-
         '</employee>'
```

wird durch folgenden Natural-Code verarbeitet:

```
PARSE XML myxml INTO PATH mypath NAME myname
  DISPLAY (AL=39) mypath myname
END-PARSE
```

und erzeugt die folgende Ausgabe:

```
               MYPATH                             MYNAME
---------------------------------    ---------------------------------


employee                             employee
employee/@personnel-id               personnel-id
employee/full-name                   full-name
employee/full-name/!
employee/full-name/first-name        first-name
employee/full-name/first-name/$
employee/full-name/first-name//      first-name
employee/full-name/name              name
employee/full-name/name/$
employee/full-name/name//            name
employee/full-name//                 full-name
employee//                           employee
```

Beispiel 3 — Benutzung von *operand4*

Der folgende XML-Code

```
myxml := '<?xml version="1.0"  ?>'-
         '<employee personnel-id="30016315" >'-
         '<full-name>'-
         '<!--this is just a comment-->'-
         '<first-name>RICHARD</first-name>'-
         '<name>FORDHAM</name>'-
         '</full-name>'-
         '</employee>'
```

wird durch folgenden Natural-Code verarbeitet:

```
PARSE XML myxml INTO PATH mypath VALUE myvalue
  DISPLAY (AL=39) mypath myvalue
END-PARSE
```

und erzeugt die folgende Ausgabe:

```
               MYPATH                             MYVALUE
---------------------------------    ---------------------------------


employee
employee/@personnel-id               30016315
employee/full-name
employee/full-name/!                 this is just a comment
employee/full-name/first-name
employee/full-name/first-name/$      RICHARD
employee/full-name/first-name//
employee/full-name/name
```

```
employee/full-name/name/$             FORDHAM
employee/full-name/name//
employee/full-name//
employee//
```

Beispiel 4 — Benutzung von *operand5* und *operand6*

Der folgende XML-Code

```
myxml := '<?xml version="1.0"  ?>'-
         '<nat:employee nat:personnel-id="30016315"'-
         ' xmlns:nat="http://namespaces.softwareag.com/natural/demo">'-
         '<nat:full-Name>'-
         '<nat:first-name>RICHARD</nat:first-name>'-
         '<nat:name>FORDHAM</nat:name>'-
         '</nat:full-Name>'-
         '</nat:employee>'
```

wird durch folgenden Natural-Code verarbeitet:

```
PARSE XML myxml INTO PATH mypath
  PRINT mypath
END-PARSE
```

und erzeugt die folgende Ausgabe:

```
nat:employee
nat:employee/@nat:personnel-id
nat:employee/@xmlns:nat
nat:employee/nat:full-Name
nat:employee/nat:full-Name/nat:first-name
nat:employee/nat:full-Name/nat:first-name/$
nat:employee/nat:full-Name/nat:first-name//
nat:employee/nat:full-Name/nat:name
nat:employee/nat:full-Name/nat:name/$
nat:employee/nat:full-Name/nat:name//
nat:employee/nat:full-Name//
nat:employee//
```

Beispiel 5 — Benutzung von *operand5* und *operand6* mit Namespace-Normalisierung

Wird `NORMALIZE NAMESPACE` verwendet, erzeugt dasselbe XML-Dokument wie in Beispiel 4 mit einem anderen `NAMESPACE PREFIX` genau dieselbe Ausgabe.

XML-Code:

```
myxml := '<?xml version="1.0"  ?>'-
         '<natural:employee natural:personnel-id="30016315"'-
         ' xmlns:natural="http://namespaces.softwareag.com/natural/demo">'-
         '<natural:full-Name>'-
         '<natural:first-name>RICHARD</natural:first-name>'-
         '<natural:name>FORDHAM</natural:name>'-
         '</natural:full-Name>'-
         '</natural:employee>'
```

Natural-Code:

```
uri(1) := 'http://namespaces.softwareag.com/natural/demo'
pre(1) := 'nat:'
*
PARSE XML myxml INTO PATH mypath NORMALIZE NAMESPACE uri(*) PREFIX pre(*)
  PRINT mypath
END-PARSE
```

Ausgabe des obigen Programms

```
nat:employee
nat:employee/@nat:personnel-id
nat:employee/@xmlns:nat
nat:employee/nat:full-Name
nat:employee/nat:full-Name/nat:first-name
nat:employee/nat:full-Name/nat:first-name/$
nat:employee/nat:full-Name/nat:first-name//
nat:employee/nat:full-Name/nat:name
nat:employee/nat:full-Name/nat:name/$
nat:employee/nat:full-Name/nat:name//
nat:employee/nat:full-Name//
nat:employee//
```