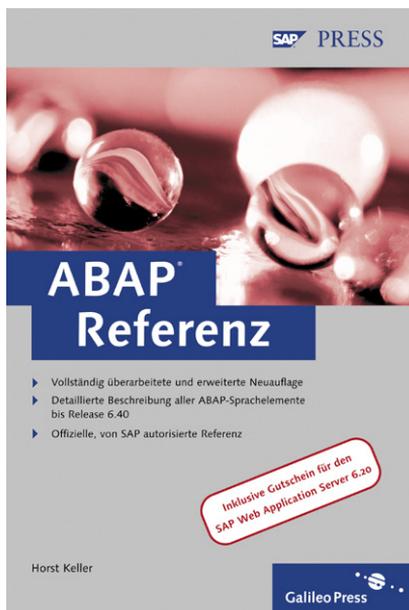


Horst Keller

ABAP-Referenz



Galileo Press

Inhaltsübersicht

	Vorwort zur zweiten Auflage	29
	Vorwort zur ersten Auflage	31
1	Einführung und Übersicht	35
	Teil 1 Syntax	49
2	ABAP-Syntax	51
	Teil 2 Programmaufbau	69
3	Programmeinleitende Anweisungen	71
4	Modularisierungsanweisungen	81
5	Eingebaute Typen, Datenobjekte und Funktionen	115
	Teil 3 Deklarative Anweisungen	135
6	Deklarative Anweisungen für Datentypen und Datenobjekte	137
7	Definition von Klassen und Interfaces	179
8	Typisierung	227
	Teil 4 Objekte erzeugen	235
9	Datenobjekte und Objekte erzeugen	237
	Teil 5 Programmeinheiten aufrufen und verlassen	255
10	ABAP-Programme aufrufen	257
11	Verarbeitungsblöcke aufrufen	287
12	Programmeinheiten verlassen	329
	Teil 6 Programmablaufsteuerung	339
13	Logische Ausdrücke	341
14	Kontrollstrukturen	361
15	Ausnahmebehandlung	371
	Teil 7 Zuweisungen	391
16	Wertzuweisungen	393
17	Referenzen setzen	403
18	Datenobjekte initialisieren	421
	Teil 8 Interne Daten bearbeiten	425
19	Rechenausdrücke	427
20	Rechenanweisungen	437
21	Byte- und Zeichenkettenverarbeitung	441

22	Interne Tabellen bearbeiten	473
23	Extraktdatenbestände bearbeiten	529
24	Eigenschaften von Datenobjekten	539
	Teil 9 Benutzerdialoge	551
25	Dynpros	553
26	Selektionsbilder	605
27	Listen	669
28	Nachrichten	757
	Teil 10 Externe Daten bearbeiten	771
29	Open SQL	773
30	Native SQL	839
31	Daten-Cluster	851
32	Die ABAP-Dateischnittstelle	879
33	Datenkonsistenz	919
	Teil 11 Programmparameter	933
34	Parameter im SAP Memory	935
35	Sprachumgebung	941
36	Datums- und Zeitinformationen	955
	Teil 12 Programmbearbeitung	967
37	Programme testen und prüfen	969
38	Dynamische Programmentwicklung	981
	Teil 13 Externe Programmierschnittstellen	1003
39	Remote Function Call	1005
40	ABAP und XML	1023
	Teil 14 Obsolete Anweisungen	1049
41	OLE-Schnittstelle	1051
42	Obsolete Anweisungen	1061
	Teil 15 Anhang	1133
A	Konvertierungsregeln für Zuweisungen	1135
B	Sprachnahe Klassen und Interfaces	1159
C	Sprachnahe Funktionsbausteine	1167
D	Vordefinierte behandelbare Ausnahmen	1173
E	Glossar	1183
F	Hinweise zu den CD-ROMs	1235
	Index	1237

Inhalt

Vorwort zur zweiten Auflage	29
Vorwort zur ersten Auflage	31
1 Einführung und Übersicht	35
1.1 Ziel des Buches	35
1.2 Beschriebene Releases	35
1.3 Von der SAP-Basis zum SAP Web Application Server	36
1.4 ABAP und Unicode	37
1.5 Aufbau des Buches	39
1.6 Suchmöglichkeiten	43
1.7 Syntaxdiagramme	43
1.8 Änderungen gegenüber der ersten Auflage	45
1.9 Weitere Informationen zum Buch	47
Teil 1 Syntax	49
2 ABAP-Syntax	51
2.1 ABAP-Anweisungen	51
2.2 ABAP-Sprachelemente	51
2.2.1 Operatoren	52
2.2.2 Operanden	53
2.2.3 Bezeichner für Operanden	53
2.2.4 Operanden statisch oder dynamisch angeben	59
2.2.5 Datenobjekte in Operandenpositionen	60
2.3 Namenskonventionen	64
2.4 Kettensätze	65
2.5 Kommentare	66

3 Programmeinleitende Anweisungen 71

3.1	Übersicht	71
3.2	Ausführbare Programme	72
3.2.1	Zusätze für die Grundliste des Programms	73
3.2.2	Zusatz für die Nachrichtenklasse	75
3.2.3	Zusatz für die Definition einer logischen Datenbank	75
3.3	Modul-Pools und Subroutinen-Pools	76
3.4	Funktionsgruppen	76
3.5	Class-Pools	77
3.6	Interface-Pools	78
3.7	Typgruppen	78

4 Modularisierungsanweisungen 81

4.1	Übersicht	81
4.2	Prozeduren	82
4.2.1	Methoden	82
4.2.2	Funktionsbausteine	83
4.2.3	Unterprogramme	85
4.3	Dialogmodule	90
4.4	Ereignisblöcke	91
4.4.1	Programmkonstruktor	91
4.4.2	Reporting-Ereignisse	92
4.4.3	Selektionsbildereignisse	98
4.4.4	Listenereignisse	103
4.5	Quelltextmodularisierung	110
4.5.1	Include-Programme	110
4.5.2	Makros	111

5 Eingebaute Typen, Datenobjekte und Funktionen 115

5.1	Übersicht	115
5.2	Eingebaute Datentypen	115
5.2.1	Eingebaute ABAP-Typen	115
5.2.2	Generische ABAP-Typen	119
5.2.3	Eingebaute Typen im ABAP Dictionary	120
5.2.4	Der eingebaute Datentyp cursor	124

5.3	Eingebaute Datenobjekte	124
5.3.1	Die Konstante space	124
5.3.2	Die Selbstreferenz me	124
5.3.3	Systemfelder	125
5.3.4	Die Struktur screen	130
5.4	Eingebaute Funktionen	131
5.4.1	Mathematische Funktionen	131
5.4.2	Beschreibungsfunktionen	133

Teil 3 Deklarative Anweisungen 135

6 Deklarative Anweisungen für Datentypen und Datenobjekte 137

6.1	Übersicht	137
6.1.1	Gültigkeit und Sichtbarkeit	137
6.1.2	Datentypen	138
6.1.3	Datenobjekte	138
6.1.4	Absolute Typnamen	141
6.1.5	Anweisungen zur Deklaration von Datentypen und -objekten ..	143
6.2	Typgruppen einbinden	143
6.3	Datentypen definieren	144
6.3.1	Typdefinitionen mit eingebauten ABAP-Typen	145
6.3.2	Typdefinition durch Bezug auf vorhandene Typen	146
6.3.3	Definition von Referenztypen	147
6.3.4	Definition von strukturierten Typen	149
6.3.5	Definition von Tabellentypen	150
6.3.6	Definition eines Ranges-Tabellentyps	154
6.4	Variablen deklarieren	155
6.4.1	Zusätze für Datenobjekte	156
6.4.2	Elementare Datenobjekte eingebauter ABAP-Typen	158
6.4.3	Datenobjekte bereits vorhandener Typen	158
6.4.4	Deklaration von Referenzvariablen	159
6.4.5	Deklaration von Strukturen	160
6.4.6	Deklaration interner Tabellen	161
6.4.7	Definition einer Ranges-Tabelle	163
6.5	Statische Attribute von Klassen deklarieren	164
6.6	Konstante Datenobjekte deklarieren	165
6.7	Statische Datenobjekte in Prozeduren deklarieren	167
6.8	Strukturkomponenten übernehmen	168
6.9	Tabellarbeitsbereiche deklarieren	170
6.9.1	Flacher Tabellenarbeitsbereich	170
6.9.2	Beliebiger Tabellenarbeitsbereich	171

6.10	Feldsymbole deklarieren	174
6.10.1	Feldsymbol typisieren	175
6.10.2	Aufprägung einer Struktur	175
6.11	Extraktdatenbestand deklarieren	176

7 Definition von Klassen und Interfaces 179

7.1	Übersicht	179
7.2	Definition von Klassen	180
7.2.1	Deklarationsteil	180
7.2.2	Implementierungsteil	190
7.2.3	Klassen bekannt machen	192
7.3	Definition von Interfaces	193
7.3.1	Deklarationsteil	193
7.3.2	Interfaces bekannt machen	195
7.4	Deklaration von Komponenten in Klassen und Interfaces	195
7.4.1	Methoden	196
7.4.2	Ereignisse	217
7.4.3	Interfaces implementieren bzw. einbinden	220

8 Typisierung 227

8.1	Übersicht	227
8.2	Syntax der Typisierung	227
8.2.1	Generische Typisierung	228
8.2.2	Vollständige Typisierung	229
8.3	Typisierung überprüfen	230
8.3.1	Allgemeine Regeln	230
8.3.2	Literale als Aktualparameter	231

Teil 4 Objekte erzeugen 235

9 Datenobjekte und Objekte erzeugen 237

9.1	Übersicht	237
9.2	Datenobjekte erzeugen	237
9.2.1	Datentyp implizit festlegen	238
9.2.2	Datentyp über eingebaute ABAP-Typen festlegen	239
9.2.3	Datentyp über vorhandenen Typ festlegen	241
9.2.4	Referenzvariablen erzeugen	243
9.2.5	Interne Tabellen erzeugen	244
9.2.6	Datentyp über Typobjekte festlegen	245
9.2.7	Behandelbare Ausnahmen beim Erzeugen von Datenobjekten ..	247

9.3	Objekte in ABAP Objects erzeugen	247
9.3.1	Klasse implizit festlegen	249
9.3.2	Klasse explizit festlegen	249
9.3.3	Statische Parameterübergabe	250
9.3.4	Dynamische Parameterübergabe	251
9.3.5	Objekt im Shared Memory anlegen	252
9.3.6	Behandelbare Ausnahmen beim Erzeugen von Objekten	254

Teil 5 Programmeinheiten aufrufen und verlassen **255**

10 ABAP-Programme aufrufen **257**

10.1	Übersicht	257
10.2	Ausführbare Programme aufrufen	258
10.2.1	Ablauf des Programmaufrufs	259
10.2.2	Zusätze für das Selektionsbild	261
10.2.3	Zusätze für die Grundliste	270
10.2.4	Zusätze für die Hintergrundverarbeitung	274
10.3	Transaktionen aufrufen	277
10.3.1	Aufruf einer Transaktion und Rückkehr zum Aufrufer	277
10.3.2	Aufruf einer Transaktion ohne Rückkehr zum Aufrufer	285

11 Verarbeitungsblöcke aufrufen **287**

11.1	Übersicht	287
11.2	Prozeduren aufrufen	288
11.2.1	Übersicht	288
11.2.2	Methodenaufruf	291
11.2.3	Funktionsbaustein aufruf	303
11.2.4	Unterprogrammaufruf	313
11.3	Ereignisbehandler aufrufen	319
11.3.1	Ereignisse auslösen	319
11.3.2	Ereignisbehandler registrieren	321
11.4	Ereignisblöcke aufrufen	325
11.4.1	Ereignisse einer logischen Datenbank auslösen	325
11.4.2	Listenereignisse auslösen	327

12 Programmeinheiten verlassen **329**

12.1	Übersicht	329
12.2	Programme verlassen	329

12.3	Verarbeitungsblöcke verlassen	330
12.3.1	Schema zum Verlassen von Verarbeitungsblöcken	330
12.3.2	Verarbeitungsblöcke mit RETURN verlassen	331
12.3.3	Verarbeitungsblöcke mit EXIT verlassen	332
12.3.4	Verarbeitungsblöcke mit CHECK verlassen	333
12.3.5	GET-Verarbeitungsblöcke mit REJECT verlassen	334
12.3.6	Verarbeitungsblöcke ausführbarer Programme mit STOP verlassen	335
12.4	Schleifen verlassen	337
12.4.1	Schleifen mit EXIT verlassen	337
12.4.2	Schleifendurchlauf mit CONTINUE verlassen	337
12.4.3	Schleifendurchlauf mit CHECK verlassen	338

Teil 6 Programmablaufsteuerung 339

13 Logische Ausdrücke 341

13.1	Übersicht	341
13.2	Logische Ausdrücke mit Vergleichsoperatoren	341
13.2.1	Vergleichsoperatoren für alle Datentypen	342
13.2.2	Vergleichsoperatoren für zeichenartige Operanden	346
13.2.3	Vergleichsoperatoren für byteartige Datentypen	348
13.2.4	Vergleichsoperatoren für Bitmuster	350
13.3	Intervallzugehörigkeit feststellen	351
13.4	Zustände überprüfen	351
13.4.1	Zuweisung an ein Feldsymbol überprüfen	351
13.4.2	Gültigkeit einer Referenz überprüfen	352
13.4.3	Operand auf Initialwert überprüfen	353
13.4.4	Ausgabeparameter auf Aktualparameter überprüfen	353
13.4.5	Formalparameter auf Aktualparameter überprüfen	354
13.5	Selektionstabelle auswerten	355
13.6	Boolesche Operatoren und Klammerung	358
13.6.1	Verknüpfung logischer Ausdrücke mit AND	358
13.6.2	Verknüpfung logischer Ausdrücke mit OR	358
13.6.3	Negation eines logischen Ausdrucks mit NOT	359
13.6.4	Klammerung logischer Ausdrücke	359

14 Kontrollstrukturen 361

14.1	Übersicht	361
14.2	Verzweigungen	362
14.2.1	Verzweigung mit IF	362
14.2.2	Verzweigung mit CASE	363

14.3	Schleifen	364
14.3.1	Unbedingte Schleifen mit DO	364
14.3.2	Bedingte Schleifen mit WHILE	368
14.4	Programmunterbrechung	369

15 Ausnahmebehandlung 371

15.1	Übersicht	371
15.1.1	Behandelbare Ausnahmen vor Release 6.10	371
15.1.2	Behandelbare Ausnahmen seit Release 6.10	372
15.1.3	Systemverhalten nach einer klassenbasierten Ausnahme	378
15.1.4	Zusammenspiel von nicht-klassenbasierten und klassenbasierten Ausnahmen	379
15.1.5	Anweisungen für Ausnahmen	380
15.2	Klassenbasierte Ausnahmen	380
15.2.1	Klassenbasierte Ausnahmen auslösen	380
15.2.2	Klassenbasierte Ausnahmen behandeln	381
15.3	Abfangbare Laufzeitfehler	385
15.3.1	Definition und Auslösen abfangbarer Laufzeitfehler	385
15.3.2	Abfangbare Laufzeitfehler behandeln	385
15.4	Nicht-klassenbasierte Ausnahmen	388
15.4.1	Nicht-klassenbasierte Ausnahmen definieren	388
15.4.2	Nicht-klassenbasierte Ausnahmen auslösen	388
15.4.3	Nicht-klassenbasierte Ausnahmen behandeln	389

Teil 7 Zuweisungen 391

16 Wertzuweisungen 393

16.1	Übersicht	393
16.2	Zuweisung von Datenobjekten	394
16.2.1	Zuweisung mit Schlüsselwort	394
16.2.2	Zuweisung mit Zuweisungsoperator	394
16.2.3	Mehrfachzuweisungen	395
16.3	Zuweisung von Strukturkomponenten	396
16.4	Formatierte Zuweisung	398
16.5	Konvertierung einer gepackten Zahl	400

17 Referenzen setzen 403

17.1	Übersicht	403
17.2	Datenobjekte Feldsymbolen zuweisen	403
17.2.1	Angabe des Speicherbereichs	405

17.2.2	Angabe des Datentyps	411
17.2.3	Angabe der Bereichsgrenzen	414
17.3	Feldsymbol initialisieren	418
17.4	Datenreferenz besorgen	418

18 Datenobjekte initialisieren 421

18.1	Übersicht	421
18.2	Beliebige Datenobjekte initialisieren	421
18.2.1	Mit Initialwert initialisieren	421
18.2.2	Mit anderen Inhalten initialisieren	422
18.3	Interne Tabelle initialisieren	423
18.4	Speicher freigeben	424

Teil 8 Interne Daten bearbeiten 425

19 Rechenausdrücke 427

19.1	Übersicht	427
19.2	Die Anweisung COMPUTE	427
19.3	Arithmetische Ausdrücke	428
19.3.1	Arithmetische Operatoren	429
19.3.2	Klammerung	430
19.3.3	Priorität von funktionalen Methoden	430
19.3.4	Rechentyp	430
19.3.5	Behandelbare Ausnahmen in arithmetischen Ausdrücken	432
19.4	Bitausdrücke	433
19.4.1	Bitoperatoren	434
19.4.2	Klammerung	434

20 Rechenanweisungen 437

20.1	Übersicht	437
20.2	Addition	437
20.3	Subtraktion	438
20.4	Multiplikation	438
20.5	Division	439

21 Byte- und Zeichenkettenverarbeitung 441

21.1	Übersicht	441
21.1.1	Byte- und Zeichenketten	441
21.1.2	Anweisungen zur Byte- und Zeichenkettenverarbeitung	441
21.1.3	Operanden in der Byte- und Zeichenkettenverarbeitung	442
21.1.4	Behandlung schließender Leerzeichen in der Zeichenkettenverarbeitung	444
21.2	Byte- oder Zeichenketten verketten	444
21.3	Byte- oder Zeichenketten durchsuchen	446
21.3.1	Durchsuchen mit FIND	446
21.3.2	Durchsuchen mit SEARCH	448
21.4	Byte- oder Zeichenketten ersetzen	452
21.4.1	Positionsbasiertes Ersetzen	454
21.4.2	Musterbasiertes Ersetzen	455
21.5	Byte- oder Zeichenketten verschieben	457
21.5.1	Anzahl der Stellen	458
21.5.2	Richtung der Verschiebung	460
21.5.3	Verschieben bestimmter Zeichen aus dem Feld	460
21.6	Byte- oder Zeichenketten zerlegen	461
21.7	Zeichenketten verdichten	463
21.8	Zeichenketten konvertieren	464
21.9	Zeichenketten überlagern	466
21.10	Zeichenketten umwandeln	467
21.10.1	Umwandlung der Groß-/Kleinschreibung	467
21.10.2	Umwandlung nach einem Muster	468
21.11	Bits setzen und lesen	469
21.11.1	Einzelbits in Byteketten setzen	469
21.11.2	Einzelbits aus Byteketten lesen	470

22 Interne Tabellen bearbeiten 473

22.1	Übersicht	473
22.2	Interne Tabellen auslesen	474
22.2.1	Einzelne Zeilen lesen	474
22.2.2	Schleifenverarbeitung interner Tabellen	485
22.2.3	Gruppenstufenverarbeitung	489
22.2.4	Summierung in Gruppenstufen	491
22.3	Interne Tabellen füllen	493
22.3.1	Zeilen einfügen	493
22.3.2	Zeilen verdichtet einfügen	498
22.3.3	Zeilen anhängen	500

22.4	Tabellenzeilen bearbeiten	504
22.4.1	Zeilen ändern	504
22.4.2	Zeilen löschen	509
22.4.3	Sortieren	515
22.5	Angabe von Komponenten	519
22.6	Interne Tabellen durchsuchen	520
22.6.1	Suchoptionen	521
22.7	Verarbeitung spezieller interner Tabellen	522

23 Extraktdatenbestände bearbeiten 529

23.1	Einführung	529
23.2	Zeilenstruktur festlegen	529
23.3	Extraktdatenbestand füllen	531
23.4	Extraktdatenbestand sortieren	532
23.5	Extraktdatenbestand auslesen	534
23.6	Gruppenstufenverarbeitung	535

24 Eigenschaften von Datenobjekten 539

24.1	Übersicht	539
24.2	Elementare Eigenschaften beliebiger Datenobjekte	539
24.2.1	Datentyp und Anzahl der Komponenten	540
24.2.2	Belegte Länge des Datenobjekts	542
24.2.3	Nachkommastellen	542
24.2.4	Ausgabelänge	543
24.2.5	Elementarer Datentyp im ABAP Dictionary	544
24.2.6	Konvertierungsroutine	545
24.3	Eigenschaften interner Tabellen	546
24.3.1	Tabellenart	546
24.3.2	Anzahl der Zeilen	547
24.3.3	Initialer Speicherbedarf	547
24.4	Abstände von Datenobjekten	548

Teil 9 Benutzerdialoge 551

25 Dynpros 553

25.1	Übersicht	553
25.1.1	Benutzungsoberfläche	553
25.1.2	Bildschirmbild	555

25.1.3	Dynpro-Felder	555
25.1.4	Dynpro-Ablauf und Dynpro-Folgen	556
25.2	Anweisungen der Dynpro-Ablauflogik	557
25.2.1	Ereignisblöcke der Dynpro-Ablauflogik	558
25.2.2	Aufruf von Dialogmodulen	560
25.2.3	Steuerung von Datentransport und Ablauflogik	564
25.2.4	Verarbeitungsketten	571
25.2.5	Verarbeitung von Table Controls	572
25.2.6	Einbinden von Subscreens	576
25.3	ABAP-Anweisungen für Dynpros	580
25.3.1	Dynpro-Folge aufrufen	581
25.3.2	GUI-Status setzen	582
25.3.3	GUI-Status feststellen	584
25.3.4	GUI-Titel setzen	585
25.3.5	Anzeige unterdrücken	587
25.3.6	Eigenschaften von Bildelementen feststellen	588
25.3.7	Eigenschaften von Bildelementen modifizieren	590
25.3.8	Cursor im Bildschirmbild setzen	592
25.3.9	Cursor-Position auswerten	594
25.3.10	Control deklarieren	595
25.3.11	Table Control initialisieren	601
25.3.12	Steploop-Verarbeitung verlassen	601
25.3.13	Eingabedaten bewahren	602
25.3.14	Folge-Dynpro setzen	603
25.3.15	Dynpro verlassen	604

26 Selektionsbilder 605

26.1	Übersicht	605
26.1.1	Selektionsbilder als Dynpros	605
26.1.2	Aufgaben von Selektionsbildern	605
26.1.3	GUI-Status von Selektionsbildern	605
26.1.4	Selektionsbildereignisse	606
26.1.5	Selektionsbilder und logische Datenbanken	606
26.1.6	Anweisungen für Selektionsbilder	606
26.2	Selektionsbilder anlegen und gestalten	606
26.2.1	Selektionsbilder anlegen	607
26.2.2	Selektionsbilder gestalten	611
26.2.3	Elemente anderer Selektionsbilder übernehmen	626
26.2.4	Varianten und Zusätze für Selektionsbilder logischer Datenbanken	630
26.3	Parameter definieren	635
26.3.1	Datentyp des Parameters	636
26.3.2	Eigenschaften der Bildelemente	639
26.3.3	Eigenschaften des Wertes und der Wertübergabe	645
26.3.4	Zusätze für Selektionsbilder logischer Datenbanken	648
26.4	Selektionskriterien definieren	651
26.4.1	Datentyp der Spalten low und high der Selektionstabelle	655

26.4.2	Eigenschaften der Bildelemente	657
26.4.3	Eigenschaften des Wertes und der Wertübergabe	660
26.4.4	Zusätze für Selektionsbilder logischer Datenbanken	662
26.5	Selektionsbilder aufrufen	664
26.5.1	Aufruf über SUBMIT	664
26.5.2	Aufruf über Reporttransaktion	664
26.5.3	Aufruf über Dialogtransaktion	664
26.5.4	Aufruf im Programm	665
26.5.5	Selektionsbildverarbeitung	666

27 Listen 669

27.1	Übersicht	669
27.1.1	Listen als Bildschirmbilder	669
27.1.2	Listen im ABAP-Programm	669
27.1.3	Grundliste	669
27.1.4	Verzweigungslisten	670
27.1.5	Aufbau einer Liste	670
27.1.6	Drucklisten	671
27.1.7	Listen und ABAP Objects	671
27.1.8	Listen und Unicode	672
27.1.9	Anweisungen zur Listenverarbeitung	672
27.2	Listen erstellen	674
27.2.1	Daten in Listen schreiben	674
27.2.2	Horizontale Linien erzeugen	701
27.2.3	Listen abschnittsweise formatieren	702
27.2.4	Anzeige von Leerzeilen	709
27.2.5	Vertikale Positionierung des Listen-Cursors	710
27.2.6	Horizontale Positionierung des Listen-Cursors	714
27.2.7	Fixbereich beim horizontalen Blättern	716
27.2.8	Seitenumbruch und Drucklistenstellung	717
27.2.9	Bedingter Seitenumbruch	725
27.2.10	Variablen mit Listenzeilen speichern	726
27.2.11	Seitenrand von Drucklisten	727
27.2.12	Steuerung von Drucklisten	728
27.3	Listen im Listenpuffer bearbeiten	732
27.3.1	Listenzeilen lesen	732
27.3.2	Listenzeilen modifizieren	735
27.3.3	Listen blättern	737
27.3.4	Listeneigenschaften auslesen	740
27.4	Angezeigte Liste an Cursor-Position auswerten	743
27.5	Anzeigeeigenschaften von Bildschirmlisten	745
27.5.1	GUI-Status einer Bildschirmliste	745
27.5.2	Titel einer Bildschirmliste	747
27.5.3	Cursor setzen	748
27.5.4	Liste im Dialogfenster	750

27.6	Listenanzeige aufrufen und verlassen	752
27.6.1	Anzeige der Grundliste aufrufen	752
27.6.2	Anzeige von Listen verlassen	753

28 Nachrichten 757

28.1	Übersicht	757
28.1.1	Ablage von Nachrichten	757
28.1.2	Nachrichtentypen	757
28.2	Nachrichten senden	761
28.2.1	Angabe einer Nachricht	762
28.2.2	Angabe eines beliebigen Textes	766
28.2.3	Zusätze zu MESSAGE	767

Teil 10 Externe Daten bearbeiten 771

29 Open SQL 773

29.1	Übersicht	773
29.1.1	Umfang von Open SQL	773
29.1.2	Datenbankschnittstelle	773
29.1.3	Datenbankzugriff	773
29.1.4	Mandantenbehandlung	773
29.1.5	SAP-Pufferung	773
29.1.6	LUW	774
29.1.7	Die Anweisungen von Open SQL	774
29.2	Daten aus Datenbanktabellen lesen	775
29.2.1	Struktur der Ergebnismenge bestimmen	777
29.2.2	Auszulesende Datenbanktabellen angeben	784
29.2.3	Zielbereich angeben	791
29.2.4	Ergebnismenge einschränken	796
29.2.5	Zeilen zusammenfassen	809
29.2.6	Zusammengefasste Zeilen einschränken	810
29.2.7	Zeilen der Ergebnismenge sortieren	811
29.3	Daten aus Datenbanktabellen über Cursor lesen	814
29.3.1	Cursor öffnen	814
29.3.2	Daten über Cursor lesen	815
29.3.3	Cursor schließen	816
29.4	Daten in Datenbanktabellen einfügen	818
29.4.1	Angabe der Datenbanktabelle	819
29.4.2	Angabe der Quelle	820
29.5	Daten in Datenbanktabellen ändern	822
29.5.1	Angabe der Datenbanktabelle	823
29.5.2	Angabe der Änderungen	824

29.6	Daten in Datenbanktabellen einfügen oder ändern	829
29.6.1	Angabe der Datenbanktabelle	830
29.6.2	Angabe der Quelle	830
29.7	Daten in Datenbanktabellen löschen	832
29.7.1	Angabe der Datenbanktabelle	833
29.7.2	Angabe der Zeilen	833
29.8	Arbeitsbereiche in Open-SQL-Anweisungen	836
29.9	Behandelbare Ausnahmen in Open-SQL-Anweisungen	837

30 Native SQL 839

30.1	Übersicht	839
30.2	Native SQL einbinden	839
30.2.1	Hostvariablen	841
30.2.2	Cursor-Verarbeitung	842
30.2.3	Datenbankprozeduren aufrufen	844
30.2.4	Datenbankverbindung festlegen	845
30.2.5	Implizite Cursor-Verarbeitung	848
30.3	Native SQL verlassen	850
30.4	Behandelbare Ausnahmen in Native SQL	850

31 Daten-Cluster 851

31.1	Übersicht	851
31.2	Daten-Cluster erstellen	851
31.2.1	Definition des Daten-Clusters	852
31.2.2	Bestimmung der Ablage	853
31.2.3	Komprimierung steuern	859
31.2.4	Behandelbare Ausnahmen beim Export von Daten-Clustern	859
31.3	Daten-Cluster lesen	859
31.3.1	Angabe der einzulesenden Datenobjekte	860
31.3.2	Bestimmung der Ablage	862
31.3.3	Konvertierungszusätze	865
31.3.4	Textsprachenregel	872
31.3.5	Behandelbare Ausnahmen beim Importieren aus Daten-Clustern	874
31.4	Inhaltsverzeichnis eines Daten-Clusters lesen	874
31.5	Löschen eines Daten-Clusters	876
31.6	Löschen eines Daten-Clusters im ABAP Memory	877

32 Die ABAP-Dateischnittstelle 879

32.1	Übersicht	879
32.1.1	Adressierung von Dateien	879

32.1.2	Berechtigungen für Dateizugriffe	879
32.1.3	Sperren	882
32.1.4	Die Dateischnittstelle in Unicode-Programmen	882
32.1.5	Dateigröße	883
32.1.6	Die Anweisungen der Dateischnittstelle	883
32.2	Datei öffnen	884
32.2.1	Definition der Zugriffsart	885
32.2.2	Definition der Ablageart	886
32.2.3	Positionsangabe	891
32.2.4	Betriebssystemabhängige Zusätze	892
32.2.5	Fehlerbehandlung	894
32.2.6	Behandelbare Ausnahmen beim Öffnen von Dateien	896
32.3	Datei schreiben	896
32.3.1	Einfluss der Zugriffsart	897
32.3.2	Einfluss der Ablageart	898
32.3.3	Anzahl der übertragenen Zeichen bzw. Bytes einschränken	899
32.3.4	Anhängen einer Zeilenende-Markierung verhindern	900
32.3.5	Behandelbare Ausnahmen beim Schreiben in Dateien	900
32.4	Datei lesen	901
32.4.1	Einfluss der Zugriffsart	902
32.4.2	Einfluss der Ablageart	902
32.4.3	Anzahl der eingelesenen Zeichen bzw. Bytes einschränken	903
32.4.4	Anzahl der eingelesenen Zeichen bzw. Bytes feststellen	905
32.4.5	Behandelbare Ausnahmen beim Lesen von Dateien	905
32.5	Eigenschaften einer geöffneten Datei bestimmen	906
32.5.1	Position des Dateizeigers feststellen	906
32.5.2	Weitere Eigenschaften feststellen	907
32.5.3	Behandelbare Ausnahmen beim Feststellen von Dateieigenschaften	909
32.6	Dateieigenschaften einer geöffneten Datei ändern	909
32.6.1	Position des Dateizeigers festlegen	910
32.6.2	Weitere Eigenschaften ändern	912
32.6.3	Behandelbare Ausnahmen beim Ändern von Dateieigenschaften	914
32.7	Größe einer Datei ändern	914
32.8	Datei schließen	915
32.9	Datei löschen	916

33 Datenkonsistenz 919

33.1	Übersicht	919
33.2	Datenbank-LUW	920
33.2.1	Datenbank-Commit	920
33.2.2	Datenbank-Rollback	921
33.3	SAP-LUW	922
33.3.1	SAP-Commit	923

33.3.2	SAP-Rollback	925
33.3.3	Lokale Verbuchung	926
33.4	Datenbanksperren	926
33.5	SAP-Sperren	927
33.5.1	SAP-Sperren verhängen	927
33.5.2	SAP-Sperren aufheben	928
33.6	Berechtigungsprüfung	929

Teil 11 Programmparameter 933

34 Parameter im SAP Memory 935

34.1	Übersicht	935
34.1.1	SPA/GPA-Parameter	935
34.1.2	SPA/GPA-Parameter verwalten	935
34.1.3	SPA/GPA-Parameter und Dynpro-Felder	936
34.1.4	Anweisungen für SPA/GPA-Parameter	936
34.2	Parameter setzen	936
34.3	Parameter lesen	938

35 Sprachumgebung 941

35.1	Übersicht	941
35.1.1	Text-Pools	941
35.1.2	Textumgebung	941
35.1.3	Länderkennung für Listenaufbereitung	943
35.1.4	Anweisungen für die Sprachumgebung	944
35.2	Text-Pool einer Sprache laden	944
35.3	Textumgebung setzen	945
35.4	Textumgebung feststellen	950
35.5	Länderkennung für Listenaufbereitung setzen	951

36 Datums- und Zeitinformationen 955

36.1	Übersicht	955
36.1.1	Systemfelder für Datum und Zeit	955
36.1.2	Zeitstempel	956
36.1.3	Anweisungen zu Datum und Zeit	959
36.2	Systemfelder für Datum und Zeit aktualisieren	960
36.3	Aktuellen Zeitstempel erstellen	960
36.4	Zeitstempel in lokale Zeit konvertieren	961
36.5	Lokale Zeit in Zeitstempel konvertieren	963

Teil 12 Programmbearbeitung 967

37 Programme testen und prüfen 969

37.1	Übersicht	969
37.2	Checkpoints	969
37.2.1	Assertion definieren	969
37.2.2	Breakpoint definieren	972
37.3	Laufzeitmessung	974
37.3.1	Relative Programmlaufzeit	974
37.3.2	Zeitauflösung festlegen	976
37.4	Messstrecke für Laufzeitanalyse	977
37.5	Erweiterte Programmprüfung ausschalten	978

38 Dynamische Programmentwicklung 981

38.1	Übersicht	981
38.2	Dynamischer Subroutinen-Pool	981
38.2.1	Zusätze zur Fehlerbehandlung	983
38.3	Einlesen eines ABAP-Programms	988
38.4	Syntaxprüfung durchführen	989
38.4.1	Eigenschaften der Syntaxprüfung festlegen	990
38.4.2	Zusätze zur Fehlerbehandlung	991
38.5	ABAP-Programm anlegen oder überschreiben	992
38.5.1	Programmeigenschaften festlegen	994
38.6	Einlesen eines Text-Pools	998
38.7	Text-Pool anlegen oder überschreiben	999
38.8	ABAP Editor aufrufen	1002

Teil 13 Externe Programmierschnittstellen 1003

39 Remote Function Call 1005

39.1	Übersicht	1005
39.1.1	Die RFC-Schnittstelle	1005
39.1.2	RFC-Destination	1006
39.1.3	Kontext eines RFC	1008
39.1.4	Ausnahmen beim RFC	1008
39.1.5	Systemfelder beim RFC	1009
39.1.6	RFC-Berechtigung	1009

39.1.7	Einschränkungen beim RFC	1010
39.1.8	Anweisungen der RFC-Schnittstelle	1011
39.2	Remote-Funktionsaufruf	1011
39.2.1	Synchroner Remote Function Call	1011
39.2.2	Asynchroner Remote Function Call	1012
39.2.3	Transaktionaler Remote Function Call	1018
39.2.4	Beispiel für Remote Function Call	1020

40 ABAP und XML 1023

40.1	XSL-Transformationen	1023
40.1.1	XSL-Transformationen im Repository	1023
40.2	Kanonische XML-Repräsentation	1024
40.2.1	Generelles asXML-Format	1024
40.2.2	asXML-Format für benannte Datenobjekte außer Referenzvariablen	1026
40.2.3	asXML-Format für Referenzvariablen und referenzierte Objekte	1027
40.3	Simple Transformations	1034
40.4	Aufruf einer XSL- oder ST-Transformation	1035
40.4.1	Angabe der Transformation	1036
40.4.2	Transformationsquelle	1036
40.4.3	Transformationsergebnis	1038
40.4.4	Parameter für eine XSL-Transformation	1040
40.4.5	Externe Objekte an eine XSL-Transformation übergeben	1041
40.4.6	Steuerung der Transformation	1042
40.5	Beispiel für eine XSL-Transformation	1043
40.6	Beispiel für eine Simple Transformation	1046

Teil 14 Obsolete Anweisungen 1049

41 OLE-Schnittstelle 1051

41.1	Automation-Objekt erzeugen	1052
41.2	Automation-Methode aufrufen	1053
41.3	Eigenschaften eines Automation-Objekts auslesen	1056
41.4	Eigenschaften eines Automation-Objekts setzen	1057
41.5	Automation-Objekt freigeben	1058
41.6	Obsoleter Aufruf eines Texteditors	1059

42.1	Übersicht	1061
42.2	Obsolete Syntax	1061
42.3	Obsolete Modularisierung	1062
42.3.1	Obsolete Ereignisblock	1062
42.4	Obsolete Deklarationen	1062
42.4.1	Obsolete Schnittstellen-Arbeitsbereiche	1062
42.4.2	Obsolete Deklarationen interner Standardtabellen	1065
42.4.3	Obsolete Deklarationen spezieller interner Tabellen	1067
42.4.4	Obsolete Hinweis für die erweiterte Programm-prüfung	1069
42.5	Obsolete Objekterzeugung	1070
42.6	Obsolete Programmaufruf	1072
42.7	Obsoletes Verlassen eines Programms	1075
42.8	Obsolete Programmablaufsteuerung	1075
42.8.1	Obsolete Vergleichsoperatoren	1075
42.8.2	Obsolete Verzweigung	1076
42.9	Obsolete Zuweisungen	1077
42.9.1	Obsolete Zuweisung eines prozentualen Teilfelds	1077
42.9.2	Obsolete Konvertierung	1078
42.9.3	Obsoletes Zwischenspeichern von Datenobjekten	1079
42.10	Obsolete Rechenanweisungen	1080
42.10.1	Addition von Feldfolgen im Speicher	1080
42.10.2	Komponentenweise addieren	1082
42.10.3	Komponentenweise subtrahieren	1084
42.10.4	Komponentenweise multiplizieren	1084
42.10.5	Komponentenweise dividieren	1085
42.10.6	Obsolete Berechnungen während der Listenerstellung	1086
42.11	Obsolete Zeichenkettenverarbeitung	1089
42.11.1	Obsolete Umsetzung	1089
42.11.2	Obsoletes Ersetzen	1091
42.11.3	Neunerkomplement eines Datums	1093
42.12	Obsolete Verarbeitung interner Tabellen	1094
42.12.1	Obsolete Schlüsselangaben beim Lesen von Zeilen	1094
42.12.2	Obsolete Zuweisung an Tabellenzeilen	1097
42.12.3	Obsolete Form der Anweisung PROVIDE	1099
42.13	Contexte	1101
42.13.1	Übersicht	1101
42.13.2	Instanzen von Contexten erzeugen	1102
42.13.3	Contexte mit Schlüsselwerten versorgen	1103
42.13.4	Contexte abfragen	1104
42.14	Obsolete Anweisungen der Dynpro-Ablauflogik	1105
42.14.1	Werteüberprüfung in der Ablauflogik	1105
42.14.2	Verarbeitung von Steploops	1108

42.15	Obsolete Anweisungen der Listenverarbeitung	1114
42.15.1	Obsolete Formatierungsanweisungen	1114
42.15.2	Obsolete Druckparameter	1115
42.15.3	Obsolete Erzeugung eines Spool-Auftrags	1117
42.16	Obsolete Datenbankzugriffe	1117
42.16.1	Obsoletes Lesen einer Zeile	1117
42.16.2	Obsoletes sequenzielles Lesen mehrerer Zeilen	1120
42.16.3	Obsoletes Lesen mehrerer Zeilen in eine interne Tabelle	1121
42.16.4	Obsolete Kurzformen in Open SQL	1123
42.17	Obsolete externe Programmierschnittstelle COI-C	1124
42.17.1	Verbindungsschritte	1126
42.17.2	Weitere Zusätze	1128

Teil 15 Anhang 1133

A Konvertierungsregeln für Zuweisungen 1135

A.1	Übersicht	1135
A.2	Konvertierungsregeln für elementare Datentypen	1135
A.2.1	Darstellung von numerischen Werten in zeichenartigen Feldern	1136
A.2.2	Konvertierungstabelle für Quellfeld Typ c	1137
A.2.3	Konvertierungstabelle für Quellfeld Typ d	1138
A.2.4	Konvertierungstabelle für Quellfeld Typ f	1140
A.2.5	Konvertierungstabelle für Quellfeld Typ i, b oder s	1141
A.2.6	Konvertierungstabelle für Quellfeld Typ n	1142
A.2.7	Konvertierungstabelle für Quellfeld Typ p	1143
A.2.8	Konvertierungstabelle für Quellfeld Typ string	1144
A.2.9	Konvertierungstabelle für Quellfeld Typ t	1145
A.2.10	Konvertierungstabelle für Quellfeld Typ x	1146
A.2.11	Konvertierungstabelle für Quellfeld Typ xstring	1147
A.3	Konvertierungsregeln für Strukturen	1148
A.3.1	Konvertierung zwischen flachen Strukturen	1148
A.3.2	Konvertierung zwischen flachen Strukturen und Einzelfeldern ..	1152
A.4	Konvertierungsregeln für interne Tabellen	1153
A.5	Zuweisungen zwischen Referenzvariablen	1154
A.5.1	Statischer und dynamischer Typ	1154
A.5.2	Up Cast und Down Cast	1155
A.5.3	Zuweisungen zwischen Datenreferenzvariablen	1156
A.5.4	Zuweisungen zwischen Objektreferenzvariablen	1157

B Sprachnahe Klassen und Interfaces 1159

B.1	Hilfsklassen	1159
B.1.1	Klassen der Run Time Type Services (RTTS)	1159

B.1.2	Klassen für Konvertierungen externer Datenformate	1160
B.1.3	Klasse für Extremwerte von Datenobjekten	1161
B.1.4	Klasse für die Eigenschaften von Zeichen	1161
B.1.5	Klassen für mathematische Operationen	1161
B.1.6	Klasse für Zeitstempel	1162
B.1.7	Klassen für Daten-Cluster	1162
B.1.8	Klasse für Transaktionen	1163
B.1.9	Klasse für die Aufbereitung von Listen	1163
B.1.10	Klassen zur Komprimierung von Daten	1163
B.1.11	Klasse für Laufzeitmessungen	1163
B.1.12	Klasse für schwache Referenzen	1164
B.2	Interface für die Serialisierung	1164
B.3	Shared Objects	1165
B.4	Object Services	1165
B.5	JavaScript-Integration	1166

C Sprachnahe Funktionsbausteine 1167

C.1	Funktionsbausteine für Druckparameter	1167
C.1.1	GET_PRINT_PARAMETERS	1167
C.1.2	SET_PRINT_PARAMETERS	1170
C.2	Funktionsbausteine für Dateien auf dem Präsentationsserver	1170
C.3	Funktionsbaustein für den Aufruf logischer Datenbanken	1171

D Vordefinierte behandelbare Ausnahmen 1173

D.1	Übersicht	1173
D.2	Vordefinierte Ausnahmeklassen	1173
D.3	Abfangbare Laufzeitfehler	1176
D.3.1	Ausnahmegruppe für arithmetische Fehler	1176
D.3.2	Ausnahmegruppe für Konvertierungsfehler	1177
D.3.3	Ausnahmegruppe für Fehler bei der Erzeugung von Datenobjekten	1178
D.3.4	Ausnahmegruppe für Fehler bei der Erzeugung von Instanzen von Klassen	1178
D.3.5	Ausnahmegruppe für Fehler beim Zugriff auf Datenobjekte	1178
D.3.6	Ausnahmegruppe für Fehler beim dynamischen Methodenaufruf	1179
D.3.7	Ausnahmegruppe für Fehler beim Dateizugriff	1180
D.3.8	Ausnahmegruppe für Fehler beim Zugriff auf Daten-Cluster	1180
D.3.9	Ausnahmegruppe für Fehler in der Sprachumgebung	1181
D.3.10	Ausnahmegruppe für Fehler beim Remote Function Call	1181
D.3.11	Nicht zugeordnete abfangbare Laufzeitfehler	1181

E	Glossar	1183
F	Hinweise zu den CD-ROMs	1235
	Index	1237

Vorwort zur zweiten Auflage

Knapp zwei Jahre nach dem Erscheinen der ersten Auflage der ABAP-Referenz liegt nun die zweite Auflage vor. In dieser Neuauflage wird zum einen der vollständige Sprachumfang von ABAP zum aktuellen Release 6.40 dargestellt und zum anderen wurde der bisherige Text korrigiert und in großen Teilen vollständig überarbeitet (siehe Abschnitt 1.8).

Als wichtigste Neuerungen zu Release 6.40 seien hier die Evolution der RTTI nach RTTS und die Shared Objects genannt. Die RTTS (Run Time Type Services) enthalten neben der bisherigen RTTI (Run Time Type Information) auch eine RTTC (Run Time Type Creation), die es erlaubt, zur Programmlaufzeit beliebige Datentypen zu erzeugen und zu verwenden. Damit wird endlich der Wunsch vieler ABAP-Entwickler erfüllt, Strukturen dynamisch erzeugen zu können. Das Beispiel in Abschnitt 9.2.6 zeigt, wie es geht. Shared Objects sind Objekte im Shared Memory, auf die alle Programme eines Applikationsservers zugreifen können. Die zugehörigen Sprachelemente und Begriffe sind in diesem Buch beschrieben. Weitere wichtige Neuerungen zu Release 6.40 sind die Einführung von Simple Transformations zur Serialisierung von ABAP-Daten nach XML und umgekehrt und das in die Sprache integrierte Testwerkzeug ABAP Unit.

Der SAP Web Application Server und mit ihm die Sprache ABAP ist seit Release 6.30 Bestandteil von SAP NetWeaver™. SAP NetWeaver ist eine offene Integrations- und Anwendungsplattform, die die vorhergehende mySAP-Technologie ablöst und erweitert. Neben der gewohnten ABAP-Umgebung enthält der SAP Web Application Server seit Release 6.30 auch eine Umgebung für die Anwendungsentwicklung in Java. Da diese aber nicht in direktem Zusammenhang mit den Sprachelementen von ABAP steht, wird in diesem Buch auch nicht auf den Java-Teil des Web Application Servers eingegangen.

Da Herr Joachim Jacobitz, mein Koautor der ersten Auflage, inzwischen andere Aufgaben innerhalb der SAP übernommen hat, bin ich jetzt als einziger Autor für den Inhalt dieses Buches verantwortlich. Trotzdem hat Herr Jacobitz mit großem Engagement alle Kapitel nochmals Korrektur gelesen und mir viele wertvolle Hinweise gegeben, für die ich ihm ganz herzlich danke.

Als alleiniger Autor dieses seitenstarken Werkes ist es mir ganz besonders wichtig, hervorzuheben, dass diese Arbeit ohne die Gruppe »NetWeaver

Development Tools ABAP«, deren hervorragende Arbeit ich hier vorstellen darf, nicht möglich gewesen wäre. Wie schon die Erstellung der ersten Auflage lebt auch die zweite Auflage von der konstruktiven Kritik und dem Korrekturlesen vieler Kollegen. Neben den im Vorwort zur ersten Auflage genannten Mitarbeitern haben sich um die zweite Auflage besonders verdient gemacht: Ulrich Brink, Christian Hansen, Ulrich Koch, Jürgen Lehmann, Mathias Müller, Helmut Prestel, Michael Redford, Christian Stork, Wolf Hagen Thümmel und Christoph Wedler. Allen anderen Kollegen, die hier nicht namentlich genannt sind, die ich aber auch mit Fragen quälte oder von denen der eine oder andere Tipp kam, sei ebenfalls von Herzen gedankt.

Mein ganz besonderer Dank gilt dieses Mal meinen studentischen Hilfskräften, Frau Agnieszka Chelminska, Frau Beata Kouchnir und Herrn Christian Pretzsch, die mir bei der Erstellung und Korrektur des Manuskripts und der Hand in Hand gehenden Überarbeitung der Online-Dokumentation äußerst engagiert und tatkräftig zur Seite standen.

Den beteiligten Mitarbeitern von Galileo Press und meinem Lektor, Herrn Florian Zimniak, danke ich dafür, dass sie die rasche Veröffentlichung einer Überarbeitung dieses mir sehr wichtigen Buches ermöglicht haben, und für die fortwährend nette Zusammenarbeit.

Abschließend danke ich wie immer meiner lieben Frau Ute für ihre Geduld und ihr Verständnis dafür, dass ich auch nach dem Erscheinen der ersten Auflage einfach nicht aufhörte, an diesem Buch zu arbeiten.

Walldorf, im Mai 2004

Horst Keller

Vorwort zur ersten Auflage

Gleich nachdem die Programmiersprache ABAP mit der Einführung von ABAP Objects zu Release 4.6 einen großen Schritt in Richtung zeitgemäße Programmierung vollzogen hatte, stellte sich für die Weiterentwicklung der Sprache bereits die nächste Herausforderung: Um den Anforderungen eines vom Internet bestimmten, internationalen Programmierumfelds gerecht zu werden, musste der SAP Web Application Server als Nachfolger der SAP-Basis und mit ihm die Programmiersprache ABAP unicode-fähig gemacht werden. Diese Anforderung wurde mit Release 6.10 erfüllt.

Mit der Implementierung von Release 6.10 bzw. 6.20 wird jedoch niemand gezwungen, seine Programme anzupassen, solange das SAP-System nicht auf Unicode umgestellt wird. Unabhängig davon, ob ein Programm in einem Unicode- oder einem Nicht-Unicode-System eingesetzt werden soll, bedeutet die Verwendung der im Zusammenhang mit Unicode eingeführten Neuerungen dennoch einen wichtigen Schritt in Richtung sicherer und fehlerfreier Programmierung.

Daher war vor allem die Einführung von Unicode ein wichtiger Beweggrund für dieses Buch. Die im System vorhandene ABAP-Schlüsselwortdokumentation wuchs mit der Sprache, indem Neues hinzugefügt und Altes unverändert gelassen wurde. Es gibt dort beispielsweise ein Teilgebiet »ABAP Objects« und seit Release 6.10 eben auch ein Teilgebiet »Unicode«, während die Anweisungen des klassischen Reporting nach wie vor so beschrieben sind, als wäre jedes ABAP-Programm ein Report, der mit einer logischen Datenbank verknüpft ist. Da diese Art der Dokumentation als Direkthilfe zwar vertretbar, für eine einheitliche Gesamtbeschreibung aber nicht mehr ausreichend ist, haben wir die Gelegenheit ergriffen, der Einführung in die SAP-Programmierung mit ABAP Objects eine ABAP-Referenz folgen zu lassen.

Das Ziel, das wir uns gesteckt haben, ist die vollständige Neufassung und Vereinheitlichung der ABAP-Dokumentation. Dies betrifft nicht nur die Gliederung und Zusammenfassung der einzelnen Anweisungen, sondern auch die inhaltliche Aufbereitung und stilistische Anpassung der Beschreibung. Speziell für die Darstellung der Syntaxdiagramme wurde daher eine neue Sichtweise in Form einer Pseudosyntax eingeführt, die jede Anweisung ausgehend von ihrer Grundform schrittweise in ihre einzelnen Bestandteile auflöst. Erstmals werden auch die Anforderungen an alle in einer ABAP-Anweisung vorkommenden Operanden vollständig und in

einer einheitlichen Form beschrieben. Unser Anspruch ist die umfassende Beschreibung aller im aktuellen Release 6.20 zur Verwendung freigegebenen ABAP-Sprachelemente. Dies beinhaltet auch eine detaillierte Beschreibung aller zwar als obsolet gekennzeichneten, aber nur in ABAP Objects verbotenen Sprachelemente. Die ABAP-Referenz soll den Leser in die Lage versetzen, jedes ABAP-Programm zu analysieren und neue Programme zu schreiben. Sie kann und will zwar kein Best-Practices-Buch sein, wir hoffen aber, dass Sinn und Zweck mancher Anweisung durch die genaue Beschreibung klarer wird, als es bisher der Fall war.

Es ist leichter gesagt als getan, eine Sprache mit etwa 500 wesentlichen Sprachelementen durchgehend auf gleich hohem Niveau und in gleichem Detaillierungsgrad zu beschreiben, zumal ABAP seit zwanzig Jahren ständig weiterentwickelt wird und aus Kompatibilitätsgründen kaum eine Entwicklung jemals wieder zurückgenommen wurde. Dies erklärt auch die wiederholten Verschiebungen des Erscheinungstermins dieses Buches und wir danken allen, die so lange gewartet haben. Als Entschädigung halten Sie jetzt aber ein Buch in Händen, das das aktuellste Release der Sprache ABAP beschreibt und lange gültig bleiben wird. Da wir jedoch trotz sorgfältiger Durchsicht aller Kapitel nicht ausschließen können, dass dieses Buch noch kleinere Unstimmigkeiten enthält, freuen wir uns auf alle diesbezüglichen Anregungen und Hinweise. Sie können sie auf den Internetseiten von SAP PRESS unter www.sap-press.de einreichen.

Ohne die mittelbare und unmittelbare Hilfe vieler Personen bei der Erstellung und Prüfung des Manuskripts wäre das Buch in der vorliegenden Form nicht zustande gekommen. An dieser Stelle gilt der Dank den Kollegen Masoud Aghadavoodi, Thomas Bareiss, Adrian Goerler, Christian Jendel, Gerd Kluger, Björn Mielenhausen, Andreas Simon Schmitt und Christoph Stöck. Erhardt Vortanz danken wir für seine Mitarbeit bei der Erstellung des Manuskripts. Dem Development Manager der Gruppe Business Programming Languages, Andreas Blumenthal, danken wir, dass er uns dieses Projekt überhaupt ermöglichte und weitgehend freie Hand in der Gestaltung ließ. Dem unermüdlichen Michael Demuth ist es zu verdanken, dass diesem Buch wieder zwei CDs mit einem SAP-System, dieses Mal in Form eines SAP Web Application Server 6.10, beiliegen. Dank gilt schließlich auch allen Mitarbeitern von Galileo Press – insbesondere Iris Warkus und Florian Zimniak – für die gute Zusammenarbeit bei der Korrektur des Manuskripts und dafür, dass sie die Hoffnung nie aufgaben, dass wir unser Manuskript irgendwann überhaupt noch einmal einreichen würden.

Horst Keller dankt ganz besonders seiner Frau Ute, die ihn in den letzten Monaten fast nur noch über dem aufgeklappten Laptop zu Gesicht bekam, für ihre Geduld und ihr Verständnis dafür, dass ein großer Anteil der ohnehin spärlichen gemeinsamen Freizeit schon wieder für ein Buchprojekt geopfert wurde.

Wir wünschen nun allen Leserinnen und Lesern viel Spaß beim Durchstöbern dieser ABAP-Referenz.

Walldorf, im Juli 2002

Horst Keller, Joachim Jacobitz

40 ABAP und XML

Durch Aufruf der Anweisung `CALL TRANSFORMATION` (siehe Abschnitt 40.4) lassen sich ABAP-Daten in das XML-Format umwandeln und umgekehrt. Dabei werden Transformationsprogramme aufgerufen, die entweder als XSL-Transformationen oder als Simple Transformations (seit Release 6.40) vorliegen können.



40.1 XSL-Transformationen

Eine XSL-Transformation ist ein in XSLT geschriebenes Programm im Repository (XSLT-Programm) zur Transformation von XML-Dokumenten. Beim Aufruf einer XSL-Transformation mit der Anweisung `CALL TRANSFORMATION` können auch ABAP-Daten direkt nach XML transformiert werden und umgekehrt. Hierfür wird implizit eine Serialisierung bzw. Deserialisierung vorgenommen.

Bei Transformationen, die ABAP-Daten als Quelle verwenden, werden die ABAP-Daten zuerst in eine kanonische XML-Repräsentation (asXML, siehe 40.2) serialisiert, die dann als eigentliche Quelle der XSL-Transformation dient. Bei Transformationen, die ABAP-Daten als Resultat erwarten, wird das Resultat der XSL-Transformation in die ABAP-Daten deserialisiert. Voraussetzung für die Deserialisierung ist, dass das Resultat eine kanonische XML-Repräsentation darstellt.

Für die Ausführung der Transformationen enthält die ABAP-Laufzeitumgebung seit Release 6.10 einen XSLT-Prozessor. Dieser unterstützt fast alle Anweisungen von XSLT und bietet Erweiterungen (so genannte Extension Instructions) wie etwa die Möglichkeit, ABAP-Methoden aus XSLT-Programmen aufzurufen.

40.1.1 XSL-Transformationen im Repository

Mit `CALL TRANSFORMATION` aufrufbare XSL-Transformationen müssen als XSLT-Programme im Repository vorhanden sein. Im Object Navigator der ABAP Workbench können XSLT-Programme über **Objekt bearbeiten · Weitere · Transformation** (bzw. **XSLT-Programm** vor Release 6.40 und **XSL-Transformation** vor Release 6.20) und Auswahl von XSLT-Programm angelegt und bearbeitet werden.

Unter der Bezeichnung ID wird von SAP die Identitäts-Transformation ausgeliefert. Bei einer Identitäts-Transformation von XML nach XML ist das Resultat eine Kopie des Quelldokuments. Bei einer Identitäts-Trans-

formation von ABAP nach XML ist das Resultat die kanonische XML-Repräsentation (asXML) der ABAP-Daten (explizite Serialisierung). Bei einer Identitäts-Transformation von XML nach ABAP wird eine kanonische XML-Repräsentation in ABAP-Daten transformiert (explizite Deserialisierung).

40.2 Kanonische XML-Repräsentation

Die kanonische XML-Repräsentation ist das Format eines XML-Dokuments, das bei einer Serialisierung von ABAP-Daten entsteht bzw. für eine Deserialisierung vorausgesetzt wird. Dieses Format wird auch als asXML (ABAP Serialization XML) bezeichnet. Die kanonische XML-Repräsentation unterstützt alle ABAP-Datentypen.

Das asXML-Format ist in folgenden Fällen von Bedeutung:

- ▶ Für selbst geschriebene XSL-Transformationen von ABAP-Daten in beliebige XML-Formate muss das asXML-Format des Ergebnisses der Serialisierung bekannt sein.
- ▶ Wenn externe XML-Dokumente erzeugt werden sollen, die in ABAP-Daten deserialisierbar sind, müssen diese im asXML-Format vorliegen.

Hinweis

Das asXML-Format von serialisierten ABAP-Daten oder Objekten kann mit der vordefinierten Identitäts-Transformation ID erzeugt und untersucht werden.

40.2.1 Generelles asXML-Format

Die folgenden Zeilen zeigen das generelle Format der kanonischen XML-Repräsentation¹ ohne den XML-Header, wobei Zeilenumbrüche und Einrückungen hier nur zur Verdeutlichung eingefügt wurden. Ein ausführliches Beispiel findet sich in Abschnitt 40.5.

```
<asx:abap version = "1.0"  
    xmlns:asx = "http://www.sap.com/abapxml">  
  <asx:values>  
    <bn1>...</bn1>  
    ...  
    <bnn>...</bnn>  
  </asx:values>
```

¹ Das asXML-Format ist ein generelles Format, das nicht vollständig durch ein XML-Schema definiert werden kann. Der Grund hierfür ist, dass auf beliebige ABAP-Typen Bezug genommen wird.

```

<asx:heap>
...
</asx:heap>
</asx:abap>

```

Das Wurzelement eines asXML-Dokuments ist `abap` im Namensraum (XML-Namespace) `http://www.sap.com/abapxml`. Das optionale Attribut `version` hat zurzeit immer den Wert »1.0« und ist für zukünftige Erweiterungen von asXML vorgesehen. Das Wurzelement `abap` muss ein Untererelement `values` des gleichen Namensraums enthalten. Die Untererelemente `bn1` von `values` repräsentieren dabei die ABAP-Datenobjekte, die im Zusatz `source` der Anweisung `CALL TRANSFORMATION` als `e1`, `e2`, ... bzw. im Zusatz `result` als `f1`, `f2`, ... angegeben sind (siehe Abschnitt 40.4). Die Namen der Elemente `bn1`, `bn2`, ... sind die dort angegebenen Namen in Großbuchstaben. Die Textinhalte der Elemente `<bn1>...</bn1>` (bzw. `<bn1 ... />`), ... stellen die Inhalte benannter Datenobjekte außer Referenzvariablen dar. Referenzvariablen werden durch Elemente ohne Textinhalt, aber mit einem speziellen Attribut dargestellt (siehe Abschnitt 40.2.3.1). Das optionale Element `heap` enthält die Inhalte von referenzierten anonymen Datenobjekten und Objekten (siehe Abschnitte 40.2.3.2 und 40.2.3.3).

Die Bezeichner der Elemente `bn1`, `bn2`, ... enthalten mit Ausnahme der Sonderfälle in Tabelle 40.1 nur Großbuchstaben. Die in den Zusätzen `source` und `result` der Anweisung `CALL TRANSFORMATION` angegebenen Namen `bn1`, `bn2`, ... (bzw. Komponenten von Strukturen oder Objekten, siehe 40.2.2.2 und 40.2.3.3) können nur dann als (großbuchstabige) Namen für XML-Elemente verwendet werden, wenn sie ausschließlich die Zeichen »a« bis »z«, »A« bis »Z«, »0« bis »9« oder »_« enthalten, wobei das erste Zeichen ein Buchstabe oder »_« sein muss. Andere Zeichen werden gemäß Tabelle 40.1 ersetzt.

Zeichen im ABAP-Namen	Ersetzungszeichen im XML-Namen
ASCII-Zeichen ungleich »a« bis »z«, »A« bis »Z«, »0« bis »9« oder »_« und Zeichen »0« bis »9« als erstes Zeichen.	»--hex(c)«, wobei hex(c) die zweistellige Hexadezimaldarstellung des ASCII-Codes des Zeichens c ist.
»/«	»_«
»xml« als die ersten drei Zeichen in beliebiger Kombination von Groß- und Kleinschreibung	»x-ml« in entsprechender Kombination von Groß- und Kleinschreibung

Tabelle 40.1 Ersetzungsregeln für Zeichen in ABAP-Namen

40.2.2 asXML-Format für benannte Datenobjekte außer Referenzvariablen

Benannte Datenobjekte außer Referenzvariablen werden als die Textinhalte der Elemente `<bn1>...</bn1>`, ... dargestellt. Die Darstellung benannter Datenobjekte in `<bn1>...</bn1>`, ... richtet sich nach dem dazugehörigen ABAP-Datentyp.

40.2.2.1 Elementare Datentypen

Die asXML-Darstellung elementarer Datenobjekte mit eingebauten ABAP-Typen aus Tabelle 5.2 entspricht der kanonischen Repräsentation von XML-Schema-Datentypen (<http://www.w3.org/TR/xmlschema-2/#built-in-datatypes>, siehe Tabelle 40.2), wobei Datum und Zeit nach ISO-8601 und binäre Daten zur Basis 64 dargestellt werden.

ABAP-Typ	ABAP-Beispiel	XML-Schematyp	XML-Beispiel
c	" Hi "	string	" Hi "
d	"20020204"	date	"2002-02-04"
f	-3.140...0E+02	double	"-3.14E2"
i, b, s	-123	int, unsignedByte, short	"-123"
n	"001234"	string (pattern [0-9]+)	"001234"
p	-1.23	decimal	"-1.23"
string	" Hello "	string	" Hello "
t	"201501"	time	"20:15:01"
x	"ABCDEF"	base64Binary	"q83v"
xstring	"456789AB"	base64Binary	"RweJqw=="

Tabelle 40.2 Kanonische XML-Repräsentation (asXML) von eingebauten ABAP-Typen

40.2.2.2 Strukturen

Die Komponenten einer ABAP-Struktur werden in asXML als eine Folge von Unterelementen des Struktur-Elements dargestellt. Der Inhalt jedes Unterelements entspricht der kanonischen Repräsentation des Komponentenwertes. Der Name jedes Unterelements ist der Name der entsprechenden Komponente. Bei der Serialisierung werden die Unterelemente in der Reihenfolge der Komponenten in der Struktur dargestellt. Bei der Deserialisierung der asXML-Repräsentation einer Struktur spielt die Reihenfolge der Unterelemente keine Rolle und überzählige XML-Ele-

mente werden ignoriert. Komponenten der Struktur, für die es kein Unterelement gibt, bleiben initial.

40.2.2.3 Interne Tabellen

Die Zeilen einer internen Tabelle werden in asXML als eine Folge von Unterelementen des Tabellen-Elements dargestellt. Der Inhalt jedes Unterelements entspricht der kanonischen Repräsentation des Zeilenwertes. Der Name eines Unterelements ist irrelevant. Wenn die kanonische XML-Repräsentation durch eine Serialisierung erstellt wird, wird bei Bezug des Zeilentyps auf das ABAP Dictionary der dortige Name verwendet, ansonsten der Name `item`. Jede Tabellenart ist erlaubt. Bei der Serialisierung wird keine Information über die Tabellenart in das XML-Dokument übertragen. Wenn das Zielfeld einer XSL-Transformation eine sortierte Tabelle ist, werden die Zeilen bei der Deserialisierung entsprechend sortiert.

40.2.3 asXML-Format für Referenzvariablen und referenzierte Objekte

Anonyme Datenobjekte und Instanzen von Klassen (Objekte) werden in ABAP ausschließlich über Referenzen in Referenzvariablen adressiert. Das zugehörige asXML-Format besteht aus Unterelementen von `values` für benannte Referenzvariablen (siehe Abschnitt 40.2.3.1) und Unterelementen von `heap` (siehe Abschnitt 40.2.3.2 und 40.2.3.3) für die referenzierten Objekte. Der Bezug der Referenz-Elemente auf die Objekt-Elemente wird über einen XML-Referenz-Mechanismus hergestellt, bei dem ein referenziertes Objekt im gleichen XML-Dokument über einen Schlüssel identifiziert wird. Bei den Objekt-Elementen unter `heap` ist der dynamische Typ der Referenzvariablen zum Zeitpunkt der Serialisierung angegeben, um eine eindeutige Deserialisierung zu gewährleisten.

40.2.3.1 Benannte Referenzvariablen

Eine benannte Referenzvariable wird als einziges Attribut des zugehörigen Unterelements von `values` ohne Textinhalt dargestellt. Ein Attribut für eine Referenzvariable hat den Namen `href` und den Inhalt `"#key"` wobei `key` der eindeutige Schlüssel eines Objekts im Element `heap` ist. Ein Element einer initialen Referenz hat kein Attribut `href` und keinen sonstigen Inhalt. Bei der Serialisierung wird der Schlüssel `key` von der ABAP-Laufzeitumgebung gesetzt, für die Deserialisierung kann der Schlüssel beliebig sein.

40.2.3.2 Anonyme Datenobjekte

Die Darstellung eines anonymen Datenobjekts als Unterelement von `heap` erfolgt in der Form:

```
<asx:heap xmlns:namespace ...>
  <type id = "key" attri="...">...</type>
</asx:heap>
```

Der Wert eines solchen Unterelements wird in der asXML-Darstellung für benannte Datenobjekte (siehe Tabelle 40.2) bzw. für benannte Referenzvariablen (siehe Abschnitt 40.2.3.1) dargestellt. Falls das anonyme Datenobjekt selbst eine nicht-initiale Referenzvariable ist, verweist sie nach obigen Regeln auf ein weiteres Element von `heap`. Der Elementname `type` ist der als XML-Schematyp-Name aus dem Namensraum `namespace` (siehe Tabelle 40.3) angegebene Datentyp des Datenobjekts (bzw. der dynamische Typ der Referenzvariablen), wobei gegebenenfalls Attribute `attri` technische Eigenschaften des Typs angeben. Das obligatorische Attribut `id` enthält den eindeutigen Schlüssel `key` des Elements, über den es von der Darstellung der zugehörigen Referenzvariablen in `values` oder `heap` referenziert wird.

Der XML-Schematyp-Name wird nach folgender Hierarchie konstruiert:

1. Falls der Datentyp des Datenobjekts im ABAP Dictionary definiert ist, ist der XML-Schematyp-Name der Name des Datentyps aus dem ABAP Dictionary im zugehörigen Namensraum (siehe Tabelle 40.3).
2. Falls der Datentyp ein elementarer ABAP-Typ ist, ist der XML-Schematyp-Name in Tabelle 40.4 angegeben.
3. Falls der Datentyp als Komponente einer globalen oder lokalen Klasse bzw. eines Interfaces definiert ist, setzt sich der XML-Schematyp-Name aus dem Namen der Klasse bzw. des Interfaces und dem Namen des Datentyps zusammen, die durch einen Punkt (.) getrennt sind. Der zugehörige Namensraum (siehe Tabelle 40.3) zeigt an, ob es sich um eine globale oder lokale Klasse bzw. ein Interface handelt.
4. Falls der Datentyp ein mit `REF TO data` bzw. `REF TO object` definierter generischer Referenztyp ist, ist der XML-Schematyp-Name `refData` bzw. `refObject`. Beide haben den Namensraum `http://www.sap.com/abapxml/types/built-in`.
5. Ansonsten ist der XML-Schematyp-Name der Name eines mit `TYPES` definierten Datentyps, wobei der zugehörige Namensraum (siehe Tabelle 40.3) anzeigt, wo der Datentyp definiert ist.

Voraussetzung für die Konstruktion eines XML-Schematyp-Namens ist, dass der Datentyp des Datenobjekts einen statisch verwendbaren Namen hat. Falls der Datentyp nur als Eigenschaft eines Datenobjekts existiert und damit nur einen technischen Namen hat (vergleiche Abschnitt 6.1.4), kommt es bei der Serialisierung zu einer behandelbaren Ausnahme.

Tabelle 40.3 zeigt die Namensräume für die XML-Schematyp-Namen, wobei `types` in der ersten Spalte für `http://www.sap.com/abapxml/types` steht. Die Namensräume zeigen an, wo ein Datentyp definiert ist. In den Bezeichnern `prg`, `cpool`, `fpool`, `tpool`, `meth`, `func`, `form` und `class` werden Zeichen ungleich »a« bis »z«, »A« bis »Z«, »0« bis »9«, »_« oder »-« als »!hex(c)« dargestellt, wobei `hex(c)` die zweistellige Hexadezimaldarstellung des ASCII-Codes des Zeichens `c` ist.

Namensraum	Stelle der Definition
<code>types/dictionary</code>	ABAP Dictionary
<code>types/program/prg</code>	ABAP-Programm <code>prg</code>
<code>types/class-pool/cpool</code>	Class-Pool <code>cpool</code>
<code>types/type-pool/tpool</code>	Typgruppe <code>tpool</code>
<code>types/function-pool/fpool</code>	Funktionsgruppe <code>fpool</code>
<code>types/function/func</code>	Funktionsbaustein <code>func</code>
<code>types/program.form/prg/frm</code>	Unterprogramm <code>frm</code> in Programm <code>prg</code>
<code>types/function-pool.form/fpool/frm</code>	Unterprogramm <code>frm</code> in Funktionsgruppe <code>fpool</code>
<code>types/method/class/meth</code>	Methode <code>meth</code> einer globalen Klasse <code>class</code>
<code>types/program.method/prg/class/meth</code>	Methode <code>meth</code> einer lokalen Klasse <code>class</code> in Programm <code>prg</code>
<code>types/class-pool.method/cpool/class/meth</code>	Methode <code>meth</code> einer lokalen Klasse <code>class</code> in Class-Pool <code>cpool</code>
<code>types/function-pool.method/fpool/class/meth</code>	Methode <code>meth</code> einer lokalen Klasse <code>class</code> in Funktionsgruppe <code>fpool</code>

Tabelle 40.3 Namensräume für XML-Schematyp-Namen, wobei `types` in der ersten Spalte für `http://www.sap.com/apaxml/types` steht

Tabelle 40.4 zeigt die XML-Schematyp-Namen für elementare ABAP-Typen. Diese unterscheiden sich teilweise von den kanonischen XML-Schema-Datentypen aus Tabelle 40.2, da der Datentyp von anonymen Datenobjekten vollständig spezifiziert sein muss. Die Namensräume

namespace für die elementaren ABAP-Typen anonymer Datenobjekte sind entweder `xsd="http://www.w3.org/2001/XMLSchema"` für allgemeine Schematypen oder `abap="http://www.sap.com/abapxml/types/built-in"` für spezielle ABAP-Schematypen, bei denen teilweise technische Attribute angegeben werden müssen.

ABAP-Typ	XML-Schematyp-Name	Attribute
c	abap:string	maxLength
d	abap:date	-
f	xsd:double	-
i, b, s	xsd:int, xsd:unsigned Byte, xsd:short	-
n	abap:digits	maxLength
p	abap:decimal	totalDigits, fractionDigits
string	xsd:string	-
t	abap:time	-
x	abap:base64Binary	maxLength
xstring	xsd:base64Binary	-

Tabelle 40.4 XML-Schematyp-Namen für elementare ABAP-Typen

Das Attribut `maxLength` gibt die Länge für die ABAP-Typen generischer Länge an. Der XML-Schematyp `abap:digits` beschränkt den Wertebereich eines Elements auf Ziffern. Beim XML-Schematyp `abap:decimal` werden die Länge und die Nachkommastellen über die Attribute `totalDigits` und `fractionDigits` angegeben. Die Längenangabe `totalDigits` gibt die Anzahl der Stellen zwischen 1 und 31 an. In ABAP-Programmen wird die Länge für Datenobjekte vom Typ `p` in Bytes angegeben und die Anzahl der Dezimalstellen berechnet sich aus $2 \times \text{len} - 1$ (siehe Tabelle 5.2). Bei der Serialisierung ist der Wert von `totalDigits` daher immer ungerade. Bei der Deserialisierung wird ein gerader Wert von `totalDigits` implizit um eins erhöht.

Instanzen von Klassen

Die Darstellung der Instanz einer Klasse (Objekt) als Unterelement von `heap` erfolgt in der Form:

```
<asx:heap xmlns:namespace ...>
  <class id = "key">
    <part classVersion = "...">
```

```

    <name>...</name>
  </part>
  ...
</class>
</asx:heap>

```

Der Elementname `class` ist der XML-Schematyp-Name der Klasse des Objekts (bzw. der dynamische Typ der Referenzvariablen) aus dem Namensraum `nspac` (siehe Tabelle 40.5) in Großbuchstaben. Das obligatorische Attribut `id` enthält den eindeutigen Schlüssel `key` des Elements, über den es von der Darstellung der zugehörigen Referenzvariablen in `values` referenziert wird. Die Unterelemente `<part>...</part>` enthalten die Werte der Instanzattribute einzelner Objektteile als Unterelemente `<name>...</name>`. Die einzelnen Objektteile werden durch die serialisierbaren Klassen der aktuellen Vererbungshierarchie definiert (siehe unten).

Der Namensraum des Klassennamens gibt an, wo die Klasse definiert ist. Tabelle 40.5 zeigt die möglichen Namensräume, wobei `classes` in der ersten Spalte für `http://www.sap.com/apapxml/classes` steht. Für die Bezeichner `prg`, `cpool` und `fpool` gilt die gleiche Ersetzungsregel wie für die Namensräume in Tabelle 40.3.

Namensraum	Stelle der Definition
<code>classes/global</code>	Klassenbibliothek
<code>classes/program/prg</code>	Programm <code>prg</code>
<code>classes/class-pool/cpool</code>	Class-Pool <code>cpool</code>
<code>classes/function-pool/fpool</code>	Funktionsgruppe <code>fpool</code>

Tabelle 40.5 Namensräume für Klassennamen, wobei `classes` in der ersten Spalte für `http://www.sap.com/apapxml/classes` steht

Die serialisierbaren Werte einer Instanz einer Klasse (Instanzattribute oder Ausgabeparameter einer speziellen Methode, siehe unten) werden in der asXML-Darstellung für benannte Datenobjekte (siehe Tabelle 40.2) bzw. für Referenzvariablen als Inhalt bzw. Attribut von `<name>...</name>` dargestellt, wobei `name` der Name eines Instanzattributs oder Ausgabeparameters in Großbuchstaben ist. Falls es sich um ein Interface-Attribut handelt, steht vor dem Namen der durch einen Punkt (`.`) abgetrennte Namen des Interfaces, um es von einem gleichnamigen Klassenattribut zu unterscheiden. Für die Bezeichner gelten die Ersetzungsregeln aus Tabelle 40.1.

Die serialisierbaren Werte einer Instanz einer Klasse werden durch Implementierung des Systeminterfaces `IF_SERIALIZABLE_OBJECT` in der Klasse festgelegt (siehe Abschnitt B.2). Wenn die Klasse das Interface `IF_SERIALIZABLE_OBJECT` nicht implementiert, enthält das Element `class` keine Unterelemente. Standardmäßig werden alle Instanzattribute einer Klasse, in der das Interface `IF_SERIALIZABLE_OBJECT` implementiert ist, serialisiert und in diese deserialisiert. Dieses Verhalten ist durch die Deklaration spezieller Hilfsmethoden änderbar (siehe unten). Statische Attribute werden weder bei der Serialisierung noch bei der Deserialisierung berücksichtigt (mit Ausnahme der speziellen Konstanten `SERIALIZABLE_CLASS_VERSION`, siehe unten).

Standardverhalten

Wenn die Klasse das Interface `IF_SERIALIZABLE_OBJECT` implementiert, enthält das Element `<class>...</class>` mindestens ein Unterelement `<part>...</part>`. Diese Unterelemente entsprechen einzelnen serialisierbaren Objektteilen und enthalten die Darstellungen der Instanzattribute des jeweiligen Objektteils im asXML-Format. Ein Objektteil wird durch die Klasse bestimmt, in der Instanzattribute deklariert sind bzw. in der ein Interface eingebunden ist, das Instanzattribute enthält. Eine serialisierbare Klasse enthält einen Objektteil für sich selbst sowie Objektteile für alle Oberklassen im aktuellen Pfad des Vererbungsbaums bis einschließlich der Klasse, die das Interface `IF_SERIALIZABLE_OBJECT` implementiert. Der Name `part` ist der Name der jeweiligen Klasse. Falls es sich um eine lokale Klasse handelt, steht vor dem Namen das durch einen Punkt (.) abgetrennte Präfix `local`, um diese von einer gleichnamigen globalen Klasse zu unterscheiden. Objektteile von Oberklassen, in denen das Interface `IF_SERIALIZABLE_OBJECT` nicht implementiert ist, sind nicht serialisierbar und haben kein entsprechendes Unterelement `part`. D.h., eine Klasse, in der das Interface `IF_SERIALIZABLE_OBJECT` nicht implementiert ist (weder in der Klasse selbst noch in einer Oberklasse), erzeugt bei der Serialisierung ein leeres XML-Element `class`.

Bei der Serialisierung werden die XML-Elemente `part` der Objektteile von den Oberklassen zu den Unterklassen hin erzeugt und standardmäßig die XML-Elemente der Instanzattribute in der Reihenfolge angelegt, wie sie in der Klasse deklariert sind.

Bei der Deserialisierung wird ein Objekt der betreffenden Klasse erzeugt, wobei der Instanzkonstruktor nicht ausgeführt wird. Alle Instanzattribute haben nach der Objekterzeugung ihren Initialwert bzw. den mit dem Zusatz `VALUE` der Anweisung `DATA` angegebenen Startwert. Standard-

mäßig werden die Instanzattribute mit den Werten der zugehörigen XML-Elemente versorgt, wobei die Reihenfolge der Objektteile und der Attribute keine Rolle spielt. Instanzattribute ohne zugehöriges XML-Element behalten ihren Wert. Überflüssige XML-Elemente werden ignoriert, solange sie keinem Namensraum angehören, ansonsten erzeugen sie eine behandelbare Ausnahme. Bei der Deserialisierung eines Elements ohne Unterelemente `part` wird kein Objekt erzeugt, sondern die Zielreferenzvariable initialisiert.

Wenn eine Klasse das Interface `IF_SERIALIZABLE_OBJECT` implementiert, kann in jedem Objektteil, d.h. in jeder beteiligten Klasse des Vererbungsbaums, die private Konstante `SERIALIZABLE_CLASS_VERSION` vom Typ `i` deklariert werden. Bei der Serialisierung wird der Wert der Konstanten dem Attribut `classVersion` des XML-Elements `part` zugewiesen. Standardmäßig kommt es bei der Deserialisierung zu einer behandelbaren Ausnahme, wenn der Wert des Attributs nicht mit dem Wert der Konstanten in der angegebenen Klasse übereinstimmt. Ein Objektteil ist nur deserialisierbar, wenn die Werte übereinstimmen oder weder das Attribut noch die Konstante vorhanden sind. Dieses Verhalten kann durch Deklaration spezieller Hilfsmethoden geändert werden.

Angepasstes Verhalten

Standardmäßig werden alle Instanzattribute eines Objektteils unabhängig von ihrer Sichtbarkeit serialisiert und die Version der Klasse wird überprüft. Um dieses Verhalten zu ändern, können für jeden Objektteil die Instanzmethoden `SERIALIZE_HELPER` und `DESERIALIZE_HELPER` in der jeweiligen Klasse deklariert und implementiert werden. Diese Methoden können nur als private Instanzmethoden in Klassen deklariert werden, die das Interface `IF_SERIALIZABLE_OBJECT` implementieren. Die Deklaration einer der Methoden bedingt die Deklaration der anderen und die Schnittstelle wird von der Syntaxprüfung wie folgt vorgeschrieben:

- ▶ Die Methode `SERIALIZE_HELPER` darf nur Ausgabe- und die Methode `DESERIALIZE_HELPER` nur Eingabeparameter haben, deren Typisierung nicht generisch ist.
- ▶ Zu jedem Ausgabeparameter der Methode `SERIALIZE_HELPER` muss es einen gleichnamigen Eingabeparameter der Methode `DESERIALIZE_HELPER` gleicher Typisierung geben. Zusätzliche Eingabeparameter der Methode `DESERIALIZE_HELPER` müssen optional sein.

- ▶ Die Methode `SERIALIZE_HELPER` darf keinen Ausgabeparameter des Namens `SERIALIZABLE_CLASS_VERSION`, die Methode `DESERIALIZE_HELPER` darf einen solchen optionalen Eingabeparameter vom Typ `i` haben. Dieser wird bei der Deserialisierung mit dem Wert des Attributs `classVersion` des Elements `part` versorgt, wobei die Standardüberprüfung der Version (siehe oben) umgangen wird.

Wenn in einem Objektteil die Methoden `SERIALIZE_HELPER` und `DESERIALIZE_HELPER` deklariert sind, werden nicht die Instanzattribute des Objektteils serialisiert und deserialisiert. Stattdessen wird bei der Serialisierung die Methode `SERIALIZE_HELPER` ausgeführt und die Werte aller Ausgabeparameter werden in der angegebenen Reihenfolge im asXML-Format als Unterelemente in das entsprechende Element `part` geschrieben. Dabei ist der Name eines Unterelements der Name des jeweiligen Ausgabeparameters in Großbuchstaben. Bei der Deserialisierung wird die Methode `DESERIALIZE_HELPER` aufgerufen, wobei die Werte der Unterelemente des entsprechenden Elements `part` den gleichnamigen Eingabeparametern der Methode übergeben werden. Dabei spielt die Reihenfolge keine Rolle und überflüssige XML-Elemente werden ignoriert.

40.3 Simple Transformations



Simple Transformations (ST) ist eine SAP-eigene Programmiersprache, um Transformationen zwischen ABAP-Daten und XML-Formaten zu beschreiben. ST ist auf die beiden für die Datenintegration wesentlichen Modi der Serialisierung (ABAP nach XML) und Deserialisierung (XML nach ABAP) von ABAP-Daten beschränkt. Transformationen von ABAP nach ABAP und XML nach XML wie beim allgemeineren XSLT sind in ST nicht möglich.

Im Gegensatz zu XSLT liegen bei ST-Programmen die Schwerpunkte wie folgt:

- ▶ ST-Programme sind deklarativ und somit leicht lesbar.
- ▶ ST-Programme haben nur seriellen Zugriff auf die XML-Daten und sind deshalb auch für große Datenmengen sehr effizient.
- ▶ ST-Programme beschreiben gleichzeitig Serialisierung und Deserialisierung, d. h., in der Regel können mit ST in XML serialisierte ABAP-Daten mit demselben ST-Programm auch wieder deserialisiert werden.

Mit `CALL TRANSFORMATION` aufrufbare Simple Transformations müssen im Repository vorhanden sein. Im Object Navigator der ABAP Workbench können ST-Programme über **Objekt bearbeiten** · **Weitere** · **Transformation** und Auswahl von **Simple Transformation** angelegt und bearbeitet werden.

Eine genaue Beschreibung der Sprache ST geht über den Rahmen der vorliegenden ABAP-Referenz hinaus. Hier sei auf die entsprechende Online-Hilfe verwiesen. Ein einführendes Beispiel findet sich in Abschnitt 40.6.

40.4 Aufruf einer XSL- oder ST-Transformation

CALL TRANSFORMATION



Aufruf eines XSLT- oder ST-Programms.

Syntax

```
CALL TRANSFORMATION transformation
    [PARAMETERS parameters]
    [OBJECTS objects]
    [OPTIONS options]
    SOURCE source
    RESULT result.
```

Diese Anweisung ruft die angegebene XSL-Transformation (XSLT) oder eine Simple Transformation (ST, ab Release 6.40) auf. Die Quelle der Transformation wird hinter `SOURCE` angegeben und das Ergebnis wie hinter `RESULT` angegeben abgelegt. Mit `PARAMETERS` und `OBJECTS` können Parameter an die Transformation übergeben werden. Mögliche Transformationsarten sind:

- ▶ von XML nach XML (nur bei XSLT)
- ▶ von XML nach ABAP (bei XSLT und ST)
- ▶ von ABAP nach XML (bei XSLT und ST)
- ▶ von ABAP nach ABAP (nur bei XSLT)

Die letzten beiden Arten sind erst ab Release 6.20 verfügbar.

Behandelbare Ausnahmen

Die gemeinsame Oberklasse aller Ausnahmeklassen für `CALL TRANSFORMATION` ist `CX_TRANSFORMATION_ERROR` (seit Release 6.40). Die zugehörigen Laufzeitfehler sind nicht abfangbar.

Ausnahmen bei XSL-Transformationen

Die Ausnahmeklassen für XSL-Transformationen sind alle Unterklassen von `CX_XSLT_EXCEPTION`.

Tritt beim Parsen eines XML-Dokuments ein Fehler auf oder wird ein anderer Fehler vom XSLT-Prozessor gemeldet, wird eine durch die Klasse `CX_XSLT_RUNTIME_ERROR` definierte Ausnahme ausgelöst. Führt der Aufruf einer ABAP-Methode aus dem XSLT-Programm zu einem Fehler, wird eine durch die Klasse `CX_XSLT_ABAP_CALL_ERROR` definierte Ausnahme ausgelöst, wobei das Attribut `PREVIOUS` auf das Ausnahmeobjekt des ursprünglichen Fehlers zeigt.

Hat ein XML-Dokument bei der Deserialisierung nicht das asXML-Format, wird eine durch die Klasse `CX_XSLT_FORMAT_ERROR` definierte Ausnahme ausgelöst, wobei das Attribut `TREE_POSITION` die Fehlerposition enthält. Treten bei der Serialisierung oder Deserialisierung ungültige Werte oder Datentypen auf, werden durch die Klassen `CX_XSLT_SERIALIZATION_ERROR` bzw. `CX_XSLT_DESERIALIZATION_ERROR` definierte Ausnahmen ausgelöst, wobei das Attribut `PREVIOUS` gegebenenfalls auf das Ausnahmeobjekt des ursprünglichen Fehlers zeigt und das Attribut `TREE_POSITION` die Fehlerposition bei der Deserialisierung enthält.

Ausnahmen bei Simple Transformations

Die Ausnahmeklassen für Simple Transformations sind alle Unterklassen von `CX_ST_ERROR`.

40.4.1 Angabe der Transformation

Syntax von *transformation*

```
... trans | (name) ...
```

Der Name der Transformation kann entweder direkt als `trans` oder als Inhalt eines eingeklammerten zeichenartigen Datenobjekts `name` angegeben werden. Die angegebene Transformation muss als XSLT-Programm oder als Simple Transformation im Repository vorhanden sein.

40.4.2 Transformationsquelle

Syntax von *source*

```
... { XML sxml }
| {{bn1 = e1 bn2 = e2 ...}}(stab) ...
```

40.4.2.1 Transformation eines XML-Dokuments

Durch die Angabe von XML `sxml` wird das in `sxml` enthaltene XML-Dokument transformiert, wobei `sxml` in folgenden Formen vorliegen kann:

- ▶ als Datenobjekt vom Typ `string` und `xstring` oder als Standardtabelle mit flachem zeichenartigem Zeilentyp
- ▶ als Interface-Referenzvariable vom Typ `IF_IXML_ISTREAM`, die auf einen iXML-Input-Stream zeigt (nur bei XSLT)
- ▶ als Interface-Referenzvariable vom Typ `IF_IXML_NODE`, die auf ein iXML-Nodeset zeigt (nur bei XSLT)
- ▶ als Klassen-Referenzvariable vom Typ `CL_FX_READER`, die auf einen XML-Reader zeigt (nur bei ST)

Hinweis

Die Interfaces `IF_IXML_ISTREAM` und `IF_IXML_NODE` sind Komponenten der Pakete »Stream« und »DOM« der von SAP ausgelieferten iXML Library.

40.4.2.2 Transformation von ABAP-Daten

Mir `bn1 = e1`, `bn2 = e2`, ... oder (`stab`) werden die zu transformierenden ABAP-Daten `e1`, `e2`, ... angegeben.

- ▶ Bei Aufruf eines XSLT-Programms werden die ABAP-Daten in die kanonische XML-Repräsentation serialisiert, die dann als Quelle der XSL-Transformation verwendet wird. Mit `bn1`, `bn2`, ... werden die Namen der XML-Elemente angegeben, die die ABAP-Datenobjekte in der kanonischen XML-Darstellung repräsentieren sollen.
- ▶ Beim Aufruf einer Simple Transformation wird in der Transformation über die Namen `bn1`, `bn2`, ... lesend auf die ABAP-Daten zugegriffen.

Statt über eine statische Parameterliste können die Datenobjekte auch dynamisch als Wertepaare in den Spalten einer internen Tabelle `stab` übergeben werden, die den Typ `ABAP_TRANS_SRCBIND_TAB` aus der Typgruppe ABAP hat.

Folgende Datenobjekte sind nicht serialisierbar und führen zu einer behandelbaren Ausnahme:

- ▶ Datenobjekte vom Typ `n`, deren aktueller Inhalt nicht ausschließlich aus Ziffern besteht.
- ▶ Datenobjekte vom Typ `p`, deren aktueller Inhalt keine gültige gepackte Zahl darstellt.



- ▶ Datenobjekte vom Typ `d` und `t`, deren aktueller Inhalt führende oder schließende Leerzeichen und gleichzeitig die in der Darstellung nach ISO-8601 verwendeten Trennzeichen (»-« bzw. »:«) enthält.
- ▶ Datenreferenzvariablen, die auf Datenobjekte zeigen, deren Datentyp nur einen technischen Namen hat (siehe Abschnitt 40.2.3).

Datenreferenzvariablen, die auf Datenobjekte zeigen, die nicht mit `CREATE DATA` erzeugt wurden, werden bei der Serialisierung wie initiale Referenzvariablen behandelt.

40.4.3 Transformationsergebnis

Syntax von *result*

```
... { XML rxml }
  | {{bn1 = f1 bn2 = f2 ...}|(rtab)} ...
```

40.4.3.1 Transformation in ein XML-Dokument

Durch die Angabe von `XML rxml` wird in ein XML-Dokument transformiert und dieses nach `rxml` gestellt, wobei `rxml` in folgenden Formen vorliegen kann:

- ▶ als Datenobjekt vom Typ `string` und `xstring` oder als Standardtabelle mit flachem zeichenartigem Zeilentyp
- ▶ als Interface-Referenzvariable vom Typ `IF_IXML_OSTREAM`, die auf einen iXML-Output-Stream zeigt (nur bei XSLT)
- ▶ als Interface-Referenzvariable vom Typ `IF_IXML_DOCUMENT`, die auf ein iXML-Dokument zeigt (nur bei XSLT)
- ▶ als Klassen-Referenzvariable vom Typ `CL_FX_WRITER`, die auf einen XML-Writer zeigt (nur bei ST)

Hinweise

- ▶ Die Interfaces `IF_IXML_OSTREAM` und `IF_IXML_DOCUMENT` sind Komponenten der Pakete »Stream« und »DOM« der von SAP ausgelieferten iXML Library.
- ▶ Wenn der Datentyp `xstring` für `rxml` verwendet wird, wird das Ergebnis in der Zeichendarstellung UTF-8 abgelegt. Dies ist vorteilhaft, wenn das resultierende XML-Dokument in einer Datei gespeichert werden soll (siehe Kapitel 32).

40.4.3.2 Transformation in ABAP-Daten

Mit $bn1 = f1$, $bn2 = f2$, ... oder (r_{tab}) werden die ABAP-Zielfelder $f1$, $f2$, ... angegeben, in die die XML-Daten transformiert werden sollen.

- ▶ Bei Aufruf eines XSLT-Programms wird das Resultat der XSL-Transformation in ABAP-Datenobjekte deserialisiert, falls es eine kanonische XML-Repräsentation darstellt. Mit $bn1$, $bn2$, ... werden die Namen der XML-Elemente angegeben, die die ABAP-Datenobjekte in der kanonischen XML-Darstellung repräsentieren, während mit $f1$, $f2$, ... vom Datentyp her passende ABAP-Datenobjekte angegeben werden, in die diese deserialisiert werden sollen.
- ▶ Beim Aufruf einer Simple Transformation wird in der Transformation über die Namen $bn1$, $bn2$, ... schreibend auf die ABAP-Daten zugegriffen

Statt über eine statische Parameterliste können die Datenobjekte auch dynamisch als Wertepaare an die Spalten der internen Tabelle r_{tab} übergeben werden, die den Typ `ABAP_TRANS_RESBIND_TAB` aus der Typgruppe `ABAP` hat.

Ein XML-Element muss in das zugehörige ABAP-Datenobjekt konvertierbar sein, wobei abweichend zu den Konvertierungsregeln in Anhang A folgende Verschärfungen gelten:

- ▶ Es darf keinen Datenverlust durch Deserialisierung in zu kurze Datenobjekte vom Datentyp c oder n geben, außer es sind nur führende und schließende Leerzeichen beim Datentyp c und führende Nullen beim Datentyp n betroffen.
- ▶ Es darf keinen Datenverlust durch Deserialisierung in ein Datenobjekt vom Datentyp p mit zu wenig Dezimalstellen geben.
- ▶ Es darf keinen Datenverlust durch Deserialisierung in ein zu kurzes Datenobjekt vom Datentyp x geben.
- ▶ Strukturen können nicht in elementare Datenobjekte konvertiert werden.

Wenn ein XML-Element nicht in das ABAP-Datenobjekt konvertierbar ist, wird eine behandelbare Ausnahme ausgelöst.

Bei der Deserialisierung in eine Referenzvariable muss diese gleich oder allgemeiner als der dynamische Typ des im XML-Dokument abgelegten Objekts sein. Die zugehörigen ABAP-Datenobjekte bzw. Instanzen einer Klasse werden während der Deserialisierung erzeugt.

40.4.4 Parameter für eine XSL-Transformation

Syntax von *parameters*

```
... {p1 = e1 p2 = e2 ...}|(ptab) ...
```

Mit diesem Zusatz können ABAP-Datenobjekte *e1*, *e2*, ... als Parameter *p1*, *p2*, ... an eine XSL-Transformation übergeben werden. In Release 6.10 müssen die Datenobjekte *e1*, *e2*, ... zeichenartig sein, ab Release 6.20 sind alle elementaren Datenobjekte und Objektreferenzen erlaubt.

Statt über eine statische Parameterliste können die Parameter auch dynamisch als Wertepaare in den Spalten der internen Tabelle *ptab* übergeben werden, die den Typ `ABAP_TRANS_PARMBIND_TAB` aus der Typgruppe `ABAP` hat.

Die angegebenen Parameter müssen in der XSL-Transformation wie folgt als Eingabeparameter definiert sein:

```
<xsl:param name="..." type="..."/>
```

Das Attribut *name* muss den Namen des Parameters in Großbuchstaben darstellen. Für das optionale Attribut *type* kann einer der Typbezeichner `string`, `number`, `boolean`, `xstring`, `nodeset` oder `object(...)` angegeben sein, wobei in den Klammern hinter `object` der Name einer globalen ABAP-Klasse anzugeben ist.

Wenn in der XSL-Transformation kein Typ angegeben ist, werden die Datentypen elementarer Parameter gemäß Tabelle 40.6 auf XSL-Typen abgebildet.

ABAP-Datentyp	XSL-Parametertyp
c, d, n, string	string
i, s, b, f, p	number
x, xstring	string, wobei der Inhalt zur Basis 64 dargestellt wird

Tabelle 40.6 Abbildung von ABAP-Datentypen auf die Typen von XSL-Parametern

Falls in der XSL-Transformation die in Tabelle 40.6 gezeigten XSL-Typen explizit angegeben sind, müssen passende elementare ABAP-Parameter angegeben werden, die in den Typ konvertierbar sind:

- ▶ Der XSL-Typ `boolean` erwartet ABAP-Parameter vom Typ `c` der Länge 1, wobei ein Leerzeichen als »falsch« und ein anderes Zeichen als »wahr« interpretiert wird.
- ▶ Der XSL-Typ `xstring` erwartet ABAP-Parameter vom Typ `x` oder `xstring` und der Inhalt wird hexadezimal dargestellt.
- ▶ Die XSL-Typen `nodeset` und `object` erwarten eine Objektreferenzvariable, die auf eine Instanz einer Klasse zeigt, wobei der Typ `nodeset` passende Objekteigenschaften erwartet.

Falls ein Parameter nicht zum XSL-Typ passt, kommt es zu einer unbehandelbaren Ausnahme. Falls ein in der XSL-Transformation definierter Parameter nicht übergeben wird, wird er in der Transformation auf einen Standardwert gesetzt. Ein angegebener Parameter, der nicht in der XSL-Transformation definiert ist, wird ignoriert.

Hinweis

Die XSL-Typen `string`, `number`, `boolean` und `nodeset` sind XSL-Standardtypen, während `xstring` und `object` spezielle SAP-Erweiterungen darstellen. Der Typ `xstring` ermöglicht es, Byteketten hexadezimal statt zur Basis 64 darzustellen. Der Typ `object` erlaubt den Aufruf von ABAP-Methoden aus dem XSL-Programm.

40.4.5 Externe Objekte an eine XSL-Transformation übergeben

Syntax von *objects*

```
... {o1 = e1 o2 = e2 ...} |(otab) ...
```

Mit diesem Zusatz können Objektreferenzen `e1`, `e2`, ... als externe Objekte `o1`, `o2`, ... an eine XSL-Transformation übergeben werden, um dort deren Methoden aufzurufen.

Statt über eine statische Parameterliste können die Objekte auch dynamisch als Wertepaare in den Spalten der internen Tabelle `otab` übergeben werden, die den Typ `ABAP_TRANS_OBJBIND_TAB` aus der Typgruppe `ABAP` hat.

Hinweis

Der Zusatz `OBJECTS` ist ab Release 6.20 obsolet. Ab Release 6.20 werden externe Objekte wie Parameter behandelt und Objektreferenzen sollen entsprechend mit dem Zusatz `PARAMETERS` (siehe Abschnitt 40.4.4) übergeben werden.

40.4.6 Steuerung der Transformation

Syntax von *options*

```
... a1 = e1 a2 = e2 ...
```

Dieser Zusatz erlaubt die Angabe von Werten *e1*, *e2*, ... zu weiteren Steuerungsoptionen *a1*, *a2*, ... der Transformation. Die Werte *e1*, *e2*, ... müssen vom Typ *c* oder *string* sein.

Für *a1*, *a2*, ... können angegeben werden:

- ▶ **XML_HEADER** zur Steuerung der Ausgabe des XML-Headers, falls nach XML transformiert und in ein Datenobjekt vom Typ *c*, *string* oder eine interne Tabelle geschrieben wird.

Mögliche Werte	Bedeutung
no	Es wird kein XML-Header ausgegeben.
without_encoding	Es wird ein XML-Header ohne Angabe des Encodings ausgegeben.
full	Standardeinstellung, es wird ein XML-Header mit Angabe des Encodings ausgegeben.

- ▶ **DATA_REFS** zur Steuerung der Ausgabe von Datenreferenzen, falls von ABAP nach XML transformiert wird.

Mögliche Werte	Bedeutung
no	Standard bei ST, es werden keine Datenreferenzen ausgegeben.
heap	Standard bei XSLT und nur dort möglich, referenzierte Daten werden als Unterelemente des asXML-Elements <code><asx:heap></code> ausgegeben.
embedded	Referenzierte Daten werden mit der Referenz ausgegeben.

- ▶ **INITIAL_COMPONENTS** zur Steuerung der Ausgabe von initialen Strukturkomponenten, falls von ABAP nach XML transformiert wird.

Mögliche Werte	Bedeutung
include	Standardeinstellung, initiale Komponenten von Strukturen werden ausgegeben.
suppress	Initiale Komponenten von Strukturen werden nicht ausgegeben.

40.5 Beispiel für eine XSL-Transformation

Serialisierung von Datenobjekten in einen String `xmlstr` über die identische Transformation `ID`. Es werden ein Datumsfeld `date`, ein Zeitfeld `time` und eine Datenreferenzvariable `dref1` serialisiert. Die Datenreferenzvariable zeigt auf eine anonyme Objektreferenzvariable, die wiederum auf ein Objekt der Klasse `c2` zeigt. Solcherart serialisierte Objekte können persistent gespeichert werden, beispielsweise in einem Daten-Cluster. Nach dem Einlesen aus der Ablage werden die Objekte in andere Datenobjekte deserialisiert. Nach der Deserialisierung zeigt `dref2` auf eine andere anonyme Referenzvariable wie `dref1`. Dieses anonyme Datenobjekt wie auch die Instanz der Klasse `c2`, auf die es zeigt, werden bei der Deserialisierung erzeugt.

```
PROGRAM xmltst.

CLASS c1 DEFINITION.
  PUBLIC SECTION.
    INTERFACES if_serializable_object.
  PROTECTED SECTION.
    DATA carriers TYPE TABLE OF scarr.
ENDCLASS.

CLASS c2 DEFINITION INHERITING FROM c1.
  PUBLIC SECTION.
    METHODS constructor.
  PRIVATE SECTION.
    DATA lines TYPE i.
    METHODS: serialize_helper
              EXPORTING count TYPE i,
              deserialize_helper
              IMPORTING count TYPE i.
ENDCLASS.

CLASS c2 IMPLEMENTATION.
  METHOD constructor.
    super->constructor( ).
    SELECT * UP TO 2 ROWS
      FROM scarr
      INTO TABLE carriers.
  ENDMETHOD.
  METHOD serialize_helper.
    count = LINES( carriers ).
  ENDMETHOD.
  METHOD deserialize_helper.
```

```

        lines = count.
    ENDMETHOD.
ENDCLASS.

DATA: oref    TYPE REF TO object,
      dref1   LIKE REF TO oref,
      xmlstr  TYPE string,
      date    TYPE d,
      time    TYPE t,
      dref2   LIKE dref1.

...

CREATE DATA dref1 LIKE oref.
CREATE OBJECT dref1->* TYPE c2.

CALL TRANSFORMATION id
      SOURCE xmldat = sy-datum
            xmltim = sy-uzeit
            ref     = dref1
      RESULT XML xmlstr.

EXPORT obj = xmlstr TO DATABASE indx(hk)
      ID 'OBJECT'.

...

IMPORT obj = xmlstr FROM DATABASE indx(hk) ID 'OBJECT'.

CALL TRANSFORMATION id
      SOURCE XML xmlstr
      RESULT xmldat = date
            xmltim = time
            ref     = dref2.

```

Das bei der Serialisierung erzeugte XML-Dokument hat unten stehenden Inhalt, wobei hier Zeilenumbrüche und Einrückungen eingefügt wurden. Das Element `values` enthält die asXML-Darstellungen der drei übergebenen Datenobjekte (siehe Abschnitt 40.2). In den Bezeichnern `X-MLDAT` und `X-MLTIM` wurde »xml« gemäß Tabelle 40.1 ersetzt. Das Attribut `href` des Elements `REF` verweist über den Schlüssel »d1« auf die Darstellung des zugehörigen anonymen Datenobjekts im Element `heap`. Diese verweist über den Schlüssel »o3« auf die Darstellung der Instanz der Klasse `c2`, die sich ebenfalls im Element `heap` befindet. Diese Darstellung ist in die Objektteile für die Klassen `c1` und `c2` unterteilt. Im Objektteil für `c1` befindet sich die Darstellung der zweizeiligen strukturierten internen Tabelle `carriers`. Im Objektteil für `c2` befindet sich die Darstellung für den Ausgabeparameter `count` der Methode `SERIALIZE_HELPER`.

```

<?xml version="1.0" encoding="iso-8859-1" ?>
<asx:abap xmlns:asx="http://www.sap.com/
abapxml" version="1.0">
  <asx:values>
    <X-MLDAT>2003-04-15</X-MLDAT>
    <X-MLTIM>14:57:53</X-MLTIM>
    <REF href="#d1" />
  </asx:values>
  <asx:heap
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:abap="http://www.sap.com/abapxml/types/built-in"
    xmlns:cls="http://www.sap.com/abapxml/classes/global"
    xmlns:dic="http://www.sap.com/abapxml/types/dictionary">
    <abap:refObject href="#o3" id="d1" />
    <prg:C2
      xmlns:prg="http://www.sap.com/abapxml/classes/program/
XMLTST"
      id="o3">
      <local.C1>
        <CARRIERS>
          <SCARR>
            <MANDT>000</MANDT>
            <CARRID>AA</CARRID>
            <CARRNAME>American Airlines</CARRNAME>
            <CURRCODE>USD</CURRCODE>
            <URL>http://www.aa.com</URL>
          </SCARR>
          <SCARR>
            <MANDT>000</MANDT>
            <CARRID>AB</CARRID>
            <CARRNAME>Air Berlin</CARRNAME>
            <CURRCODE>DEM</CURRCODE>
            <URL>http://www.airberlin.de</URL>
          </SCARR>
        </CARRIERS>
      </local.C1>
      <local.C2>
        <COUNT>2</COUNT>
      </local.C2>
    </prg:C2>
  </asx:heap>
</asx:abap>

```

40.6 Beispiel für eine Simple Transformation

Serialisierung einer geschachtelten Struktur. In folgendem ABAP-Programmabschnitt wird eine geschachtelte Struktur `struc1` mit der Simple Transformation `ST_TRAFO` nach `xml_string` serialisiert und mit der gleichen Transformation wieder deserialisiert.

```
DATA: BEGIN OF struc1,  
      col1(10) TYPE c VALUE 'ABCDEFGHIJ',  
      col2      TYPE i VALUE 111,  
      BEGIN OF struc2,  
        col1 TYPE d VALUE '20040126',  
        col2 TYPE t VALUE '084000',  
      END OF struc2,  
    END OF struc1.
```

```
DATA: xml_string TYPE string,  
      result LIKE struc1.
```

```
TRY.
```

```
  CALL TRANSFORMATION st_trafo  
    SOURCE para = struc1  
    RESULT XML xml_string.
```

```
  ...
```

```
  CALL TRANSFORMATION st_trafo  
    SOURCE XML xml_string  
    RESULT para = result.
```

```
  CATCH cx_st_error.
```

```
  ...
```

```
ENDTRY.
```

Die Simple Transformation `ST_TRAFO` habe folgende Form:

```
<?sap.transform simple?>  
<tt:transform template="temp"  
  xmlns:tt="http://www.sap.com/transformation-templates"  
  version="0.1">  
  <tt:root name="PARAM"/>  
  <tt:template name="temp">  
    <X>  
    <X1>  
    <tt:value ref="PARAM.COL1" />
```

```

    </X1>
    <X2>
      <tt:value ref="PARA.COL2" />
    </X2>
    <X3>
      <X1>
        <tt:value ref="PARA.STRUC2.COL1" />
      </X1>
      <X2>
        <tt:value ref="PARA.STRUC2.COL2" />
      </X2>
    </X3>
  </X>
</tt:template>
</tt:transform>

```

Die Transformation besteht aus einem Template `temp`, das den Aufbau des XML-Dokuments definiert und Beziehungen zwischen Werteknoten und Komponenten der Struktur herstellt. Das Ergebnis der Transformation sieht wie folgt aus, wobei Zeilenumbrüche und Einrückungen zur besseren Verdeutlichung eingefügt wurden:

```

<X>
  <X1>ABCDEFGHIJ</X1>
  <X2>111</X2>
  <X3>
    <X1>2004-01-26</X1>
    <X2>08:40:00</X2>
  </X3>
</X>

```

Die Konvertierung der elementaren Datentypen ist wie bei `asXML` (siehe Tabelle 40.2). Die Rücktransformation erzeugt in der Struktur `result` den gleichen Inhalt wie in `struc1`.

Index

- <Tabellenkörper 61
 - - <Strukturkomponenten-<Selektor 54
 - #<EC
 - <Pseudokommentar 66
 - &
 - <Literaloperator 140
 - '
 - <Textfeldliteral 139
 - (,)
 - <arithmetischer <Ausdruck 428
 - <Bit-<Ausdruck 435
 - <CALL <METHOD 292
 - <logischer <Ausdruck 359
 - <Offset/<Längenangabe 61
 - <SELECT <INTO 793
 -) 65
 - *
 - <Kommentar 66
 - <SELECT 779
 - <TABLES, <obsolet 1064
 - <WRITE 680
 - **
 - <WRITE 680
 - *-<INPUT
 - <FIELD 567
 - + , -
 - <Vorzeichen 428
 - + , - , * , / , **
 - <arithmetische <Operatoren 429
 - '
 - <Kettensatz 65
 - <SELECT <INTO 793
 - <WHERE <IN 800
 - .
 - <Anweisung 51
 - /
 - <Namensraumpräfix 65
 - <Operator 428
 - <SELECTION-<SCREEN <COMMENT 614
 - <SELECTION-<SCREEN <PUSHBUTTON 616
 - <SELECTION-<SCREEN <ULINE 613
 - <ULINE 701
 - <WRITE 680
 - =
 - <CALL <FUNCTION 305
 - <CALL <METHOD 293
 - <COMPUTE 427
 - <EXPORT 852
 - <FORMAT 702
 - <IMPORT 860
 - <MOVE 394
 - <UPDATE 825
 - <Verknüpfungsoperator 52
 - <WRITE 695
 - =, <>, <, >, <=, >=
 - <WHERE 798
 - =, <>, <, >, =
 - <Vergleichsoperatoren 52
 - =>
 - <Klassenkomponenten-<Selektor 56
 - >
 - <Objektkomponenten-<Selektor 54
 - >*
 - <Dereferenzierungsoperator 408
 - ><, =<, =>
 - <Vergleichsoperatoren 1075
 - ?<TO
 - <MOVE 394
 - ?=
 - <MOVE 394
 - `
 - <String-<Literal 140
 - ~
 - <Interfacekomponenten-<Selektor 57
- ## A
- ABBREVIATED
 - <SEARCH 451
 - <SEARCH – <interne <Tabelle 522
 - abs
 - <eingebaute <Funktion 132
 - ABSTRACT
 - <CLASS 184
 - <METHODS 202
 - ABSTRACT <METHODS
 - <INTERFACES 220

ACCEPT
 <COMMUNICATION 1127
 ACCEPTING <DUPLICATE <KEYS
 <INSERT 822
 ACCEPTING <PADDING
 <IMPORT 866
 ACCEPTING <TRUNCATION
 <IMPORT 868
 ACCORDING
 <ADD, <obsolet 1081
 ACCP
 <Datentyp 121
 acos
 <eingebaute <Funktion 133
 ACTIVATION
 <SET <HANDLER 322
 ACTUAL <LENGTH
 <READ <DATASET 905
 ADD
 <Anweisung 437
 <Anweisung, <obsolet 1080
 ADD-<CORRESPONDING
 <Anweisung, <obsolet 1082
 ADJACENT <DUPLICATES
 <DELETE 515
 AFTER <INPUT
 <PROCESS 559
 ALIASES
 <Anweisung 223
 ALL
 <Subquery 807
 ALL <FIELDS
 <DELETE 515
 <READ <TABLE 480
 ALL <INSTANCES
 <SET <HANDLER 322
 ALL <METHODS <ABSTRACT
 <INTERFACES 220
 ALL <METHODS <FINAL
 <INTERFACES 220
 ALL <OCCURRENCES
 <REPLACE 455
 ALLOCATE
 <COMMUNICATION 1126
 AND
 <boolescher <Operator 358
 <WHERE 804
 ANY
 <Subquery 807
 any
 <generischer <Datentyp 119
 ANY <TABLE
 <TYPES 151
 any <table
 <generischer <Datentyp 119
 APPEND
 <Anweisung 500
 APPENDING
 <FETCH 815
 <OPEN <DATASET 886
 <SELECT <INTO 794
 ARCHIVE <MODE
 <Druckparameter, <obsolet 1116
 ARCHIVE <PARAMETERS
 <NEW-<PAGE 723
 <SUBMIT 273
 AREA
 <GET <CURSOR 595
 AREA <HANDLE
 <CREATE <OBJECT 252
 AS
 <INCLUDE <STRUCTURE 169
 <INCLUDE <TYPE 169
 <SELECT 780
 <SELECT <FROM 784
 ASCENDING
 <SELECT 812
 <SORT – <Extrakt 533
 <SORT – <interne <Tabelle 516
 <SORT <BY – <Extrakt 534
 <SORT <BY – <interne <Tabelle 517
 asin
 <eingebaute <Funktion 133
 ASSERT
 <Anweisung 969
 ASSIGN
 <Anweisung 403
 ASSIGN <LOCAL <COPY <OF
 <Anweisung, <obsolet 1070
 ASSIGNED
 <logischer <Ausdruck 351
 ASSIGNING
 <APPEND 504
 <COLLECT 499
 <INSERT 497
 <LOOP <AT 487

<MODIFY 508
 <READ <TABLE 482
AT
 <Extrakt 535
 <interne <Tabelle 489
 <TRUNCATE <DATASET 914
 <ULINE 701
 <WRITE 680
AT <LINE-<SELECTION
 <Anweisung 106
AT <PF
 <Anweisung, <obsolet 1062
AT <SELECTION-<SCREEN
 <Anweisung 98
AT <USER-<COMMAND
 <Anweisung 107
atan
 <eingebaute <Funktion 133
ATTRIBUTES
 <GET <DATASET 907
 <SET <DATASET 912
AUTHORITY-<CHECK
 <Anweisung 929
AVG
 <SELECT 781

B
b
 <Datentyp 115
BACK
 <Anweisung 713
BACKGROUND <TASK
 <CALL <FUNCTION 1018
BACKUP <INTO
 <EDITOR-<CALL, <obsolet 1059
BACKWARD
 <SCROLL <LIST 739
BEFORE <OUTPUT
 <PROCESS 559
BEGIN <OF
 <DATA 160
 <DATA, <obsolet 1066
 <SELECTION-<SCREEN 608
 <TYPES 149
BETWEEN
 <DESCRIBE <DISTANCE 548
 <logischer <Ausdruck 351
 <PROVIDE 524

 <PROVIDE, <obsolet 1099
 <WHERE 799
BIG <ENDIAN
 <OPEN <DATASET 889
BINARY <MODE
 <OPEN <DATASET 887
BINARY <SEARCH
 <READ <TABLE 477
BIT-<AND, -<NOT, -<OR, -<XOR
 <Bit-<Operatoren 433
BLOCK
 <AT <SELECTION-<SCREEN 100
 <SELECTION-<SCREEN 620
BLOCKS
 <SELECTION-<SCREEN <EXCLUDE 631
 <SELECTION-<SCREEN <INCLUDE 629
BOUND
 <logischer <Ausdruck 352
BOUNDS
 <PROVIDE 523
BREAK-<POINT
 <Anweisung 972
BYPASSING <BUFFER
 <SELECT 791
BYTE <MODE
 <Bytekettenverarbeitung 442
BYTE-<CA, -<CN, -<CO, -<CS, -<NA, -<NS
 <Vergleichsoperatoren 348

C
c
 <Datentyp 115
 <generischer <Datentyp 119
CA
 <Vergleichsoperator 347
CALL <CUSTOMER-<FUNCTION
 <Anweisung 311
CALL <DIALOG
 <Anweisung, <obsolet 1072
CALL <FUNCTION
 <Anweisung 303
CALL <FUNCTION ... <DESTINATION
 <Anweisung 1011
CALL <FUNCTION ... <IN <UPDATE <TASK
 <Anweisung 312
CALL <FUNCTION ... <STARTING <NEW <TASK
 <Anweisung 1012

CALL <FUNCTION ... <IN <BACKGROUND
 <TASK
 <Anweisung 1018
 CALL <METHOD
 <Anweisung 291
 CALL <METHOD <OF
 <OLE, <obsolet 1053
 CALL <SCREEN
 <Anweisung 581
 CALL <SELECTION-<SCREEN
 <Anweisung 665
 CALL <SUBSCREEN
 <Dynpro-<Anweisung 578
 CALL <TRANSACTION
 <Anweisung 277
 CALL <TRANSFORMATION
 <Anweisung 1035
 CALLING
 <CALL <FUNCTION <STARTING <NEW
 <TASK 1015
 CASE
 <Anweisung 363
 <FIND 447
 <REPLACE 456
 <TRANSLATE 467
 CASTING
 <ASSIGN 412
 CATCH
 <Anweisung 381
 CATCH <SYSTEM-<EXCEPTIONS
 <Anweisung 385
 ceil
 <eingebaute <Funktion 132
 CENTERED
 <WRITE 683
 CHAIN
 <Dynpro-<Anweisung 571
 CHAIN-<INPUT
 <FIELD <MODULE 566
 <MODULE 563
 CHAIN-<REQUEST
 <FIELD <MODULE 566
 <MODULE 563
 CHANGING
 <CALL <FUNCTION 305
 <CALL <FUNCTION <DESTINATION 1012
 <CALL <METHOD 294
 <CLASS-<METHODS 213
 <FORM 86
 <METHODS 197
 <PERFORM 317
 <RECEIVE <RESULTS 1017
 CHAR
 <Datentyp 121
 CHAR-<TO-<HEX <MODE
 <IMPORT 870
 CHARACTER <MODE
 <Zeichenkettenverarbeitung 442
 charlen
 <eingebaute <Funktion 134
 CHECK
 <Schleife 338
 <Verarbeitungsblock 333
 CHECKBOX
 <PARAMETERS 641
 <WRITE 696
 CIRCULAR
 <SHIFT 460
 CLASS
 <Anweisung 180
 CLASS-<DATA
 <Anweisung 164
 CLASS-<EVENTS
 <Anweisung 219
 CLASS-<METHODS
 <Anweisung 212
 CLASS-<POOL
 <Anweisung 77
 class_constructor
 <statischer <Konstruktor 214
 CLEANUP
 <Anweisung 381
 CLEAR
 <Anweisung 421
 CLIENT
 <DELETE 876
 <EXPORT 856
 <IMPORT 863
 <IMPORT <DIRECTORY 874
 CLIENT <SPECIFIED
 <DELETE 833
 <INSERT 820
 <MODIFY 830
 <SELECT 790
 <UPDATE 823
 clike

<generischer <Datentyp 119
 CLNT
 <Datentyp 121
 CLOSE
 <EXEC <SQL 843
 CLOSE <DATASET
 <Anweisung 915
 CLOSE_CURSOR
 <Anweisung 816
 CN
 <Vergleichsoperator 347
 cnt
 <AT 537
 CO
 <Vergleichsoperator 346
 CODE <PAGE
 <OPEN <DATASET 889
 <TRANSLATE, <obsolete 1089
 CODEPAGE <INTO
 <IMPORT 871
 COL_...
 <FORMAT 703
 COLLECT
 <Anweisung 498
 COLOR
 <FORMAT 703
 <PRINT-<CONTROL 729
 <WRITE 695
 COLUMN
 <SCROLL <LIST 738
 <SET <LEFT <SCROLL-<BOUNDARY 716
 COMMENT
 <SELECTION-<SCREEN 614
 <SELECTION-<SCREEN <INCLUDE 628
 COMMIT <WORK
 <Anweisung 923
 COMMON <PART
 <DATA 1062
 COMMUNICATION
 <Anweisung, <obsolete 1124
 COMPARING
 <DELETE 515
 <READ <TABLE 480
 COMPONENT
 <ASSIGN 410
 COMPONENTS
 <DESCRIBE <FIELD 540
 COMPRESSION
 <EXPORT 859
 COMPUTE
 <Anweisung 427
 CONCATENATE
 <Anweisung 444
 CONDENSE
 <Anweisung 463
 CONDITION
 <ASSERT 970
 CONNECT
 <EXEC <SQL 846
 CONSTANTS
 <Anweisung 165
 constructor
 <Konstruktor 205
 CONTEXT
 <DEMAND 1104
 <SUPPLY 1103
 CONTEXTS
 <Anweisung, <obsolete 1102
 CONTINUE
 <Anweisung 337
 CONTROL
 <LOOP 573
 CONTROLS
 <Anweisung 595
 CONVERT <DATE
 <Anweisung 963
 <Anweisung, <obsolete 1093
 CONVERT <TEXT
 <Anweisung 464
 CONVERT <TIME <STAMP
 <Anweisung 961
 COPIES
 <Druckparameter, <obsolete 1116
 CORRESPONDING <FIELDS
 <SELECT 791
 cos
 <eingebaute <Funktion 133
 cosh
 <eingebaute <Funktion 133
 COUNT
 <SELECT 781
 count(*)
 <SELECT 781
 COUNTRY
 <GET <LOCALE 950
 <SET <LOCALE 945

COVER <PAGE
 <Druckparameter, <obsolet 1116
 COVER <TEXT
 <Druckparameter, <obsolet 1116
 CP
 <Vergleichsoperator 347
 CPI
 <PRINT-<CONTROL 729
 CREATE <DATA
 <Anweisung 237
 CREATE <OBJECT
 <Anweisung 247
 <OLE, <obsolet 1052
 CREATE <PRIVATE
 <CLASS 186
 CREATE <PROTECTED
 <CLASS 186
 CREATE <PUBLIC
 <CLASS 186
 CS
 <Vergleichsoperator 347
 csequence
 <generischer <Datentyp 119
 CUKY
 <Datentyp 121
 CURR
 <Datentyp 121
 CURRENCY
 <WRITE 687
 CURRENT <LINE
 <MODIFY <LINE 735
 <READ <LINE 733
 CURRENT <PAGE
 <MODIFY <LINE 735
 <READ <LINE 733
 CURRENT <POSITION
 <TRUNCATE <DATASET 914
 CURSOR
 <LOOP <WITH <CONTROL 575
 <LOOP, <obsolet 1112
 cursor
 <Datentyp 124
 CURSOR-<SELECTION
 <FIELD <MODULE 566
 <MODULE 562

D

d

 <Datentyp 115
 DATA
 <Anweisung 155
 <Anweisung, <obsolet 1063
 data
 <generischer <Datentyp 120
 DATA <BUFFER
 <EXPORT 853
 <IMPORT 862
 DATA <VALUES
 <INTERFACES 220
 DATABASE
 <DELETE 876
 <EXPORT 855
 <IMPORT 863
 <IMPORT <DIRECTORY 874
 DATASET <EXPIRATION
 <Druckparameter, <obsolet 1116
 DATE
 <CONVERT <DATE 963
 <CONVERT <DATE, <obsolet 1093
 DATS
 <Datentyp 121
 dbmaxlen
 <eingebaute <Funktion 134
 DDMMYY
 <WRITE 694
 DEALLOCATE
 <COMMUNICATION 1128
 DEC
 <Datentyp 121
 DECIMALS
 <ASSIGN <CASTING 412
 <CREATE <DATA 240
 <DATA 158
 <DESCRIBE <FIELD 543
 <PARAMETERS 637
 <TYPES 146
 <WRITE 688
 DEFAULT
 <CALL <FUNCTION <STARTING <NEW
 <TASK 1014
 <CLASS-<EVENTS 219
 <EVENTS 217
 <FIELD-<SYMBOLS 175
 <METHODS 198
 <OPEN <DATASET 888
 <PARAMETERS 645

<SELECT-<OPTIONS 660
 <SELECTION-<SCREEN <TAB 622
 DEFAULT <KEY
 <TYPES 152
 DEFERRED
 <CLASS 192
 <INTERFACE 195
 DEFINE
 <Anweisung 111
 DEFINING <DATABASE
 <REPORT 75
 DEFINITION
 <CLASS 180
 DELETE
 <Datenbanktabelle 832
 <Datenbanktabelle, <obsolet 1123
 <interne <Tabelle 509
 DELETE <DATASET
 <Anweisung 916
 DELETE <FROM
 <Anweisung 876
 DELETING <LEADING
 <SHIFT 460
 DELETING <TRAILING
 <SHIFT 460
 DEMAND
 <Anweisung, <obsolet 1104
 DEPARTMENT
 <Druckparameter, <obsolet 1116
 DESCENDING
 <SELECT 812
 <SORT – <Extrakt 533
 <SORT – <interne <Tabelle 516
 <SORT <BY – <Extrakt 534
 <SORT <BY – <interne <Tabelle 517
 DESCRIBE <DISTANCE
 <Anweisung 548
 DESCRIBE <FIELD
 <Anweisung 539
 DESCRIBE <LIST
 <Anweisung 740
 DESCRIBE <TABLE
 <Anweisung 546
 DESTINATION
 <CALL <FUNCTION 1011
 <CALL <FUNCTION <IN <BACK-
 GROUND <TASK 1018
 <CALL <FUNCTION <STARTING <NEW
 <TASK 1012
 <Druckparameter, <obsolet 1116
 DETAIL
 <Anweisung, <obsolet 1114
 DIRECTORY <ENTRY
 <INSERT <REPORT 997
 <SYNTAX-<CHECK 990
 DISCONNECT
 <EXEC <SQL 847
 DISPLAY
 <GET <CURSOR – <Liste 744, 745
 <SET <CURSOR – <Dynpro 592
 <SET <CURSOR – <Liste 749, 750
 DISPLAY <LIKE
 <MESSAGE 767
 DISPLAY-<MODE
 <EDITOR-<CALL <FOR <REPORT 1002
 <EDITOR-<CALL, <obsolet 1059
 DISTINCT
 <SELECT 778
 <SELECT – <Aggregat 781
 DIV
 <arithmetischer <Operator 429
 DIVIDE
 <Anweisung 439
 DIVIDE-<CORRESPONDING
 <Anweisung, <obsolet 1085
 DO
 <Anweisung 364
 DUMMY
 <AUTHORITY-<CHECK 929
 DURING <LINE-<SELECTION
 <TOP-<OF-<PAGE 104
 DYNAMIC <SELECTIONS
 <SELECTION-<SCREEN 632

E

EDIT <MASK
 <DESCRIBE <FIELD 545
 <WRITE 691
 EDITOR-<CALL
 <Anweisung, <obsolet 1059
 EDITOR-<CALL <FOR <REPORT
 <Anweisung 1002
 ELSE
 <Anweisung 362
 ELSEIF

<Anweisung 362
 ENCODING
 <OPEN <DATASET 887
 END <OF
 <AT – <Extrakt 536
 <AT – <interne <Tabelle 490
 <AT <SELECTION-<SCREEN 100
 <CLASS-<DATA 165
 <CONSTANTS 166
 <DATA 160
 <DATA, <obsolet 1066
 <SELECTION-<SCREEN 608
 <STATICS 167
 <TYPES 149
 END <OF <FILE
 <SET <DATASET 910
 END <OF <TASK
 <CALL <FUNCTION <STARTING <NEW
 <TASK 1015
 END-<LINES
 <DESCRIBE <LIST 742
 END-<OF-<DEFINITION
 <Anweisung 112
 END-<OF-<PAGE
 <Anweisung 104
 END-<OF-<SELECTION
 <Anweisung 96
 ENDAT
 <Anweisung – <Extrakt 536
 <Anweisung – <interne <Tabelle 490
 ENDCASE
 <Anweisung 363
 ENDCATCH
 <Anweisung 386
 ENDCHAIN
 <Dynpro-<Anweisung 571
 ENDDO
 <Anweisung 365
 ENDEXEC
 <Anweisung 839
 ENDFORM
 <Anweisung 85
 ENDFUNCTION
 <Anweisung 84
 ENDIAN
 <IMPORT 871
 <OPEN <DATASET 889
 ENDIF
 <Anweisung 362
 ENDING <AT
 <CALL <SCREEN 582
 <CALL <SELECTION-<SCREEN 665
 <SEARCH 451
 <SEARCH – <interne <Tabelle 521
 <WINDOW 750
 ENDINTERFACE
 <Anweisung 194
 ENDLOOP
 <ABAP-<Anweisung 588
 <Dynpro-<Anweisung 573
 <Extrakt 534
 <interne <Tabelle 485
 ENDMETHOD
 <Anweisung 82
 ENDMODULE
 <Anweisung 90
 ENDON
 <Anweisung, <obsolet 1076
 ENDPROVIDE
 <Anweisung 523
 ENDSELECT
 <Anweisung 775
 ENDTRY
 <Anweisung 381
 ENDWHILE
 <Anweisung 368
 EQ
 <Vergleichsoperator 342
 <WHERE 798
 error_message
 <CALL <FUNCTION 306
 ERRORMESSAGE
 <FIELD, <obsolet 1107
 ESCAPE
 <WHERE 800
 EVENT
 <CLASS-<METHODS 216
 <METHODS 207
 EVENTS
 <Anweisung 217
 EXCEPTION-<TABLE
 <CALL <FUNCTION 309
 <CALL <METHOD 301
 <CREATE <OBJECT 251
 EXCEPTIONS
 <CALL <FUNCTION 306

<CALL <FUNCTION <DESTINATION 1012
 <CALL <METHOD 295
 <CLASS-<METHODS 213
 <CREATE <OBJECT 250
 <METHODS 201
 <RECEIVE <RESULTS 1017
 EXCLUDE
 <SELECTION-<SCREEN 631
 EXCLUDING
 <GET <PF-<STATUS 584
 <SET <PF-<STATUS 583
 EXEC <SQL
 <Anweisung 839
 EXECUTE <PROCEDURE
 <EXEC <SQL 844
 EXISTS
 <Subquery 806
 EXIT
 <Schleife 337
 <Verarbeitungsblock 332
 EXIT <FROM <SQL
 <Anweisung 850
 EXIT <FROM <STEP-<LOOP
 <Anweisung 601
 EXIT-<COMMAND
 <AT <SELECTION-<SCREEN 103
 <MODULE 561
 exp
 <eingebaute <Funktion 133
 EXPONENT
 <WRITE 685
 EXPORT
 <Anweisung 851
 EXPORTING
 <CALL <DIALOG 1074
 <CALL <FUNCTION 305
 <CALL <FUNCTION <DESTINATION 1012
 <CALL <METHOD 293
 <CALL <METHOD <OF – <OLE 1054
 <CLASS-<EVENTS 219
 <CLASS-<METHODS 213
 <CREATE <OBJECT 250
 <EVENTS 217
 <METHODS 197
 <RAISE <EVENT 319
 <RAISE <EXCEPTION 380
 EXPORTING <LIST <TO <MEMORY
 <SUBMIT 271
 EXTRACT
 <Anweisung 531
F
 f
 <Datentyp 115
 FETCH
 <Anweisung 815
 <EXEC <SQL 843
 FIELD
 <AUTHORITY-<CHECK 929
 <Dynpro-<Anweisung 564
 <Dynpro-<Anweisung, <obsolet 1105
 <GET <CURSOR – <Dynpro 594
 <GET <CURSOR – <Liste 743
 <GET <PARAMETER 938
 <GET <RUN <TIME 974
 <GET <TIME 960
 <GET <TIME <STAMP 960
 <SET <CURSOR – <Dynpro 592
 <SET <CURSOR – <Liste 748
 <SET <PARAMETER 937
 FIELD <FORMAT
 <MODIFY <LINE 736
 FIELD <MODULE
 <Dynpro-<Anweisung 564
 FIELD <SELECTION
 <SELECTION-<SCREEN 632
 FIELD <VALUE
 <MODIFY <LINE 735
 <READ <LINE 733
 FIELD-<GROUPS
 <Anweisung 176
 FIELD-<SYMBOLS
 <Anweisung 174
 FIELDS
 <Anweisung, <obsolet 1069
 <ASSERT 971
 <GET <node 95
 <PROVIDE 523
 FILTER
 <OPEN <DATASET 893
 FINAL
 <CLASS 184
 <METHODS 203
 FINAL <METHODS
 <INTERFACES 220
 FIND

<Anweisung 446
 FIRST
 <AT – <Extrakt 536
 <AT – <interne <Tabelle 490
 FIRST <OCCURRENCE
 <REPLACE 455
 FIRST <PAGE
 <SCROLL <LIST 738
 FIRST-<LINE
 <DESCRIBE <LIST 742
 FIXED-<POINT <ARITHMETIC
 <INSERT <REPORT 996
 floor
 <eingebaute <Funktion 132
 FLTP
 <Datentyp 121
 FONT
 <PRINT-<CONTROL 730
 FOR
 <SELECT-<OPTIONS 655
 FOR <ALL <ENTRIES
 <WHERE 807
 FOR <ALL <INSTANCES
 <SET <HANDLER 322
 FOR <EVENT
 <CLASS-<METHODS 216
 <METHODS 207
 FOR <FIELD
 <SELECTION-<SCREEN <COMMENT
 615
 FOR <SELECT
 <OPEN <CURSOR 814
 FORM
 <Anweisung 85
 FORMAT
 <Anweisung 702
 FORWARD
 <SCROLL <LIST 739
 frac
 <eingebaute <Funktion 132
 FRAME
 <SELECTION-<SCREEN 620
 FRAMES
 <FORMAT 707
 <WRITE 695
 FREE
 <Anweisung 424
 FREE <MEMORY

 <Anweisung 877
 FREE <OBJECT
 <OLE, <obsolet 1058
 FREE <SELECTIONS
 <SUBMIT 268
 FRIENDS
 <CLASS 188
 FROM
 <DELETE – <Datenbanktabelle 834
 <DELETE – <interne <Tabelle 510
 <INSERT 820
 <LOOP 487
 <MODIFY – <Datenbanktabelle 831
 <MODIFY – <interne <Tabelle 505
 <READ <TABLE 475
 <SELECT 784
 <UPDATE 827
 FROM <TABLE
 <DELETE 835
 <INSERT 821
 <MODIFY 831
 <REFRESH 1121
 <UPDATE 828
 FUNCTION
 <Anweisung 83
 <PRINT-<CONTROL 729
 FUNCTION <KEY
 <SELECTION-<SCREEN 624
 FUNCTION-<POOL
 <Anweisung 76

G

GE
 <Vergleichsoperator 342
 <WHERE 798
 GENERATE <SUBROUTINE <POOL
 <Anweisung 981
 GET <BIT
 <Anweisung 470
 GET <CONNECTION
 <EXEC <SQL 847
 GET <CURSOR
 <Dynpro 594
 <Liste 743
 GET <DATASET
 <Anweisung 906
 GET <LOCALE
 <Anweisung 950

GET <node
 <Anweisung 95
 GET <PARAMETER
 <Anweisung 938
 GET <PF-<STATUS
 <Anweisung 584
 GET <PROPERTY
 <OLE, <obsolet 1056
 GET <REFERENCE
 <Anweisung 418
 GET <RUN <TIME
 <Anweisung 974
 GET <TIME
 <Anweisung 960
 GET <TIME <STAMP
 <Anweisung 960
 GIVING
 <ADD, <obsolet 1081
 GLOBAL <FRIENDS
 <CLASS 189
 GROUP
 <CALL <FUNCTION <STARTING <NEW
 <TASK 1014
 GROUP <BY
 <SELECT 809
 GT
 <Vergleichsoperator 342
 <WHERE 798

H

HANDLE
 <CREATE <DATA 246
 HASHED <TABLE
 <CREATE <DATA 244
 <DATA 161
 <TYPES 151
 hashed <table
 <generischer <Datentyp 120
 HAVING
 <SELECT 810
 HEAD-<LINES
 <DESCRIBE <LIST 742
 header
 <EXTRACT – <Feldgruppe 531
 <FIELD-<GROUPS 177
 <INSERT – <Feldgruppe 530
 HEADER <LINE
 <DATA 163

HELP-<ID
 <DESCRIBE <FIELD 544
 HELP-<REQUEST
 <AT <SELECTION-<SCREEN 102
 <PARAMETERS 649
 <PROCESS 559
 <SELECT-<OPTIONS 662
 HIDE
 <Anweisung 726
 HIGH
 <SET <RUN <TIME <CLOCK <RESO-
 LUTION 976
 HOLD
 <COMMUNICATION 1130
 <OPEN <CURSOR 815
 HOTSPOT
 <FORMAT 705
 <WRITE 695

I

i
 <Datentyp 115
 ICON
 <WRITE 697
 ID
 <ASSERT 970
 <AUTHORITY-<CHECK 929
 <BREAK-<POINT 972
 <COMMUNICATION 1124
 <DELETE 876
 <EXPORT 853
 <FREE <MEMORY 877
 <GET <PARAMETER 938
 <IMPORT 862
 <IMPORT <DIRECTORY 874
 <MESSAGE 763
 <SELECTION-<SCREEN 634
 <SET <PARAMETER 937
 IDS
 <SELECTION-<SCREEN <EXCLUDE 631
 IF
 <Anweisung 362
 IF <FOUND
 <INCLUDE 111
 <PERFORM 314
 IGNORING <CASE
 <FIND 447
 <REPLACE 456

IGNORING <CONVERSION <ERRORS
 <IMPORT 870
 <OPEN <DATASET 895
 IGNORING <STRUCTURE <BOUNDARIES
 <IMPORT 869
 IMMEDIATELY
 <Druckparameter, <obsolet 1116
 <SET <PF-<STATUS 746
 IMPLEMENTATION
 <CLASS 190
 IMPORT
 <Anweisung 859
 IMPORT <DIRECTORY
 <Anweisung 874
 IMPORTING
 <CALL <DIALOG 1074
 <CALL <FUNCTION 305
 <CALL <FUNCTION <DESTINATION 1012
 <CALL <METHOD 294
 <CLASS-<METHODS 213
 <METHODS 197
 <RECEIVE <RESULTS 1017
 IN
 <logischer <Ausdruck 355
 <REPLACE 453
 <Subquery 806
 <WHERE – <Selektionstabelle 801
 <WHERE – <Werteliste 800
 INCLUDE
 <Anweisung 110
 <GENERATE <SUBROUTINE <POOL 984
 <SELECTION-<SCREEN 626
 <SYNTAX-<CHECK 991
 INCLUDE <STRUCTURE
 <Anweisung 168
 INCLUDE <TYPE
 <Anweisung 168
 INCLUDING
 <CALL <SUBSCREEN 578
 INCLUDING <GAPS
 <PROVIDE 525
 INCREMENT
 <ASSIGN 409
 INDEX
 <DELETE 512
 <DESCRIBE <LIST 740
 <INSERT 497
 <MODIFY 506
 <MODIFY <LINE 735
 <READ <LINE 732
 <READ <TABLE 478
 <SCROLL <LIST 737
 <WRITE <TO, <obsolet 1097
 INDEX <TABLE
 <TYPES 151
 index <table
 <generischer <Datentyp 120
 INDEX-<LINE
 <PRINT-<CONTROL 730
 INFOTYPES
 <Anweisung, <obsolet 1067
 INHERITING <FROM
 <CLASS 183
 INIT <DESTINATION
 <COMMUNICATION 1126
 INITIAL
 <ASSIGN <LOCAL <COPY 1070
 <logischer <Ausdruck 353
 INITIAL <LINE
 <APPEND 501
 <INSERT 494
 INITIAL <SIZE
 <CREATE <DATA 244
 <DATA 161
 <TYPES 153
 INITIALIZATION
 <Anweisung 92
 INNER <JOIN
 <SELECT 785
 INOUT
 <EXEC <SQL 844
 INPUT
 <Anweisung, <obsolet 1115
 <FIELD 566
 <FORMAT 706
 <MODULE 90
 <OPEN <DATASET 885
 <WRITE 695
 INSERT
 <Datenbanktabelle 818
 <Datenbanktabelle, <obsolet 1123
 <Feldgruppe 529
 <interne <Tabelle 493
 INSERT <REPORT
 <Anweisung 992
 INSERT <TEXTPOOL

<Anweisung 999
 INT1, <INT2, <INT4
 <Datentyp 121
 INTENSIFIED
 <FORMAT 704
 <WRITE 695
 INTERFACE
 <Anweisung 193
 INTERFACE-<POOL
 <Anweisung 78
 INTERFACES
 <Anweisung 220
 INTERNAL <TABLE
 <EXPORT 854
 <IMPORT 862
 INTO
 <FETCH 815
 <INSERT 530
 <LOOP <AT 487
 <MESSAGE 769
 <READ <TABLE 480
 <SELECT 791
 INVERSE
 <FORMAT 704
 <WRITE 695
 INVERTED-<DATE
 <CONVERT, <obsolet 1093
 IS
 <logischer <Ausdruck 351
 IS <INITIAL
 <CLASS-<DATA 165
 <CONSTANTS 166
 <DATA 157
 <STATICS 167

J

JOB
 <SUBMIT 274
 JOIN
 <SELECT 785

K

KEEP <IN <SPOOL
 <Druckparameter, <obsolet 1116
 KEEPING <DIRECTORY <ENTRY
 <INSERT <REPORT 995
 KEEPING <TASK
 <RECEIVE <RESULTS 1016

KEY
 <READ <TABLE – <Datenbanktabelle,
 <obsolet 1118
 <READ <TABLE – <interne <Tabelle 477
 <READ <TABLE – <interne <Tabelle,
 <obsolet 1094

KIND
 <DESCRIBE <TABLE 546

L

LANG
 <Datentyp 121
 LANGUAGE
 <GET <LOCALE 950
 <INSERT <TEXTPOOL 999
 <READ <TEXTPOOL 998
 <SET <LOCALE 945
 LAST
 <AT – <Extrakt 536
 <AT – <interne <Tabelle 491
 LAST <PAGE
 <SCROLL <LIST 738
 LATE
 <GET <node 96
 LAYOUT
 <Druckparameter, <obsolet 1116
 LCHR
 <Datentyp 121
 LE
 <Vergleichsoperator 342
 <WHERE 798
 LEAVE
 <Anweisung, <obsolet 1075
 LEAVE <LIST-<PROCESSING
 <Anweisung 753
 LEAVE <PROGRAM
 <Anweisung 329
 LEAVE <SCREEN
 <Anweisung 604
 LEAVE <TO <LIST-<PROCESSING
 <Anweisung 752
 LEAVE <TO <SCREEN
 <Anweisung 604
 LEAVE <TO <TRANSACTION
 <Anweisung 285
 LEFT
 <MOVE <PERCENTAGE 1077
 <SCROLL <LIST 738

<SHIFT 460
 LEFT <MARGIN
 <PRINT-<CONTROL 730
 LEFT <OUTER <JOIN
 <SELECT 785
 LEFT-<JUSTIFIED
 <WRITE 683
 LEGACY <BINARY <MODE
 <OPEN <DATASET 888
 LEGACY <TEXT <MODE
 <OPEN <DATASET 890
 LENGTH
 <CLASS-<DATA 165
 <COMMUNICATION 1129
 <CONSTANTS 166
 <CREATE <DATA 239
 <DATA 158
 <DESCRIBE <FIELD 542
 <FIND 447
 <GET <CURSOR – <Dynpro 595
 <GET <CURSOR – <Liste 744, 745
 <PARAMETERS 635
 <READ <DATASET 905
 <REPLACE 454, 1092
 <STATICS 167
 <TRANSFER 899
 <TYPES 145
 LEVEL
 <PERFORM <ON 318
 LIKE
 <ASSIGN <CASTING 412
 <CLASS-<DATA 165
 <CONSTANTS 166
 <CREATE <DATA 238
 <DATA 155
 <PARAMETERS 637
 <STATICS 167
 <TYPES 145
 <Typisierung 228
 <WHERE 799
 LINE
 <DESCRIBE <LIST 741
 <GENERATE <SUBROUTINE <POOL 984
 <GET <CURSOR – <Dynpro 594
 <GET <CURSOR – <Liste 743
 <GET <CURSOR <FIELD – <Liste 744
 <MODIFY <LINE 735
 <PRINT-<CONTROL 729
 <READ <LINE 732
 <SCROLL <LIST 739
 <SELECTION-<SCREEN 618
 <SET <CURSOR – <Dynpro 592
 <SET <CURSOR – <Liste 748
 <SET <CURSOR <FIELD – <Liste 748
 <SKIP 711
 <SYNTAX-<CHECK 989
 <WRITE 699
 LINE <FORMAT
 <MODIFY <LINE 736
 LINE <OF
 <ASSIGN <LOCAL <COPY 1070
 <CREATE <DATA 242
 <DATA 159
 <TYPES 147
 <Typisierung 228
 LINE <VALUE
 <MODIFY <LINE 735
 <READ <LINE 733
 LINE-<COUNT
 <DESCRIBE <LIST 742
 <Druckparameter, <obsolet 1116
 <NEW-<PAGE 719
 <REPORT 74
 <SUBMIT 271
 LINE-<SIZE
 <DESCRIBE <LIST 742
 <Druckparameter, <obsolet 1116
 <NEW-<PAGE 720
 <REPORT 74
 <SUBMIT 271
 LINES
 <DESCRIBE <LIST 742
 <DESCRIBE <TABLE 547
 <RESERVE 725
 lines
 <eingebaute <Funktion 134
 LINES <OF
 <APPEND 502
 <INSERT 495
 LIST <AUTHORITY
 <Druckparameter, <obsolet 1116
 LIST <DATASET
 <Druckparameter, <obsolet 1116
 LIST <NAME
 <Druckparameter, <obsolet 1116
 LISTBOX

<PARAMETERS 643
 LITTLE <ENDIAN
 <OPEN <DATASET 889
 LOAD
 <CLASS 192
 <INTERFACE 195
 LOAD-<OF-<PROGRAM
 <Anweisung 91
 LOCAL
 <Anweisung, <obsolet 1079
 LOCAL <FRIENDS
 <CLASS 190
 log
 <eingebaute <Funktion 133
 log10
 <eingebaute <Funktion 133
 LOOP
 <Dynpro-<Anweisung, <obsolet 1109
 <Extrakt 534
 LOOP <AT
 <interne <Tabelle 485
 LOOP <AT <SCREEN
 <Anweisung 588
 LOOP <WITH <CONTROL
 <Dynpro-<Anweisung 573
 LOOP_AT
 <Datenbanktabelle, <_obsolet 1120
 LOW
 <SET <RUN <TIME <CLOCK <RESO-
 LUTION 976
 LOWER <CASE
 <PARAMETERS 646
 <SELECT-<OPTIONS 661
 <TRANSLATE 467
 LPI
 <PRINT-<CONTROL 729
 LRAW
 <Datentyp 121
 LT
 <Vergleichsoperator 342
 <WHERE 798

M
 M
 <Vergleichsoperator 350
 MAJOR-<ID
 <IMPORT, <obsolet 864
 MARK
 <SEARCH 452
 <SEARCH – <interne <Tabelle 522
 MATCH <LENGTH
 <FIND 448
 MATCH <OFFSET
 <FIND 448
 MATCHCODE <OBJECT
 <PARAMETERS 646
 <SELECT-<OPTIONS 661
 MAX
 <SELECT 781
 MAXIMUM
 <Anweisung, <obsolet 1087
 MAXIMUM <LENGTH
 <READ <DATASET 903
 MAXIMUM <WIDTH
 <INSERT <REPORT 993
 <READ <REPORT 988
 me
 <Selbstreferenz 124
 MEMORY
 <DELETE 876
 <EXPORT 854
 <GET <CURSOR 744, 745
 <IMPORT 863
 <SET <CURSOR 749, 750
 MEMORY <ID
 <PARAMETERS 647
 <SELECT-<OPTIONS 662
 MESSAGE
 <Anweisung 761
 <CALL <FUNCTION <DESTINATION 1012
 <CALL <FUNCTION <STARTING <NEW
 <TASK 1014
 <GENERATE <SUBROUTINE <POOL 984
 <OPEN <DATASET 894
 <RECEIVE <RESULTS 1017
 <SYNTAX-<CHECK 989
 MESSAGE-<ID
 <GENERATE <SUBROUTINE <POOL 985
 <REPORT 75
 <SYNTAX-<CHECK 991
 MESSAGES <INTO
 <CALL <TRANSACTION 283
 <DEMAND 1104
 METHOD
 <Anweisung 82
 METHODS

<Anweisung 196
 MIN
 <SELECT 781
 MINIMUM
 <Anweisung, <obsolet 1086
 MINOR-<ID
 <IMPORT, <obsolet 864
 MMDDYY
 <WRITE 694
 MOD
 <arithmetischer <Operator 429
 MODE
 <CALL <DIALOG 1073
 <CALL <TRANSACTION 281
 <INFOTYPES 1068
 MODIF <ID
 <PARAMETERS 644
 <SELECT-<OPTIONS 659
 <SELECTION-<SCREEN <COMMENT
 615
 <SELECTION-<SCREEN <PUSHBUTTON
 617
 <SELECTION-<SCREEN <ULINE 614
 MODIFIER
 <GET <LOCALE 950
 <SET <LOCALE 945
 MODIFY
 <Datenbanktabelle 829
 <Datenbanktabelle, <obsolet 1123
 <interne <Tabelle 504
 MODIFY <LINE
 <Anweisung 735
 MODIFY <SCREEN
 <Anweisung 590
 MODULE
 <ABAP-<Anweisung 90
 <Dynpro-<Anweisung 560
 <FIELD 566
 MOVE
 <Anweisung 394
 <Anweisung, <obsolet 1077
 MOVE-<CORRESPONDING
 <Anweisung 396
 MULTIPLY
 <Anweisung 438
 MULTIPLY-<CORRESPONDING
 <Anweisung, <obsolet 1084

N

n
 <Datentyp 116
 <generischer <Datentyp 120
 NA
 <Vergleichsoperator 347
 NAME
 <GENERATE <SUBROUTINE <POOL 982
 <INFOTYPES 1068
 NE
 <Vergleichsoperator 342
 <WHERE 798
 NESTING <LEVEL
 <SELECTION-<SCREEN 609
 NEW
 <AT – <Extrakt 536
 <AT – <interne <Tabelle 490
 NEW <LIST <IDENTIFICATION
 <Druckparameter, <obsolet 1116
 NEW-<LINE
 <Anweisung 712
 NEW-<PAGE
 <Anweisung 717
 <Anweisung, <obsolet 1115
 NEW-<SECTION
 <Anweisung, <obsolet 1117
 <NEW-<PAGE 722
 NEXT <CURSOR
 <FETCH 815
 NO <DATABASE <SELECTION
 <SELECT-<OPTIONS 664
 NO <DIALOG
 <NEW-<PAGE 723
 NO <END <OF <LINE
 <TRANSFER 900
 NO <FLUSH
 <CALL <METHOD <OF – <OLE 1054
 <CREATE <OBJECT – <OLE 1052
 <FREE <OBJECT – <OLE 1058
 <GET <PROPERTY <OF – <OLE 1056
 <SET <PROPERTY <OF – <OLE 1057
 NO <INTERVALS
 <SELECT-<OPTIONS 659
 <SELECTION-<SCREEN <BEGIN <OF
 <BLOCK 621
 <SELECTION-<SCREEN <BEGIN <OF
 <SCREEN 609
 NO-<DISPLAY

<PARAMETERS 640
 <SELECT-<OPTIONS 658
 NO-<EXTENSION
 <SELECT-<OPTIONS 659
 NO-<GAP
 <ULINE 701
 <WRITE 684
 NO-<GAPS
 <CONDENSE 463
 NO-<GROUPING
 <WRITE 686
 NO-<HEADING
 <NEW-<PAGE 718
 NO-<SCROLLING
 <NEW-<LINE 713
 NO-<SIGN
 <WRITE 686
 NO-<TITLE
 <NEW-<PAGE 718
 NO-<TOPOFFPAGE
 <NEW-<PAGE 720
 NO-<ZERO
 <WRITE 686
 NODE
 <PARAMETERS 649
 <SELECTION-<SCREEN 633
 NODES
 <Anweisung 171
 NON-<UNICODE
 <OPEN <DATASET 888
 NON-<UNIQUE <KEY
 <DATA 162
 <TYPES 152
 NON-<UNIQUE_KEY
 <CREATE_DATA 244
 NOT
 <boolescher <Operator 359
 <WHERE 804
 NP
 <Vergleichsoperator 348
 NS
 <Vergleichsoperator 347
 NULL
 <CLEAR 423
 <WHERE 802
 NUMBER
 <MESSAGE 763
 NUMBER <FORMAT
 <TRANSLATE, <obsolet 1090
 NUMBER <OF <PAGES
 <DESCRIBE <LIST 741
 NUMC
 <Datentyp 121
 numeric
 <generischer <Datentyp 120
 numofchar
 <eingebaute <Funktion 134
O
 O
 <Vergleichsoperator 350
 object
 <generischer <Datentyp 120
 OBJECTS
 <CALL <TRANSFORMATION 1035
 OBLIGATORY
 <PARAMETERS 640
 <SELECT-<OPTIONS 658
 OCCURS
 <DATA <BEGIN <OF, <obsolet 1066
 <DATA, <obsolet 1065
 <DESCRIBE <TABLE 547
 <INFOTYPES 1068
 <RANGES 1069
 <TYPES, <obsolet 1065
 OF
 <PERFORM 315
 OFFSET
 <FIND 447
 <GENERATE <SUBROUTINE <POOL 985
 <GET <CURSOR – <Dynpro 595
 <GET <CURSOR – <Liste 744, 745
 <REPLACE 454
 <SET <CURSOR – <Dynpro 592
 <SET <CURSOR – <Liste 748, 749
 <SYNTAX-<CHECK 991
 ON
 <AT <SELECTION-<SCREEN 99
 <FIELD 566
 <MODULE 563
 <PERFORM 318
 <SELECT 785
 ON <CHANGE <OF
 <Anweisung, <obsolet 1076
 ONLY
 <OVERLAY 466

OPEN
 <EXEC <SQL 843
 OPEN <CURSOR
 <Anweisung 814
 OPEN <DATASET
 <Anweisung 884
 OPTION
 <SELECT-<OPTIONS 660
 OPTIONAL
 <CLASS-<EVENTS 219
 <EVENTS 217
 <METHODS 198
 OPTIONS <FROM
 <CALL <TRANSACTION 282
 OR
 <boolescher <Operator 358
 <WHEN 363
 <WHERE 804
 ORDER <BY
 <SELECT 811
 OTHERS
 <CALL <FUNCTION 306
 <CALL <FUNCTION <DESTINATION 1012
 <CALL <FUNCTION <STARTING <NEW
 <TASK 1013
 <CALL <METHOD 295
 <CATCH <SYSTEM-<EXCEPTIONS 386
 <RECEIVE <RESULTS 1017
 <WHEN 363
 OUT
 <EXEC <SQL 844
 OUTPUT
 <AT <SELECTION-<SCREEN 99
 <MODULE 90
 <OPEN <DATASET 885
 OUTPUT-<LENGTH
 <DESCRIBE <FIELD 543
 OVERLAY
 <Anweisung 466

P
 p
 <Datentyp 116
 <generischer <Datentyp 120
 PACK
 <Anweisung, <obsolet 1078
 PACKAGE <SIZE
 <SELECT 794

 PAGE
 <MODIFY <LINE 735
 <READ <LINE 732
 <SCROLL <LIST 739
 PAGES
 <SCROLL <LIST 739
 PARAMETER-<TABLE
 <CALL <FUNCTION 308
 <CALL <METHOD 300
 <CREATE <OBJECT 251
 PARAMETERS
 <Anweisung 635
 <CALL <TRANSFORMATION 1035
 <NEW-<PAGE 723
 <SELECTION-<SCREEN <EXCLUDE 631
 <SELECTION-<SCREEN <INCLUDE 627
 PERCENTAGE
 <MOVE, <obsolet 1077
 PERFORM
 <Anweisung 313
 PERFORMING
 <CALL <FUNCTION <STARTING <NEW
 <TASK 1015
 <EXEC <SQL 849
 PERSON <TABLE
 <INFOTYPES 1068
 PLACES
 <SCROLL <LIST 738
 <SHIFT 458
 POSITION
 <Anweisung 714
 <GET <DATASET 906
 <OPEN <DATASET 891
 <PRINT-<CONTROL 729
 <SELECTION-<SCREEN 618
 <SET <DATASET 910
 <TRUNCATE <DATASET 914
 PREC
 <Datentyp 121
 PREFERRED <PARAMETER
 <CLASS-<METHODS 213
 <METHODS 198
 PRIMARY <KEY
 <SELECT 811
 PRINT <OFF
 <NEW-<PAGE 722
 PRINT <ON
 <NEW-<PAGE 721

<NEW-<PAGE, <obsolet 1115
 PRINT-<CONTROL
 <Anweisung 728
 PRIVATE <SECTION
 <Anweisung 181
 PROCESS
 <Dynpro-<Anweisung 558
 PROGRAM
 <Anweisung 76
 <PERFORM 314
 <SET <PF-<STATUS 583
 <SET <TITLEBAR 585
 <SYNTAX-<CHECK 990
 PROGRAM <TYPE
 <INSERT <REPORT 995
 PROTECTED <SECTION
 <Anweisung 181
 PROVIDE
 <Anweisung 522
 <Anweisung, <obsolet 1099
 PUBLIC
 <CLASS 183
 <CLASS <DEFERRED 192
 <INTERFACE 194
 PUBLIC <SECTION
 <Anweisung 181
 PUSHBUTTON
 <SELECTION-<SCREEN 616
 <SELECTION-<SCREEN <INCLUDE 629
 PUT
 <Anweisung 325

Q

QUAN
 <Datentyp 121
 QUEUE-<ONLY
 <CALL <METHOD <OF - <OLE 1054
 <CREATE <OBJECT - <OLE 1052
 <GET <PROPERTY <OF - <OLE 1056
 QUICKINFO
 <WRITE 700

R

RADIOBUTTON <GROUP
 <AT <SELECTION-<SCREEN 100
 <PARAMETERS 642
 RADIOBUTTON <GROUPS
 <SELECTION-<SCREEN <EXCLUDE 631

RAISE
 <Anweisung 388
 RAISE <EVENT
 <Anweisung 319
 RAISE <EXCEPTION
 <Anweisung 380
 RAISING
 <CLASS-<METHODS 213
 <FORM 88
 <MESSAGE 768
 <METHODS 199
 RANGE
 <ADD, <obsolet 1081
 <ASSIGN 416
 <DO 366
 <WHILE 368
 RANGE <OF
 <DATA 163
 <TYPES 154
 RANGES
 <Anweisung, <obsolet 1069
 RAW
 <Datentyp 121
 RAWSTRING
 <Datentyp 121
 READ <DATASET
 <Anweisung 901
 READ <LINE
 <Anweisung 732
 READ <REPORT
 <Anweisung 988
 READ <TABLE
 <Datenbanktabelle, <obsolet 1117
 <interne <Tabelle 474
 <interne <Tabelle, <obsolet 1094
 READ <TEXTPOOL
 <Anweisung 998
 READ-<ONLY
 <DATA 157
 RECEIVE <BUFFER
 <COMMUNICATION 1127
 RECEIVE <RESULTS
 <Anweisung 1016
 RECEIVED
 <COMMUNICATION 1129
 RECEIVER
 <Druckparameter, <obsolet 1117
 RECEIVING

- <CALL <METHOD 294
- REDEFINITION
 - <METHODS 209
- REF <TO
 - <CREATE <DATA 243
 - <DATA 159
 - <TYPES 147
 - <Typisierung 228
- REFERENCE
 - <METHODS 198
- REFERENCE <INTO
 - <APPEND 504
 - <COLLECT 499
 - <INSERT 497
 - <LOOP <AT 487
 - <MODIFY 508
 - <READ <TABLE 483
- REFRESH
 - <Anweisung 423
 - <Anweisung, <_obsolete 1121
- REFRESH <CONTROL
 - <Anweisung 601
- REJECT
 - <Anweisung 334
- RENAMING
 - <INCLUDE 169
- REPLACE
 - <Anweisung 452
 - <Anweisung, <obsolete 1091
- REPLACEMENT <CHARACTER
 - <IMPORT 870
 - <OPEN <DATASET 895
- REPLACEMENT <COUNT
 - <REPLACE 456
- REPLACEMENT <LENGTH
 - <REPLACE 456
- REPLACEMENT <OFFSET
 - <REPLACE 456
- REPORT
 - <Anweisung 72
- REQUEST
 - <FIELD 567
- REQUESTED
 - <logischer <Ausdruck 354
- RESERVE
 - <Anweisung 725
- RESET
 - <FORMAT 708
- <WRITE 695
- RESPECTING <CASE
 - <FIND 447
 - <REPLACE 456
- RESULT
 - <CALL <TRANSFORMATION 1035
- RETURN
 - <Anweisung 331
 - <SUBMIT 259
- RETURN <TO <SCREEN
 - <LEAVE <TO <LIST-<PROCESSING 753
- RETURNCODE
 - <COMMUNICATION 1128
- RETURNING
 - <CLASS-<METHODS 213
 - <METHODS 203
- RIGHT
 - <MOVE <PERCENTAGE 1077
 - <SCROLL <LIST 738
 - <SHIFT 460
- RIGHT-<JUSTIFIED
 - <WRITE 683
- ROLLBACK <WORK
 - <Anweisung 925
- ROUND
 - <WRITE 689

S

- s
 - <Datentyp 116
- SAP <COVER <PAGE
 - <Druckparameter, <obsolete 1117
- SAP-<SPOOL
 - <SUBMIT 272
 - <SUBMIT, <obsolete 1116
- SCREEN
 - <REFRESH <CONTROL 601
 - <SELECTION-<SCREEN 608
- screen
 - <LOOP <AT <SCREEN 588
 - <MODIFY <SCREEN 590
 - <Struktur 130
- SCROLL <LIST
 - <Anweisung 737
- SCROLLING
 - <NEW-<LINE 713
- SEARCH
 - <Byte- <und <Zeichenketten 448

<interne <Tabelle 520
 <READ <TABLE, <obsolet 1118
 SEARCH <PATTERN
 <PARAMETERS 650
 SECONDS
 <WAIT <UNTIL 1017
 <WAIT <UP <TO 369
 SECTION <OF
 <FIND 447
 <REPLACE 454
 SELECT
 <Anweisung 775
 <Anweisung, <obsolet 1123
 <FIELD, <obsolet 1107
 <Subquery 805
 SELECT-<OPTIONS
 <Anweisung 651
 <CHECK 333
 <SELECTION-<SCREEN <EXCLUDE 631
 <SELECTION-<SCREEN <INCLUDE 627
 SELECTION-<SCREEN
 <Anweisung 606
 <SUBMIT 261
 SELECTION-<SET
 <CALL <SELECTION-<SCREEN 666
 <SUBMIT 263
 SELECTION-<SETS
 <SUBMIT 264
 SELECTION-<TABLE
 <SUBMIT 264
 SEND <BUFFER
 <COMMUNICATION 1127
 sender
 <CLASS-<METHODS 216
 <Ereignisparameter 218
 <METHODS 208
 SEPARATE <UNIT
 <CALL <FUNCTION <IN <BACK-
 GROUND <TASK 1018
 SEPARATED <BY
 <CONCATENATE 445
 SET
 <UPDATE 824
 SET <BIT
 <Anweisung 469
 SET <BLANK <LINES
 <Anweisung 709
 SET <CONNECTION
 <EXEC <SQL 846
 SET <COUNTRY
 <Anweisung 951
 SET <CURSOR
 <Dynpro 592
 <Liste 748
 SET <DATASET
 <Anweisung 909
 SET <EXTENDED <CHECK
 <Anweisung 978
 SET <HANDLER
 <Anweisung 321
 SET <HOLD <DATA
 <Anweisung 602
 SET <LANGUAGE
 <Anweisung 944
 SET <LEFT <SCROLL-<BOUNDARY
 <Anweisung 716
 SET <LOCALE
 <Anweisung 945
 SET <MARGIN
 <Anweisung 727
 SET <PARAMETER
 <Anweisung 936
 SET <PF-<STATUS
 <Dynpro 582
 <Liste 745
 SET <PROPERTY
 <OLE, <obsolet 1057
 SET <RUN <TIME <ANALYZER
 <Anweisung 977
 SET <RUN <TIME <CLOCK <RESOLUTION
 <Anweisung 976
 SET <SCREEN
 <Anweisung 603
 SET <TITLEBAR
 <Dynpro 585
 <Liste 747
 SET <UPDATE <TASK <LOCAL
 <Anweisung 926
 SET <USER-<COMMAND
 <Anweisung 327
 SHARED <BUFFER
 <DELETE 876
 <EXPORT 858
 <IMPORT 865
 SHARED <MEMORY
 <DELETE 876

- <EXPORT 858
- <IMPORT 865
- SHARED <MEMORY <ENABLED
 - <CLASS 186
- SHIFT
 - <Anweisung 457
- SHORTDUMP-<ID
 - <GENERATE <SUBROUTINE <POOL 986
- SIGN
 - <SELECT-<OPTIONS 661
- sign
 - <eingebaute <Funktion 132
- simple
 - <generischer <Datentyp 120
- sin
 - <eingebaute <Funktion 133
- SINGLE
 - <SELECT 778
- sinh
 - <eingebaute <Funktion 133
- SIZE
 - <PRINT-<CONTROL 730
- SKIP
 - <Anweisung 710
 - <SELECTION-<SCREEN 613
- SKIP <FIRST <SCREEN
 - <CALL <DIALOG 1073
 - <CALL <TRANSACTION 279
 - <LEAVE <TO <TRANSACTION 285
- SOME
 - <Subquery 807
- SORT
 - <Extrakt 532
 - <interne <Tabelle 515
- SORTABLE <CODE
 - <CONVERT <TEXT 464
- SORTED <BY
 - <APPEND 502
- SORTED <TABLE
 - <CREATE <DATA 244
 - <DATA 161
 - <TYPES 151
- sorted <table
 - <generischer <Datentyp 120
- SOURCE
 - <CALL <TRANSFORMATION 1035
- space
 - <Konstante 124
- SPLIT
 - <Anweisung 461
- SPOOL <DYNPRO
 - <SUBMIT 273
- SPOOL <PARAMETERS
 - <SUBMIT 273
- sqrt
 - <eingebaute <Funktion 133
- SSTRING
 - <Datentyp 121
- STABLE
 - <SORT – <Extrakt 533
 - <SORT – <interne <Tabelle 516
- STANDARD <TABLE
 - <CREATE <DATA 244
 - <DATA 161
 - <TYPES 151
- standard <table
 - <generischer <Datentyp 120
- START-<OF-<SELECTION
 - <Anweisung 93
- STARTING <AT
 - <CALL <SCREEN 581
 - <CALL <SELECTION-<SCREEN 665
 - <SEARCH 451
 - <SEARCH – <interne <Tabelle 521
 - <WINDOW 750
- STARTING <NEW <TASK
 - <CALL <FUNCTION 1012
- STATICS
 - <Anweisung 167
- STATUSINFO
 - <COMMUNICATION 1127
- STOP
 - <Anweisung 335
- STRING
 - <Datentyp 121
- string
 - <Datentyp 116
- strlen
 - <eingebaute <Funktion 134
- STRUCTURE
 - <ASSIGN <COMPONENT <OF 410
 - <FIELD-<SYMBOLS 175
 - <FORM 88
 - <INCLUDE 169
- SUBKEY
 - <ASSERT 971

SUBMIT
 <Anweisung 258
 <Anweisung,<_obsolet 115
 SUBSCREEN
 <CALL 578
 <SELECTION-<SCREEN 609
 SUBSTRING
 <REPLACE 455
 SUBTRACT
 <Anweisung 438
 SUBTRACT-<CORRESPONDING
 <Anweisung, <obsolet 1084
 SUM
 <Anweisung 491
 <SELECT 781
 sum
 <AT 537
 SUMMARY
 <Anweisung, <obsolet 114
 SUMMING
 <Anweisung, <obsolet 1087
 super->
 <METHODS <constructor 205
 <METHODS <REDEFINITION 209
 SUPPLIED
 <logischer <Ausdruck 354
 SUPPLY
 <Anweisung, <obsolet 1103
 SUPPRESS <DIALOG
 <Anweisung 587
 sy
 <Systemfelder 125
 SYMBOL
 <WRITE 698
 SYNTAX-<CHECK
 <Anweisung 989
 SYST
 <Systemfelder 125

T
 t
 <Datentyp 116
 TAB
 <SELECTION-<SCREEN 621
 TABBED <BLOCK
 <SELECTION-<SCREEN 621
 TABLE
 <DELETE 510
 <INSERT 495
 <MODIFY 506
 <PARAMETERS 649
 <SELECT <INTO 794
 <SELECTION-<SCREEN 633
 <SPLIT 461
 table
 <generischer <Datentyp 120
 TABLE <FIELD
 <ASSIGN 407
 TABLE <KEY
 <DELETE 511
 <READ <TABLE 475
 TABLE <OF
 <CREATE <DATA 244
 <DATA 161
 <TYPES 150
 table_line
 <Pseudokomponente 520
 <Tabellenschlüssel 152
 TABLES
 <Anweisung 170
 <CALL <FUNCTION 305
 <CALL <FUNCTION <DESTINATION 1012
 <FORM 89
 <PERFORM 316
 <RECEIVE <RESULTS 1017
 TABLES_*
 <Anweisung,<_obsolet 1064
 TABLEVIEW
 <CONTROLS 596
 TABSTRIP
 <CONTROLS 596
 tan
 <eingebaute <Funktion 133
 tanh
 <eingebaute <Funktion 133
 TESTING
 <CLASS 187
 <METHODS 211
 TEXT
 <SORT – <Extrakt 533
 <SORT – <interne <Tabelle 516
 <SORT <BY – <Extrakt 534
 <SORT <BY – <interne <Tabelle 517
 TEXT <MODE
 <OPEN <DATASET 887
 TEXTPOOL

<INSERT 999
 <READ 998
 THEN
 <ADD, <obsolete 1081
 TIME
 <CONVERT <TIME <STAMP 961
 <GET 960
 TIME <STAMP
 <CONVERT 961
 <CONVERT <DATE 964
 <GET 960
 TIME <ZONE
 <CONVERT <DATE 964
 <CONVERT <TIME <STAMP 961
 <WRITE 690
 TIMES
 <DO 365
 TIMS
 <Datentyp 121
 TITLE
 <EDITOR-<CALL, <obsolete 1059
 <SELECTION-<SCREEN <BEGIN <OF
 <BLOCK 620
 <SELECTION-<SCREEN <BEGIN <OF
 <SCREEN 608
 TITLE-<LINES
 <DESCRIBE <LIST 742
 TO
 <LOOP 488
 TOP-<LINES
 <DESCRIBE <LIST 742
 TOP-<OF-<PAGE
 <Anweisung 103
 TRANSFER
 <Anweisung 896
 TRANSLATE
 <Anweisung 467
 <Anweisung, <obsolete 1089
 TRANSPORTING
 <MODIFY 507
 <READ <TABLE 480
 TRANSPORTING <NO <FIELDS
 <LOOP <AT 487
 <READ <TABLE 484
 trunc
 <eingebaute <Funktion 132
 TRUNCATE <DATASET
 <Anweisung 914
 TRY
 <Anweisung 381
 TYPE
 <ASSIGN <CASTING 412
 <ASSIGN, <obsolete 413
 <CLASS-<DATA 165
 <CONSTANTS 166
 <CONTROLS 596
 <CREATE <DATA 238
 <CREATE <OBJECT 249
 <DATA 155
 <DESCRIBE <FIELD 540
 <INCLUDE 169
 <MESSAGE 762
 <MESSAGE – <Text 766
 <NODES 173
 <OPEN <DATASET 892
 <PARAMETERS 637
 <RAISE <EXCEPTION 380
 <STATICS 167
 <TYPES 145
 <Typisierung 228
 TYPE <HANDLE
 <ASSIGN <CASTING 412
 TYPE-<POOL
 <Anweisung 78
 TYPE-<POOLS
 <Anweisung 143
 TYPES
 <Anweisung 144
U
 ULINE
 <Anweisung 701
 <SELECTION-<SCREEN 613
 UNASSIGN
 <Anweisung 418
 UNDER
 <WRITE 684
 UNICODE <ENABLING
 <INSERT <REPORT 996
 UNIQUE <KEY
 <DATA 162
 <TYPES 152
 UNIQUE_KEY
 <CREATE_DATA 244
 UNIT
 <Datentyp 121

- <WRITE 689
- UNPACK
 - <Anweisung 400
- UNTIL
 - <ADD, <obsolete 1081
- UP <TO
 - <SELECT 790
 - <SHIFT 459
 - <WAIT <UNTIL 1017
- UPDATE
 - <CALL <TRANSACTION 282
 - <Datenbanktabelle 822
 - <Datenbanktabelle, <obsolete 1123
 - <OPEN <DATASET 886
 - <SELECT 778
- UPDATE <TASK
 - <CALL <FUNCTION 312
- UPPER <CASE
 - <TRANSLATE 467
- USER
 - <SUBMIT 274
- USER-<COMMAND
 - <PARAMETERS <AS <CHECKBOX 641
 - <PARAMETERS <AS <LISTBOX 643
 - <PARAMETERS <RADIOBUTTON
 - <GROUP 642
 - <SELECTION-<SCREEN <PUSHBUTTON 616
 - <SELECTION-<SCREEN <TAB 622
- USING
 - <CALL <DIALOG 1073
 - <CALL <SELECTION-<SCREEN 666
 - <CALL <TRANSACTION 279
 - <FORM 86
 - <PERFORM 317
 - <TRANSLATE 468
 - <WRITE 691
- USING <SCREEN
 - <CONTROLS 596
- UTF-<8
 - <OPEN <DATASET 888

V

- VALID <BETWEEN
 - <DATA, <obsolete 1066
- VALID <FROM <TO
 - <INFOTYPES 1068
- VALUE
 - <CLASS-<EVENTS 219
 - <CONSTANTS 166
 - <DATA 156
 - <EVENTS 217
 - <FORM 87
 - <GET <CURSOR – <Dynpro 595
 - <GET <CURSOR – <Liste 744, 745
 - <METHODS 198
 - <STATICS 168
- VALUE <CHECK
 - <PARAMETERS 648
- VALUE-<REQUEST
 - <AT <SELECTION-<SCREEN 102
 - <PARAMETERS 650
 - <PROCESS 559
 - <SELECT-<OPTIONS 663
- VALUES
 - <FIELD, <obsolete 1106
 - <INSERT 820
- VARY
 - <WHILE 368
- VARYING
 - <DO 366
- VERSION
 - <DELETE, <obsolete 1123
 - <LOOP, <obsolete 1120
 - <MODIFY, <obsolete 1123
 - <READ <TABLE, <obsolete 1118
 - <SELECTION-<SCREEN 631
- VISIBLE <LENGTH
 - <PARAMETERS 641
 - <PARAMETERS <AS <LISTBOX 643
 - <SELECT-<OPTIONS 658
 - <SELECTION-<SCREEN <COMMENT 615
 - <SELECTION-<SCREEN <PUSHBUTTON 617

W

- WAIT
 - <COMMIT <WORK 923
- WAIT <UNTIL
 - <Anweisung 1017
- WAIT <UP <TO
 - <Anweisung 369
- WARNING
 - <FIELD, <obsolete 1107
- WHEN

- <Anweisung 363
- WHENEVER <FOUND
 - <FIELD, <obsolete 1107
- WHERE
 - <DELETE – <Datenbanktabelle 833
 - <DELETE – <interne <Tabelle 513
 - <LOOP 488
 - <MODIFY – <interne <Tabelle 508
 - <PROVIDE 525
 - <SELECT 797
 - <UPDATE 824
- WHILE
 - <Anweisung 368
- WINDOW
 - <Anweisung 750
 - <SELECTION-<SCREEN <BEGIN <OF 608
- WITH
 - <AT 536
 - <CLEAR 422
 - <FIELD 570
 - <MESSAGE 770
 - <SET <TITLEBAR 586
 - <SUBMIT 262
- WITH-<HEADING
 - <NEW-<PAGE 718
- WITH-<TITLE
 - <NEW-<PAGE 718
- WORD
 - <GENERATE <SUBROUTINE <POOL 985
 - <SYNTAX-<CHECK 989

- WRITE
 - <Anweisung 674
- WRITE /
 - <Anweisung 712
- WRITE <TO
 - <Anweisung 398
 - <interne <Tabelle, <obsolete 1097

X

- x
 - <Datentyp 116
 - <generischer <Datentyp 120
- XML
 - <CALL <TRANSFORMATION <RESULT 1038
 - <CALL <TRANSFORMATION <SOURCE 1037
- xsequence
 - <generischer <Datentyp 120
- xstring
 - <Datentyp 116
- xstrlen
 - <eingebaute <Funktion 134

Y

- YYMMDD
 - <WRITE 694

Z

- Z
 - <Vergleichsoperator 350