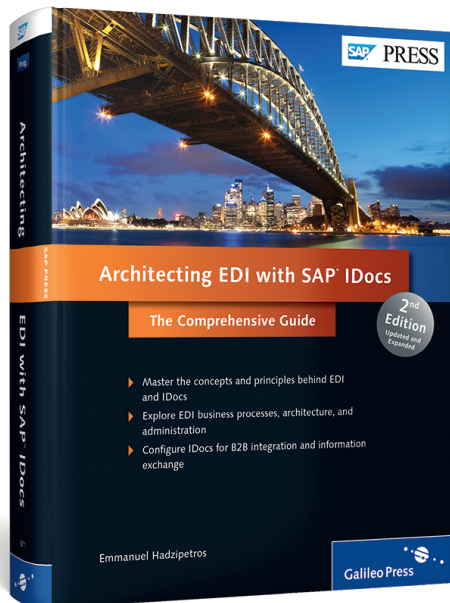


Emmanuel Hadzipetros

Architecting EDI with SAP® IDocs

The Comprehensive Guide



 Galileo Press®

Bonn • Boston

Contents at a Glance

ACT I Hollywood, DVDs, and the After-Life of Movies

1	Hollywood's B-Movie Queen Does SAP and EDI	33
2	The Blueprint: Discovery and Documentation	51
3	Designing the New SAP EDI Architecture	79

ACT II Taming Chaos with Standards—EDI in an SAP Environment

4	EDI: The Ugly Stepsister of E-Commerce	115
5	Real-World Business Process Integration with EDI	149
6	EDI Architecture in SAP: IDoc Basics	209
7	Configuring IDocs in SAP for EDI Exchange	243
8	Custom IDocs and IDoc Extensions	309

ACT III Realizing the Dream—Building Acme's SAP EDI System

9	Generating the PO for Replication Services	367
10	The Inbound Goods Receipt	395
11	Processing the Incoming Supplier Invoice	413
12	The Inbound Customer Purchase Order	441
13	Building the Outbound Order Confirmation	481
14	Sending a Shipping Order to the Supplier	547
15	The Inbound Shipping Confirmation	575
16	The Advanced Shipping Notice to the Customer	605
17	Generating the Outbound Customer Invoice	639
18	Processing the Inbound Payment Advice	703

ACT IV Finishing Touches

19	Extending the Interface: Custom IDoc Tools	739
20	Testing the EDI System in SAP	787
21	Troubleshooting and Recovery	841

Contents

Prologue	23
----------------	----

ACT I Hollywood, DVDs, and the After-Life of Movies

1 Hollywood's B-Movie Queen Does SAP and EDI 33

1.1	SAP and EDI: Getting to Know Each Other	34
1.2	A Brief History of Fame: Our Imaginary Dream Factory	35
1.2.1	A Bird's-Eye View of the Business	37
1.2.2	Enter Plan Q: A New Project Is Born	40
1.2.3	We Define Our Scope	41
1.3	Defining Some Basic Ground Rules	42
1.3.1	The Technical Environment	42
1.3.2	The EDI System	42
1.3.3	Nothing but IDocs	45
1.3.4	Batch Processing of Large Files	46
1.3.5	XML File Ports	47
1.3.6	Partner Profiles	48
1.3.7	EDI Mapping Strategy	48
1.4	Summary	50

2 The Blueprint: Discovery and Documentation 51

2.1	A Business Process Overview	52
2.1.1	The As-Is Purchasing Process Flow	52
2.1.2	The As-Is Sales and Distribution Processing Flow	57
2.1.3	Selling the Dream with Vendor Management Inventory ...	59
2.2	Legacy Systems, Data Flows, and Interfaces	63
2.2.1	Title Master and DVD Repository	65
2.2.2	Release Planning and Promotions Systems	66
2.2.3	Legacy SD	66
2.2.4	Manufacturing	67
2.2.5	Finance	67
2.2.6	VMI and StoreData	68
2.2.7	EDI	69

2.3	Legacy EDI Data Flows	69
2.3.1	Replicating Success: Outsourcing Production	70
2.3.2	Order-to-Cash and Legacy EDI	75
2.4	Summary	78

3 Designing the New SAP EDI Architecture 79

3.1	The To-Be Systems and Interfaces Emerge	80
3.1.1	The Resource Integration Manager	80
3.1.2	Let's Get Technical: A More Intimate Look at the RIM ...	83
3.2	Laying the Foundations for EDI: Master Data	87
3.2.1	A Brief Word on Conversion Strategy	87
3.2.2	General Ledger Chart of Accounts and Cost Centers	88
3.2.3	Customer Master Sold-To	88
3.2.4	Customer Master Ship-To	89
3.2.5	Vendor Master	90
3.2.6	Material Master	90
3.2.7	Customer Material Info Records	92
3.2.8	Bill of Materials	93
3.2.9	Pricing Conditions	93
3.3	The Typical Lifecycle of an Order from Gordy	94
3.3.1	VMI Sales Orders	95
3.3.2	Catalog Planning	100
3.3.3	Purchasing/Manufacturing	102
3.3.4	Delivery	104
3.3.5	Billing	106
3.3.6	Payment	108
3.4	Interfaces in the Order-to-Cash Cycle	110
3.5	Considering the Project Plan	111
3.6	Summary	112

ACT II Taming Chaos with Standards—EDI in an SAP Environment

4 EDI: The Ugly Stepsister of E-Commerce 115

4.1	A Brief History of e-Commerce	116
4.1.1	A Flowery Beginning for e-Commerce	117
4.1.2	The Berlin Airlift, the Supply Chain, and Transportation ...	117
4.1.3	The Birth of ASC X12	119

4.1.4	Global Trade and E-Commerce: UN/EDIFACT	120
4.1.5	Other EDI Standards	122
4.1.6	Communications, VANs, and the Internet	123
4.2	The Anatomy of an EDIFACT Interchange	125
4.3	The Anatomy of an X12 Interchange	128
4.3.1	Syntax and Semantics: X12 as Language	130
4.3.2	The Envelope Segments	132
4.3.3	Dissecting Gordy's 850 to Acme	134
4.4	Enveloping and De-enveloping: The X12 in Action	145
4.4.1	Unwrapping an Inbound EDI Interchange	146
4.4.2	Building an Outbound EDI Interchange	147
4.5	Summary	148

5 Real-World Business Process Integration with EDI 149

5.1	The Basic EDI Interface	150
5.2	Trading Partner Management	151
5.3	The Impact of VMI Collaboration through EDI	154
5.4	The Role of Acme's EDI RIM	155
5.4.1	Outbound Services	155
5.4.2	Inbound Services	164
5.4.3	Archiving EDI Data	169
5.5	Reporting EDI Status to SAP	171
5.5.1	SAP Status Codes	171
5.5.2	Creating Custom Messages	174
5.5.3	Mapping the STATUS IDoc	175
5.5.4	So How Do We Get the Enveloping Data?	179
5.5.5	The Status Interface Business Process Workflow	182
5.5.6	The 997 Functional Acknowledgment Interface	186
5.6	Putting All the Pieces Together	190
5.6.1	Inbound	191
5.6.2	Outbound	193
5.7	Archiving and Deleting IDocs in SAP	196
5.7.1	Archive and Delete	196
5.7.2	Deleting IDoc Generated Work Items	204
5.8	Summary	207

6	EDI Architecture in SAP: IDoc Basics	209
6.1	Intelligent Messages: The Anatomy of an IDoc	210
6.1.1	Logical Message Type	210
6.1.2	IDoc Record Types: External Representation	212
6.1.3	IDoc Record Types: Internal Representation	216
6.1.4	IDoc Basic Type	223
6.1.5	IDoc Extended Type	225
6.1.6	IDoc Instance	225
6.1.7	IDoc File	226
6.1.8	XML IDocs	226
6.2	IDoc Architecture and the Data Dictionary	230
6.2.1	Domains	231
6.2.2	Data Elements	232
6.2.3	Fields	232
6.2.4	Segments	233
6.2.5	IDoc Basic and Extended Types	234
6.3	One-Time EDI Configuration for IDocs in SAP	235
6.3.1	EDI User Name	235
6.3.2	Logical System	236
6.3.3	Connecting Systems to SAP	237
6.3.4	XML File Port	239
6.4	Summary	241
7	Configuring IDocs in SAP for EDI Exchange	243
7.1	Inbound Configuration is About Posting IDocs	243
7.1.1	Key Values for Inbound IDoc Posting	244
7.1.2	Partner Type	245
7.1.3	Inbound Partner Profile for an ORDERS PO	246
7.1.4	Inbound EDI Reference Data	252
7.1.5	Tying It All Together: The SAP EDI Inbound Processing Flow	264
7.2	Outbound Configuration Generates IDocs	274
7.2.1	Message Control	276
7.2.2	Create an Output Type	279
7.2.3	Assign Output Type to a Procedure	283
7.2.4	Condition Records	286

7.2.5	The Outbound Partner Profile with Message Control	292
7.2.6	EDI Outbound Reference Data	296
7.2.7	The SAP EDI Outbound Process Flow	300
7.3	Summary	308

8 Custom IDocs and IDoc Extensions 309

8.1	IDoc Development and Configuration Tools	309
8.1.1	Transaction SE11—Data Dictionary	310
8.1.2	Transaction WE31—Segment Editor	311
8.1.3	Transaction WE30—IDoc Type Editor	312
8.1.4	Transaction WE81—Logical Messages	313
8.1.5	Transaction WE82—Message to Basic Type Link	313
8.1.6	Transaction SE37—Function Editor: Function Groups	313
8.1.7	Transaction SE37—Function Editor: Function Modules ...	315
8.1.8	Transaction SMOD—SAP Enhancements	319
8.1.9	Transaction CMOD—Project Management for SAP Enhancements	320
8.1.10	Transaction WE57—Link Function to Message and Basic Type	321
8.1.11	Transaction BD51—Define IDoc Attributes	322
8.1.12	Transaction WE42—Inbound Process Code	322
8.1.13	Transaction WE41—Outbound Process Code	323
8.2	Building a Custom IDoc: Inbound Inventory Report	324
8.2.1	Custom IDoc Development Workflow	324
8.2.2	Building the IDoc Interface	326
8.3	Extending an IDoc: Outbound PO with BOMs	341
8.3.1	IDoc Outbound Development Workflow	341
8.3.2	Create Segment Z1EDP01	348
8.3.3	Build Extension ZORDRS01	348
8.3.4	Link Message to Basic and Extended Types	350
8.3.5	Create the Modification Project	350
8.3.6	Coding the Exit	353
8.3.7	Customize Message Control	355
8.3.8	Build Outbound Partner Profile	359
8.4	Summary	363

ACT III Realizing the Dream—Building Acme's SAP EDI System

9	Generating the PO for Replication Services	367
9.1	Technical Overview of the Interface	368
9.2	Functional Specifications	370
9.2.1	Process Overview	370
9.2.2	Requirements	371
9.2.3	Dependencies	371
9.2.4	Assumptions	372
9.2.5	Data That Will be Passed to an Outbound Purchase Order	372
9.2.6	Custom Enhancements	374
9.2.7	Reconciliation Procedure	374
9.2.8	Errors and Error Handling	374
9.3	Generating the ORDERS PO with Message Control	374
9.3.1	Message Control Configuration for the ORDERS Message	375
9.3.2	Overview of the End-to-End Process Flow	378
9.4	Technical Specifications	382
9.4.1	Technical Requirements	382
9.4.2	Dependencies	383
9.4.3	Assumptions	383
9.5	Mapping Specifications	384
9.6	EDI Configuration in SAP	388
9.6.1	EDPAR Entries: Transaction VOE4	388
9.6.2	ZEDIXREF Entries	388
9.6.3	Partner Profiles: Transaction WE20	390
9.7	Summary	392
10	The Inbound Goods Receipt	395
10.1	Technical Overview of Interface	396
10.2	Functional Specifications	398
10.2.1	Process Overview	398
10.2.2	Requirements	398
10.2.3	Dependencies	399
10.2.4	Assumptions	400
10.2.5	Data That Will Post to a Material Document	400

10.2.6	Reconciliation Procedure	401
10.2.7	Enhancements to the Process	402
10.2.8	Errors and Error Handling	402
10.3	End-to-End Process Flow	402
10.4	Technical Specifications	406
10.4.1	Technical Requirements	406
10.4.2	Dependencies	406
10.4.3	Assumptions	407
10.5	Mapping Specifications	407
10.6	EDI Configuration in SAP	409
10.6.1	EDPAR Entries: Transaction VOE4	409
10.6.2	ZEDIXREF Entries	409
10.6.3	Partner Profile: Transaction WE20	410
10.7	Summary	411

11 Processing the Incoming Supplier Invoice 413

11.1	Technical Overview of Interface	414
11.2	Functional Specifications	415
11.2.1	Process Overview	416
11.2.2	Requirements	416
11.2.3	Dependencies	417
11.2.4	Assumptions	418
11.2.5	Data That Will Post to an Inbound Supplier Invoice	419
11.2.6	Reconciliation Procedure	421
11.2.7	Configuring Logistics Invoice Verification	421
11.2.8	Enhancements to the Process	426
11.2.9	Errors and Error Handling	426
11.3	End-to-End Process Flow	426
11.4	Technical Specifications	430
11.4.1	Enhancements to the Process	430
11.4.2	Technical Requirements	430
11.4.3	Dependencies	430
11.4.4	Assumptions	431
11.5	Mapping Specifications	431
11.6	EDI Configuration in SAP	435
11.6.1	Extending Process Code INVL	435
11.6.2	EDPAR Entries: Transaction VOE4	436

11.6.3	ZEDIXREF Entries	436
11.6.4	Partner Profile: Transaction WE20	436
11.7	Summary	438

12 The Inbound Customer Purchase Order 441

12.1	Technical Overview of Interface	442
12.2	Functional Specifications	443
12.2.1	Process Overview	443
12.2.2	Requirements	444
12.2.3	Dependencies	444
12.2.4	Assumptions	445
12.2.5	Data That Will Post to an Inbound Sales Order	446
12.2.6	Reconciliation Procedure	447
12.2.7	Enhancements to the Process	447
12.2.8	Errors and Error Handling	448
12.3	End-to-End Process Flow	449
12.3.1	VMI Processing	450
12.3.2	EDI 850 Processing	452
12.3.3	VMI and EDI Processes Merge	452
12.4	Technical Specifications	455
12.4.1	Technical Requirements	455
12.4.2	Dependencies	455
12.4.3	Assumptions	456
12.5	Mapping Specifications	456
12.5.1	Structure of the 850 to IDoc Build Array	460
12.5.2	Mapping the Build Array to the Target IDoc	463
12.6	SDQ Processing Program Logic	465
12.7	Duplicate Checking Enhancement	471
12.7.1	Create Error Message	472
12.7.2	Create Modification Project	472
12.7.3	Program Flow	473
12.8	EDI Configuration in SAP	476
12.8.1	EDPAR Entries: Transaction VOE4	476
12.8.2	EDSDC Entry: Transaction VOE2	477
12.8.3	ZEDIXREF Entries	477
12.8.4	Partner Profile: Transaction WE20	478
12.9	Summary	479

13 Building the Outbound Order Confirmation	481
13.1 Technical Overview	482
13.2 Functional Specifications	483
13.2.1 Process Overview	483
13.2.2 Requirements	484
13.2.3 Dependencies	485
13.2.4 Assumptions	485
13.2.5 Data That Will Pass to an Outbound Order Confirmation	486
13.2.6 Custom Enhancements	487
13.2.7 Reconciliation Procedure	489
13.2.8 Errors and Error Handling	489
13.3 Generating the ORDRSP with Message Control	489
13.3.1 Message Control Configuration for the ORDRSP	489
13.3.2 Overview of the End-to-End Process Flow	495
13.4 Technical Specifications	499
13.4.1 Technical Requirements	499
13.4.2 Dependencies	500
13.4.3 Assumptions	500
13.4.4 Extended IDoc Type ZORSDQ01	501
13.4.5 Creating the Extended IDoc Type	501
13.4.6 SDQ Bundling and IDoc Output Program	509
13.5 Mapping Specifications	534
13.6 EDI Configuration in SAP	538
13.6.1 EDPAR Entries: Transaction VOE4	538
13.6.2 ZEDIXREF Entries	538
13.6.3 Partner Profiles: Transaction WE20	539
13.7 Summary	544
14 Sending a Shipping Order to the Supplier	547
14.1 Technical Overview of Interface	548
14.2 Functional Specifications	549
14.2.1 Process Overview	550
14.2.2 Requirements	551
14.2.3 Dependencies	552
14.2.4 Assumptions	553

14.2.5	Data That Will Pass to an Outbound Ship Order	554
14.2.6	Enhancements to the Process	556
14.2.7	Reconciliation	556
14.2.8	Errors and Error Handling	557
14.3	Generating a SHPORD IDoc with Message Control	557
14.3.1	Configuring Message Control	558
14.3.2	Overview of the End-to-End Process Flow	561
14.4	Technical Specifications	565
14.4.1	Technical Requirements	565
14.4.2	Dependencies	565
14.4.3	Assumptions	566
14.5	Mapping Specifications	567
14.6	EDI Configuration in SAP	570
14.6.1	EDPAR Entries: Transaction V0E4	570
14.6.2	ZEDIXREF Entries	570
14.6.3	Partner Profiles: Transaction WE20	571
14.7	Summary	573

15 The Inbound Shipping Confirmation 575

15.1	Technical Overview of the Interface	576
15.2	Functional Specifications	577
15.2.1	Process Overview	578
15.2.2	Requirements	578
15.2.3	Dependencies	580
15.2.4	Assumptions	580
15.2.5	Delivery Document Data after Ship Confirm Update	581
15.2.6	Enhancements to the Process	583
15.2.7	Reconciliation	583
15.2.8	Errors and Error Handling	584
15.3	End-to-End Process Flow	585
15.4	Technical Specifications	588
15.4.1	Technical Requirements	588
15.4.2	Dependencies	588
15.4.3	Assumptions	589
15.5	Mapping Specifications	590
15.5.1	Hierarchical Structure of the 856	594
15.5.2	The DELVRY03 IDoc	599
15.6	EDI Configuration in SAP	600

15.6.1	EDPAR Entries: Transaction VOE4	600
15.6.2	ZEDIXREF Entries	600
15.6.3	Partner Profiles: Transaction WE20	601
15.7	Summary	602

16 The Advanced Shipping Notice to the Customer 605

16.1	Technical Overview of Interface	606
16.2	Functional Specifications	608
16.2.1	Process Overview	608
16.2.2	Requirements	609
16.2.3	Dependencies	610
16.2.4	Assumptions	610
16.2.5	Data That Pass to the IDoc from the Delivery	611
16.2.6	Enhancements to the Process	613
16.2.7	Reconciliation	614
16.2.8	Errors and Error Handling	614
16.3	Generating an ASN IDoc with Message Control	615
16.3.1	Configuring Message Control	615
16.3.2	Overview of the End-to-End Process Flow	619
16.4	Technical Specifications	626
16.4.1	Technical Requirements	626
16.4.2	Dependencies	626
16.4.3	Assumptions	627
16.5	Mapping Specifications	627
16.6	EDI Configuration in SAP	632
16.6.1	EDPAR Entries	632
16.6.2	PUMA Entries: Transaction VNPU	633
16.6.3	ZEDIXREF Entries	634
16.6.4	Partner Profiles: Transaction WE20	634
16.7	Summary	636

17 Generating the Outbound Customer Invoice 639

17.1	Technical Overview of Interface	640
17.2	Functional Specifications	642
17.2.1	Process Overview	642
17.2.2	Requirements	643

17.2.3	Dependencies	644
17.2.4	Assumptions	645
17.2.5	Data That Pass to the IDoc from the Billing Document ...	646
17.2.6	Enhancements to the Process	648
17.2.7	Enhancement Details	648
17.2.8	Reconciliation	649
17.2.9	Errors and Error Handling	650
17.3	Generating an INVOIC IDoc with Message Control	651
17.3.1	Configuring Message Control	652
17.3.2	Overview of the End-to-End Process Flow	655
17.4	Technical Specifications	663
17.4.1	Technical Requirements	663
17.4.2	Dependencies	664
17.4.3	Assumptions	664
17.4.4	Purchase Order Number IDoc Edit Report	665
17.5	Mapping Specifications	694
17.6	EDI Configuration in SAP	697
17.6.1	EDPAR Entries: Transaction VOE4	697
17.6.2	ZEDIXREF Entries	697
17.6.3	Outbound Partner Profile: Transaction WE20	698
17.7	Summary	700

18 Processing the Inbound Payment Advice 703

18.1	Technical Overview of the Interface	704
18.2	Functional Specifications	706
18.2.1	Process Overview	706
18.2.2	Requirements	707
18.2.3	Dependencies	708
18.2.4	Assumptions	709
18.2.5	Payment Advice Note Data	709
18.2.6	Enhancements to the Process	711
18.2.7	Reconciliation	712
18.2.8	Errors and Error Handling	712
18.3	End-to-End Process Flow	713
18.3.1	Following the 820 Flow	714
18.3.2	An Intelligent Split	714
18.3.3	Posting the Payment	715

18.4	Technical Specifications	717
18.4.1	Technical Requirements	717
18.4.2	Dependencies	717
18.4.3	Assumptions	718
18.4.4	EDI Process to Split Very Large 820s	719
18.4.5	User Transaction Code for the SAP Split Program	724
18.5	Mapping Specifications	725
18.6	EDI Configuration in SAP	731
18.6.1	EDPAR Entries	731
18.6.2	Company Code to Sold-to Party Transaction OCBA	731
18.6.3	Reason Code Conversion	732
18.6.4	ZEDIXREF Entries	733
18.6.5	Inbound Partner Profile Transaction WE20	734
18.7	Summary	735

ACT IV Finishing Touches

19 Extending the Interface: Custom IDoc Tools 739

19.1	EDI to IDoc Trading Partner Conversion	740
19.1.1	The Issue	740
19.1.2	The Solution	741
19.1.3	Development Work Flow	741
19.1.4	Writing the Code	746
19.2	Mass Upload of Partner Profiles to SAP	749
19.2.1	The Issue	750
19.2.2	The Solution	751
19.2.3	Dependencies	753
19.2.4	Coding ZEDI_UPLDPP	754
19.2.5	Further Automating Partner Profile Processing	762
19.3	Mass Transfer of IDocs between Systems	764
19.3.1	The Issue	764
19.3.2	The Solution	764
19.3.3	Dependencies	765
19.3.4	Program ZEDI_TRNSFIDOCs	765
19.4	Sending IDoc Status to an External System	772
19.4.1	The Issue	772
19.4.2	The Solution	772
19.4.3	Dependencies	772

19.4.4	Defining the Distribution Model	773
19.5	Adding Qualifiers to IDoc XML Schema	778
19.5.1	The Issue	778
19.5.2	The Solution	778
19.5.3	Dependencies	778
19.5.4	Extracting an XML Schema from SAP	779
19.5.5	Creating the IDoc Enumerators	781
19.5.6	Using the Enumerator in an IDoc	784
19.6	Summary	786

20 Testing the EDI System in SAP 787

20.1	IDoc Test Tools	788
20.1.1	The Main IDoc Test Tool: Transaction WE19	788
20.1.2	Outbound from Message Control: Transaction WE15	791
20.1.3	Outbound from IDoc: Transaction WE14	792
20.1.4	Status File Testing: Transactions WE18 and WE17	793
20.1.5	Turnaround Utility: Transaction WE12	794
20.1.6	Inbound IDoc File Processing: Transaction WE16	795
20.1.7	Inbound IDocs and startRFC	796
20.2	Interface Testing Strategy	798
20.2.1	Testing Environments	799
20.2.2	Break-Fix Procedures	802
20.2.3	Test Teams and Responsibilities	803
20.2.4	Documenting Tests	805
20.3	Unit Testing	805
20.3.1	Scope	806
20.3.2	Criteria for Success	806
20.3.3	Dependencies	807
20.3.4	Execution	807
20.4	String Testing	808
20.4.1	Scope	808
20.4.2	Criteria for Success	809
20.4.3	Dependencies	810
20.4.4	Execution	810
20.5	Interface Testing	815
20.5.1	Scope	815
20.5.2	Criteria for Success	816

20.5.3	Dependencies	817
20.5.4	Execution	818
20.6	Integration Testing	822
20.6.1	Scope	823
20.6.2	Criteria for Success	824
20.6.3	Dependencies	825
20.6.4	Execution	826
20.7	Stress Testing	835
20.7.1	Scope	836
20.7.2	Criteria for Success	837
20.7.3	Dependencies	837
20.7.4	Execution	838
20.8	Summary	839
21 Troubleshooting and Recovery		841
21.1	Identifying Issues	842
21.1.1	Defining Success	842
21.1.2	Defining Failure	844
21.1.3	Functional or Business Failure	848
21.2	Monitoring and Recovery Tools	850
21.2.1	Transaction BD87: The Status Monitor	851
21.2.2	Transaction WE05: The IDoc List	867
21.2.3	Processing Log for Output Control	869
21.2.4	Transaction WE07: Errors History	870
21.2.5	Transaction WE09: IDoc Search by Business Content	871
21.2.6	Transaction WE08: IDoc Interruptus	873
21.3	Recovering from Key Failure Points	873
21.3.1	Technical Troubleshooting	874
21.3.2	Functional Troubleshooting	881
21.4	Summary	888
Epilogue		889
The Author		893
Index		895

Prologue

Let's return to Hollywood and once again take the studio tour of Acme Pictures and revisit its legendary founder, Darryl Q. Fernhausen.

Since the first edition of this book was published, we've had time to reconsider Acme's implementation of an Electronic Data Interchange (EDI) system in an SAP environment. While the first edition covered a lot of ground, it left out even more. And, frankly, we made some mistakes, not all of which were typos.

This second edition is our attempt to plug some of these gaps. While much of the original remains, we've tried to put more emphasis on the business context and have added new interfaces from the purchasing cycle.

We have consolidated and expanded our discussions of message control and added a number of new custom tools and utilities. We have changed our IDoc format from ASCII to XML and introduced some basic concepts of working with, and extending, XML schema.

All in all, we hope that the result is a tighter book that covers more subjects in a clearer manner. But this book is still about SAP and EDI. And the basics haven't changed.

EDI is still the most widely used form of electronic commerce in the world today. It is highly unlikely that this will change anytime soon. EDI has been around for decades. It is reliable, proven, stable, and supported by long-established standards developed and maintained by such global bodies as the United Nations, the International Organization for Standardization (ISO), GS1, and the American National Standards Institute (ANSI).

If money talks, EDI has the eloquence of a Shakespearean actor. It supports trillions of dollars in transactions every year in a wide variety of industries. Many companies will not buy goods or services from suppliers that cannot exchange standard business documents through EDI.

In the United States, Wal-Mart, the world's largest retailer, is at the heart of a gigantic global EDI ecosystem with thousands of suppliers that rivals most governments

in its use of information and communications technology. If you want to sell to Wal-Mart, you can do it only through EDI.

The importance of EDI as an enterprise integration application for thousands of SAP customers is beyond dispute. EDI, and the large-volume batch processing of business transactions that it enables, is a key element of the SAP environment in thousands of locations around the world. Where EDI is present, most of the transactional data that flow between SAP and external trading partners are carried by EDI.

With more than 70 percent of the global business software market, SAP is the business system of record that a majority of EDI consultants, developers, and production support teams work with every day.

Together, SAP and EDI are the heart, bones, arteries, and brains of modern business and government organizations.

So why are we touring an imaginary Hollywood studio? Look beyond the glitz and the glitter and it is just another business. Acme Pictures is a good model for the challenges faced by many businesses when they implement SAP and EDI together.

Acme Pictures sells movies on DVD, a consumer product sold through retail, not all that different from other products that wind up on store shelves around the world. We could just as easily be talking about video games, pharmaceuticals, carpet, shoes, software, beverages, snack foods, or the book that I hope you are now enjoying.

For Acme Pictures, it's a simple equation: The volume of business it does with large retail chains across North America, and the huge number of documents that this business generates, can only be managed through EDI.

The other thing, of course, is that Hollywood is a lot of fun. But so is SAP and EDI, even if the details can get a little dry at times. I'll never forget the advice of a project manager during kick-off for an SAP implementation at a steel mill in Ohio: "Work hard and have lots of fun!"

The fun is in the creative process of designing and building a system that fits the client's business. It's discovering how far you can push the limits of your knowledge to provide your client with a useful system that will support its business for years to come.

The Book and Its Audience

Audiences are the lifeblood of Hollywood. Books, too. And while this book will never be made into a movie, it may provide some useful information to SAP and EDI consultants, developers, managers, and anyone else implementing, supporting, or considering EDI in an SAP environment.

This book is the culmination of my 20-year odyssey as an SAP consultant and developer with a perennial fascination for data flows and integration. It represents ideas about integration architecture considered throughout my SAP career, which includes project work in four countries, three continents, and such industries as beverages, electrical utilities, steel, electronics, textiles, pharmaceuticals, and entertainment.

We will take the studio tour of an SAP EDI implementation project, lovingly referred to as Plan Q from Outer Space, or just plain Plan Q—in honor of our imaginary studio's most famous film.

This project-based approach attempts to deal in a holistic manner with the entire SAP EDI ecosystem at Acme Pictures, beginning with an overview of the business it supports, just enough to discuss technical solutions. The real-world business of a Hollywood studio is far more complex and nuanced than we could describe in these pages.

This book is not an SAP or EDI programming guide. It does assume an ability to follow program logic and visualize end-to-end systems. We rely on standards and standard functionality wherever possible.

Where potential solutions to common problems are presented, we will step through the process flow and logic. But we will not write the code for you. Consider our logic as starting points for your own creative exploration. And forgive me in advance if some of our ideas don't add up for you.

Our real hope is that this book helps you formulate questions that you may not have considered. After all, every business, no matter how big or how small, is as unique as the people who run it.

Structure of the Book

The book loosely follows the phases of our imaginary SAP project with detours for background information about IDocs and EDI.

In deference to our Hollywood theme, the book is organized into four acts, which are meant to build your knowledge of Acme's business, its systems, and SAP IDoc and EDI development. A summary of each chapter follows.

Act I—Hollywood, DVDs, and the After Life of Movies

▶ **Chapter 1—Hollywood's B-Movie Queen Does SAP and EDI**

This chapter introduces Acme Pictures, its visionary founder, Darryl Q. Fernhausen, and its unique approach to the movie business. We also touch on SAP EDI development strategy.

▶ **Chapter 2—The Blueprint: Discovery and Documentation**

In this chapter we document Acme's DVD business and the legacy systems that support it. It also provides an overview of key customer and vendor processes, including purchasing and order-to-cash.

▶ **Chapter 3—Designing the New SAP EDI Architecture**

We present the vision for the new system. We'll introduce to-be systems and interfaces, and the Resource Integration Manager (RIM). We also examine the key business processing cycles enabled by EDI.

Act II—Taming Chaos with Standards: EDI in an SAP Environment

▶ **Chapter 4—EDI: The Ugly Stepsister of E-Commerce**

This chapter covers EDI for SAP professionals, including a brief tour of its fascinating history and introduce the major EDI standards, with special emphasis on EDIFACT and ANSI X12.

▶ **Chapter 5—Real-World Business Process Integration with EDI**

Relationships and integrating processes between trading partners are at the heart of EDI. In this chapter we look at the role of Acme's EDI RIM, with its adapters and services, and its connections to SAP.

▶ **Chapter 6—EDI Architecture in SAP: IDoc Basics**

In this chapter we discuss Intermediate Documents (IDocs)—the intelligent messages defined by the Data Dictionary and the underlying logic that determines how they are used.

▶ **Chapter 7—Configuring IDocs in SAP for EDI Exchange**

From partner profiles to message control and mapping tables, inbound and outbound IDoc configuration and processing flows in SAP are the focus of this chapter.

▶ **Chapter 8—Custom IDocs and IDoc Extensions**

An introduction to IDoc development in SAP. We go over development tools and process flows and build, code, and configure one custom and one extended IDoc.

Act III—Realizing the Dream: Building Acme's SAP EDI System

▶ **Chapter 9—Generating the PO for Replication Services**

This chapter defines the function and technical setup for the outbound purchase order for replication services to Acme's contract manufacturer, including configuration of message control to output an ORDERS IDoc.

▶ **Chapter 10—The Inbound Goods Receipt**

In this chapter we discuss the functionality and configuration of inbound inventory adjustments and goods receipt through an EDI X12 867 transaction.

▶ **Chapter 11—Processing the Inbound Supplier Invoice**

We review posting requirements for the inbound supplier invoice for contract manufacturing services, including invoice verification and configuration.

▶ **Chapter 12—The Inbound Customer Purchase Order**

This chapter examines the inbound X12 850 to ORDERS IDoc customer purchase order, including logic for SDQ processing and code to block posting of duplicate POs to SAP Sales Orders.

▶ **Chapter 13—Building the Outbound Order Confirmation**

Covers the generation of the outbound ORDRSP confirmation from an SAP sales order. We'll build an extended IDoc and look at logic for a custom program to bundle multiple sales orders from the same SDQ PO into a single X12 855 interchange.

▶ **Chapter 14—Sending a Shipping Order to the Supplier**

We discuss the outbound SHPORD to X12 830 shipping order to the supplier, including message control configuration enabling output of IDocs from the SAP delivery document.

▶ **Chapter 15—The Inbound Shipping Confirmation**

This chapter focuses on the X12 856 shipping confirmation from the vendor, which updates pick quantity and posts goods issue in the delivery document in Acme's SAP system after the order ships.

▶ **Chapter 16—The Advanced Shipping Notice to the Customer**

In this chapter we focus on the advanced ship notice (ASN), which tells the

customer what to expect in its shipment. We will emphasize the critical business requirement for accuracy and timeliness and discuss conditions for creating the DESADV IDoc.

▶ **Chapter 17—Generating the Outbound Customer Invoice**

Here we cover the INVOIC to X12 810 customer invoice, generated from the SAP billing document. We detail output requirements for the IDoc and step through a custom ALV grid program for changing the PO number in the IDoc.

▶ **Chapter 18—Processing the Inbound Payment Advice**

The focus of this chapter is the inbound payment advice, which records details of a customer payment on all invoices, including debits and credits. We also discuss common issues with very large X12 820 files.

Act IV—Finishing Touches

▶ **Chapter 19—Extending the Interface: Custom IDoc Tools**

Fun with ABAP, ALE, and XML as we look at custom utilities that take advantage of standard SAP functionality.

▶ **Chapter 20—Testing the EDI System in SAP**

Acme's testing strategy is the focus of this chapter. We examine the composition and role of the test team and outline the key test phases.

▶ **Chapter 21—Troubleshooting and Recovery**

Defining success and failure in Acme's SAP EDI architecture. We'll look at standard monitoring tools and consider situations that appear successful but could lead to errors in later stages of the EDI cycle.

▶ **Epilogue**

With the project complete, the integration team relaxes at a famous Hollywood watering hole to toast the successful release of an Acme *film noir* classic and the success of the new SAP EDI system.

Acknowledgments

If no man is an island, as the English poet and preacher John Donne once observed, the same is especially true for authors, who often labor in solitude and obscurity. During the long, lonely hours spent writing a book, it is easy to forget the many people who enrich our lives and our work every day.

The quest to define, begin, complete, and rewrite this book was a labor of love that extended over many years. It would have been a mission impossible without the

people—family, friends, and colleagues—who sometimes endured yawn-inducing dissertations of SAP integration issues.

First and foremost, this is dedicated to my son and two daughters and especially to the grandchildren that I've been blessed with since publication of the first edition. The future belongs to you.

To all my family and friends, wherever in or out of this crazy, beautiful world you may find yourselves: thank you for being you.

A good programmer never stops learning. A good consultant learns so that he can pass on his knowledge to benefit his client. A passion to learn and to acquire and pass on new skills is the key to success and to having fun in this business. Learning means working with other people—listening, studying, discussing, playing, and poking around systems.

I've been lucky in the colleagues that I've worked with over the years. I've learned so much from so many people that it's impossible to remember them all. So thanks to everybody. You know who you are. It's been a privilege working with you.

I also want to thank the folks at SAP PRESS for their support and continuing belief in this project.

Finally, I want to express a heartfelt thank you to the countless numbers of extraordinary working people in Hollywood who labor quietly every day in the offices and back lots of the studios beyond the glare of the klieg lights.

When the director shouts "Lights...cameras...action!" they turn on the lights, run the cameras, build the sets, feed the crews, park the star wagons, clean up, and run the IT systems that keep the business humming.

They are, always have been, and always will be the real stars of Hollywood.

"I couldn't build my way out of a paper bag," Darryl Q would tell the carpenters who built his cheesy sets. Acme's legendary founder knew that the success of his films depended on his workers' ability to build. Just as the success of the new SAP EDI system, and the business that relies on it, rests on the ability of the team to build and extend IDocs. So let's delve into this fascinating topic and go over the tools and techniques we use to craft our own custom IDocs.

8 Custom IDocs and IDoc Extensions

Now we're getting to the interesting part: creating, coding, and configuring custom IDocs and extending standard IDocs.

Our custom IDoc will post an X12 846 inventory report to a custom table. Acme will use the data to write custom reports to support inventory balancing.

We'll build our extended IDoc from basic type ORDERS05 for an outbound supplier purchase order using message type ORDERS. We'll add a BOM segment just beneath the E1EDP01 item parent to send the bill of materials associated with a purchase order to Acme's third-party manufacturer, Disc Services International, in the outbound 850 supplier purchase order.

Let's begin by taking a quick tour of the tools that we'll use to develop and configure our IDoc interfaces.

8.1 IDoc Development and Configuration Tools

Our starting point for the development of custom IDocs and extensions is the EDI area menu. You can access it with Transaction WEDI from the SAP EASY ACCESS main menu. WEDI is an area menu and cannot be accessed from other transaction screens.

The key IDoc development tools that we'll be using for our customization are clustered in the DEVELOPMENT folder, illustrated in Figure 8.1.

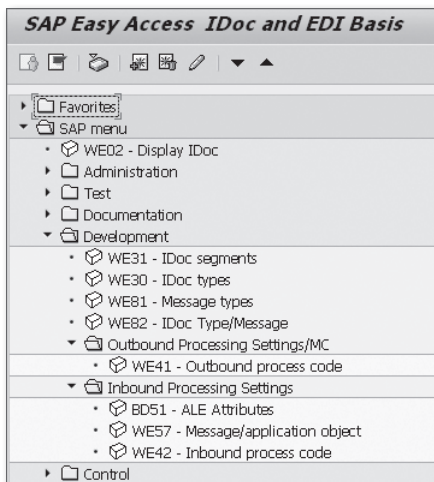


Figure 8.1 IDoc Development Tools in the EDI Area Menu

There are other development tools that we'll be using in addition to those in the EDI area menu. We'll look each tool in the order of use in our development workflow.

8.1.1 Transaction SE11—Data Dictionary

Get to the Data Dictionary with Transaction SE11 through the REPOSITORY INFORMATION SYSTEM (Transaction SE90) in the ABAP DICTIONARY folder, or through SAP menu TOOLS • ABAP WORKBENCH • DEVELOPMENT • ABAP DICTIONARY.

We've touched on the role of the Data Dictionary in defining IDoc syntax and architecture (see Chapter 6, Section 6.2, IDoc Architecture and the Data Dictionary). We'll use the Data Dictionary to create the following objects:

- ▶ Domains: Tables DD01L and DD01T.
- ▶ Data elements: Tables DD04L and DD04T. Domains and data elements will be used for custom fields in custom segments.
- ▶ Structures and transparent tables to store master and transactional data for use in custom programs: Tables DD02L and DD02T.

We'll also use the Data Browser to look at data stored in transparent tables and to do informal extracts for analysis. The Data Browser can be reached with Transaction SE16.

8.1.2 Transaction WE31—Segment Editor

We'll use the segment editor to build and edit segments for custom and extended IDoc basic types. The segments are saved as structures in the Data Dictionary in tables DD02L and DD02T.

As shown in Figure 8.2, the segment editor keeps track of all external names and versions of the segment, along with its string lengths, the number of fields it contains, and the SAP release number current when it was last changed.

Development segments: Initial screen

Segment type: E1EDK01
IDoc: Document header general data

Definitions

Ver...	Segm. definition	Rele...	Rele...	App...	No. ...	Lgth	Date of last c...	Time ...
005	E2EDK01005	<input checked="" type="checkbox"/>	45B		32	357	03/15/1999	15:07:47
004	E2EDK01004	<input checked="" type="checkbox"/>	45A		31	345	03/27/1998	09:21:13
003	E2EDK01003	<input checked="" type="checkbox"/>	40A		29	338	08/11/1997	13:19:29
002	E2EDK01002	<input checked="" type="checkbox"/>	30F		26	291	02/03/1997	12:40:32
001	E2EDK01001	<input checked="" type="checkbox"/>	30C		19	222	01/15/1996	15:31:10
000	E2EDK01	<input checked="" type="checkbox"/>	30A		18	212	01/15/1996	15:31:09

Figure 8.2 Segment Versions in the Initial Screen of the Segment Editor

Double-click any segment definition row to get to its fields, as illustrated in Figure 8.3 for segment type E1EDK01.

Segment type attributes

Segment type: E1EDK01 Qualified segment
Short Description: IDoc: Document header general data

Segm. definition: E2EDK01005 Released
Last Changed By: SAP

P...	Field Name	Data element	ISO...	E...
1	ACTION	EDI1225_A	<input type="checkbox"/>	3
2	KZABS	EDI_KZABS	<input type="checkbox"/>	1
3	CURCY	EDI6345_A	<input type="checkbox"/>	3
4	HWAER	EDI_HWAER	<input type="checkbox"/>	3
5	WKURS	EDI5402_A	<input type="checkbox"/>	12
6	ZTERM	EDI4297_A	<input type="checkbox"/>	17
7	KUNDEUINR	STCEG	<input type="checkbox"/>	20
8	EIGENUINR	STCEG	<input type="checkbox"/>	20
9	BSART	EDI_BSART	<input type="checkbox"/>	4
10	BELNR	EDI_BELNR	<input type="checkbox"/>	35

Figure 8.3 Field List in Segment Editor

Fields are added to this screen during the creation of segments, and data elements are assigned to the fields. The data element is linked to a domain. Double-clicking on the data element name opens the `DISPLAY DATA ELEMENT` screen. Double-clicking on the domain name opens the `DISPLAY DOMAIN` screen. If there are qualifiers associated with the domain in a value range, you can click on the `VALUE RANGE` tab to see them.

The segment is activated after it has been released in the initial screen by selecting menu option `EDIT • SET RELEASE`. The segment must be released before it can be used in a basic type.

To get to the segment editor, open the `DEVELOPMENT` folder in the `WEDI` area menu, and double-click `IDOC SEGMENTS`, or use Transaction `WE31`, `SAP` menu `TOOLS • ALE • ALE DEVELOPMENT • IDOC • IDOC TYPE DEVELOPMENT • SEGMENTS`.

8.1.3 Transaction WE30—IDoc Type Editor

We use the type editor to assemble segments into custom or extended IDoc basic types. It's also a great way to display the structure and segment attributes of standard IDoc basic types such as `ORDERS05`.

Double-click on any segment name to open the `ATTRIBUTE DISPLAY` dialog. It records key parameter values for the segment that controls its place within the IDoc basic type, including the following:

- ▶ Segment type name
- ▶ Mandatory segment flag
- ▶ Minimum and maximum number of occurrences
- ▶ Parent segment number
- ▶ Hierarchy level

Click the `SEGMENT EDITOR` button to open the `FIELD DISPLAY` screen of the segment editor.

The custom or extended IDoc is released after it's been assembled in the initial screen of the type editor with menu option `EDIT • SET RELEASE`. It must be activated before it can be used in an interface or transported to other SAP clients.

You can get to the type editor by opening the `DEVELOPMENT` folder in the `WEDI` area menu, and double-clicking `IDOC TYPES`, or by using either Transaction `WE30`

or SAP menu **TOOLS • ALE • ALE DEVELOPMENT • IDOC • IDOC TYPE DEVELOPMENT • IDOC TYPES**.

IDoc basic and extended types are stored in table IDOCSYN.

8.1.4 Transaction WE81—Logical Messages

Create logical message types when building custom IDoc in the message type editor. To get there, double-click **LOGICAL MESSAGES** in the **DEVELOPMENT** folder in the **WEDI** area menu, use Transaction **WE81**, or follow menu path **TOOLS • ALE • ALE DEVELOPMENT • IDOC • IDOC TYPE DEVELOPMENT • LOGICAL MESSAGES**.

Logical message types are stored in tables **EDMSG** and **EDIMSGT**.

8.1.5 Transaction WE82—Message to Basic Type Link

This links the message type to the IDoc basic type, providing structure to the logical message. Multiple message types can be linked to one basic type. Transaction **WE82** is also used to link IDoc extensions to messages and basic types. Message, basic, and extended types are linked in table **EDIMSG**.

The relevant menu paths are **IDOC TYPE/MESSAGE** in the **DEVELOPMENT** folder of the **WEDI** area menu or **TOOLS • ALE • ALE DEVELOPMENT • IDOC • IDOC TYPE DEVELOPMENT • IDOC TYPE FOR MESSAGE**.

8.1.6 Transaction SE37—Function Editor: Function Groups

All function modules are created within a function group. Create function groups in the **PROGRAM LIBRARY** folder in the **Repository Information System** (Transaction **SE80**). Click the **EDIT OBJECT** button and navigate to the **FUNCTION GROUP** tab, as illustrated in Figure 8.4.

You can also use Transaction **SE37**, SAP menu path **TOOLS • ABAP WORKBENCH • DEVELOPMENT • FUNCTION BUILDER**, or menu path **GoTo • FUNCTION GROUPS • CREATE GROUP**.

Function groups are programs that logically group related function modules into a common package with global data types, declarations, constants, and so on. The naming convention for the function pool program generated for the function group is always **SAPL<FUNCGRP>**, where **FUNCGRP** is the name of the function group.

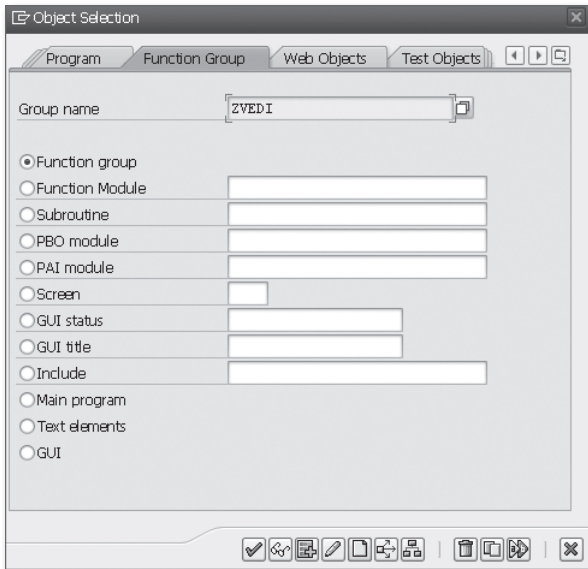


Figure 8.4 Creating a Function Group in the Repository Info System

For example, function group E1NM (Figure 8.5), which includes function IDOC_OUTPUT_ORDERS, has the program name SAPLE1NM.

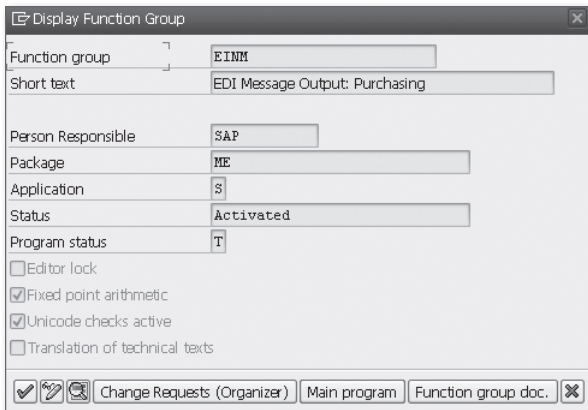


Figure 8.5 Attributes for Function Group E1NM

The main function group program always contains the following includes:

- ▶ L<FUNCGRP>TOP
Global data declarations for the function group.

- ▶ L<FUNCGRP>UXX
Stores includes with the function modules.
- ▶ L<FUNCGRP>FOX
Form subroutines called by the functions. The X character can be a number from 0 to N or a letter to distinguish multiple include programs.

There could also be other includes that do not follow this naming convention with form routines that perform special functions.

We'll create one function group per function module for Acme's custom IDoc development.

8.1.7 Transaction SE37—Function Editor: Function Modules

This is where the code hits the road. Most of our programming in the IDoc interface will be in the Function Builder.

Function modules are created using the same transactions and menu paths as function groups.

Functions are encapsulated programs that perform one function. They have a standard interface with import and export parameters and can transfer internal tables for processing at runtime. They also return error codes that can be trapped for error message processing.

The great bulk of the work of the IDoc interface is done with functions, even with the trend toward object-oriented programming in SAP. Functions are so pervasive throughout the interface that it's highly unlikely they'll be replaced any time soon, although method calls are increasingly being used and exits, in particular, are mirrored by BADIs.

For example, functions are used to build partner profiles within WE20. This is a three-step process implemented by four function calls that populate the partner profile tables:

1. Build the general view:
 - ▶ EDI_AGREE_PARTNER_INSERT: Table EDPP1
2. Build outbound parameters and message control:
 - ▶ EDI_AGREE_OUT_MESSTYPE_INSERT: Table EDP13

- ▶ EDI_AGREE_OUT_IDOC_INSERT: Table EDP12

3. Build inbound parameters:

- ▶ EDI_AGREE_IN_MESSTYPE_INSERT: Table EDP21

Functions are also used as customer exits for IDocs and other standard SAP applications. Customer exits allow the user to extend the functionality of standard IDoc functions to accommodate unique business requirements.

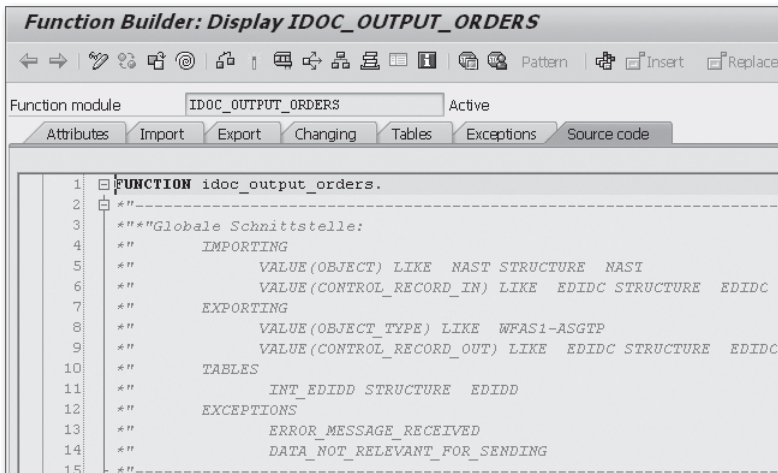
IDocs are mostly processed through function modules. Workflow tasks can be used to process some IDocs but we won't be doing this in our Acme implementation except for the standard task that processes the STATUS message type. Our main concern here is with the IDOC_INPUT and OUTPUT functions, which follow the naming conventions:

- ▶ IDOC_INPUT_<MESSAGE>
The application programming interface (API) for inbound functions is described in the upcoming subsection "API for Inbound IDoc Processing Functions." An example is function IDOC_INPUT_ORDERS.
- ▶ IDOC_OUTPUT_<MESSAGE>
The API for outbound functions with message control is described in the upcoming subsection "API for Outbound IDoc Processing Functions." An example is function IDOC_OUTPUT_INVOIC.
- ▶ MASTER_IDOC_CREATE_<MESSAGE>
These are standalone IDocs that are not generated by message control. An example is the BOMMAT message that sends bills of material master data with function MASTER_IDOC_CREATE_BOMMAT.

The Function Builder is divided into the seven tabs seen in Figure 8.6:

1. ATTRIBUTES: Administrative data for the function:
 - ▶ Function group, program names, descriptions, and package
 - ▶ Processing type flag, including remote-enabled (RFC) function
2. IMPORT: Structured strings based on Data Dictionary types used to bring data into the function for processing at runtime.
3. EXPORT: Structured strings based on Data Dictionary types used to return data from the function after processing at runtime.

4. **CHANGING:** Tables or structured strings to carry data that will be changed by the function by runtime processing. Rarely used.
5. **TABLES:** Internal tables based on Data Dictionary tables or structures that will carry table data for processing at runtime.
6. **EXCEPTIONS:** String descriptions of error conditions that can be raised at runtime. Exceptions end processing at the point they are raised and return control to the calling program, which can then use them to trigger error message or other processing.
7. **SOURCE CODE:** Default view where the function's ABAP code is written. The function name at the top of the window is followed by a commented block documenting import and export parameters, tables, and exceptions, as illustrated in Figure 8.6.



```

1 FUNCTION idoc_output_orders.
2  *-----
3  *"Globale Schnittstelle:
4  *"
5  *"   IMPORTING
6  *"       VALUE(OBJECT) LIKE NAST STRUCTURE NAST
7  *"       VALUE(CONTROL_RECORD_IN) LIKE EDIDC STRUCTURE EDIDC
8  *"   EXPORTING
9  *"       VALUE(OBJECT_TYPE) LIKE WFAS1-ASGTP
10 *"       VALUE(CONTROL_RECORD_OUT) LIKE EDIDC STRUCTURE EDIDC
11 *"   TABLES
12 *"       INT_EDIDD STRUCTURE EDIDD
13 *"   EXCEPTIONS
14 *"       ERROR_MESSAGE_RECEIVED
15 *"       DATA_NOT_RELEVANT_FOR_SENDING

```

Figure 8.6 Function Builder Source Code Screen

API for Outbound IDoc Processing Functions

The API for outbound IDoc processing functions is standard for all outbound processing actions that use message control. The outbound function is identified and called by form EDI_PROCESSING in program RSNASTED, but can also be used in custom code. Check the Function Builder for the structures of each parameter.

- ▶ **IMPORTING:** Parameters passed from RSNASTED.
 - ▶ **OBJECT:** The NAST table record containing the document key and output type for the business document that will be read to build the IDoc.

- ▶ **CONTROL_RECORD_IN:** The control record built by `RSNASTED` from the partner profile read before the IDoc function is called.
- ▶ **EXPORTING:** Parameters passed back to `RSNASTED`.
 - ▶ **OBJECT_TYPE:** Business object name for the business document. Links the IDoc to the business document.
 - ▶ **CONTROL_RECORD_OUT:** Fully populated control record built by the IDoc processing function returned to `RSNASTED`.
- ▶ **TABLES:** Internal tables that pass data arrays from the calling function or back to it from the IDoc processing function.
 - ▶ **INT_EDIDD:** Output. Returns the data records for the IDoc from the processing function.
- ▶ **EXCEPTIONS:** Parameters that raise errors in the IDoc processing function. Stop execution of the function at the point the error is trapped and return control to the calling program.
 - ▶ **ERROR_MESSAGE_RECEIVED:** Identifies errors called during IDoc processing.
 - ▶ **DATA_NOT_RELEVANT_FOR_SENDING:** Added to customer functions and method calls in outbound IDoc processing.

API for Inbound IDoc Processing Functions

The API for inbound IDoc processing functions is standard for all inbound processing actions. Inbound functions are called by the standard IDoc interface function `IDOC_INPUT`, but you can also call them from custom code. Check the Function Builder for the structure of each parameter.

- ▶ **IMPORTING:** Parameters passed from the calling function.
 - ▶ **INPUT_METHOD:** Used only for call transaction posting. Default is blank for background mode. Other modes are A for All screens in foreground, and E for Display Error screen only.
 - ▶ **MASS_PROCESSING:** For workflow processing. Default is blank.
- ▶ **EXPORTING:** Parameters passed back to the calling function.
 - ▶ **WORKFLOW_RESULT:** Workflow error handling. Triggers tasks that pass success or error messages to the SAP workplace inbox or other workflow targets.

- ▶ **APPLICATION_VARIABLE:** Advanced workflow programming. Default value is space.
- ▶ **IN_UPDATE_TASK:** Triggers a follow-up task to handle database commit. Default value is space for no update task. "X" delays posting until an explicit commit is called.
- ▶ **CALL_TRANSACTION_DONE:** Set to "X" if the status record isn't updated in the code of the IDoc processing function. In this case, it is updated within the calling function's processing flow.
- ▶ **DOCUMENT_NUMBER:** This is an example of how the standard can vary. `IDOC_INPUT_ORDERS` uses this parameter to return the sales order number after posting, but not all IDoc functions do this.
- ▶ **TABLES:** Internal tables that pass data arrays from the calling function or back to it from the IDoc processing function.
 - ▶ **IDOC_CNTRL:** Input. Passes control record data to the processing function.
 - ▶ **IDOC_DATA:** Input. Passes data records to the IDoc processing function for posting to the document.
 - ▶ **IDOC_STATUS:** Output. Returns status records indicating success or failure in posting for each IDoc passed to the function. Linked to `IDOC_CNTRL` and `IDOC_DATA` through the IDoc number.
 - ▶ **RETURN_VARIABLES:** Output. Returns additional posting results for each IDoc that are used in workflow processing.
 - ▶ **SERIALIZATION_INFO:** Output. Used by the IDoc interface to sort a batch of IDocs in a particular order.

8.1.8 Transaction SMOD—SAP Enhancements

Enhancements collect one or more customer exits that are called from strategic points in the code of the IDoc processing function. The customer exits are grouped together in components within each enhancement.

Click on the **COMPONENTS** button to access the exits. All customer exits that are associated with the component are listed in the **CHANGE PROJECT** screen.

For example, enhancement `SIDOC001` contains one component: customer exit function `EXIT_SAPLEDI1_001`, used to process the control record of the IDoc before

it's created on the IDoc database during inbound and outbound processing (see Figure 8.7). It gives the customer the opportunity to add data to the control record that aren't provided by standard processing.

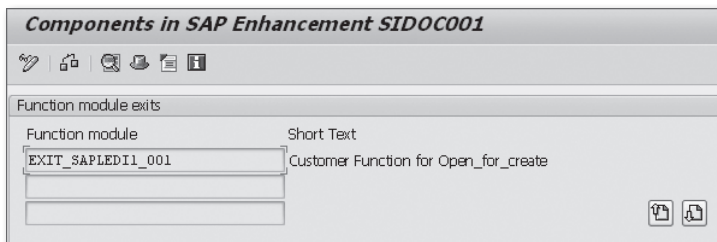


Figure 8.7 Components for Enhancement SIDOC001

This list of functions is our entry point to code the customer exit. Double-click the exit name to open the Function Builder SOURCE CODE window. Note the include statement and program name. We write the code in the include program. But we don't do this through the enhancement object. We first assign the enhancement to a modification project.

Choose ENHANCEMENTS • CUSTOMER EXITS • ENHANCEMENTS in the Repository Information System, use Transaction SMOD, or follow menu path TOOLS • ABAP WORKBENCH • UTILITIES • ENHANCEMENTS • DEFINITION.

Enhancements are stored in tables MODSAP and MODSAPT.

8.1.9 Transaction CMOD—Project Management for SAP Enhancements

We'll use CMOD to create and manage modification to code customer exits for IDoc functions. One or more enhancements can be assigned to each project. For the sake of simplicity, we'll assign only one enhancement per project.

To get to modification projects, use Transaction CMOD, go to ENHANCEMENTS • CUSTOMER EXITS • PROJECTS in the Repository Information System, or follow menu path TOOLS • ABAP WORKBENCH • UTILITIES • ENHANCEMENTS • PROJECT MANAGEMENT.

Modification projects are stored in tables MODACT and MODTEXT.

8.1.10 Transaction WE57—Link Function to Message and Basic Type

This is a key piece of configuration for setting up partner profiles and processing inbound IDocs. It links a processing function, workflow, or task to basic types, extended types, and logical messages.

Different function modules can be assigned to different messages linked to the same basic type. We can add a message code and/or message function and associate different processing functions to the same message and basic type. The message code and function are optional but, if used, become a mandatory part of the partner profile read key.

This gives us tremendous control over how IDocs can be processed under different scenarios and use cases.

The standard link between logical message ORDERS, basic type ORDERS05, and function IDOC_ORDERS_INPUT posts a sales order.

We can add a new link for the same message and basic type to a custom function by creating a message code that will be used, for example, to route IDoc data to a custom table for follow-up reporting. The possibilities are limited only by our imagination and our development budget.

These links drive our partner profile. If we were to link ORDERS and ORDERS05 to custom function module ZSD_INPUT_ORDREPORT and message code SDO, we would create a partner profile with the following parameters:

- ▶ Customer number
- ▶ Partner type KU
- ▶ Partner function SP
- ▶ Message type ORDERS
- ▶ Message code SDO
- ▶ Custom process code linked to the custom function

Although an optional data element, the message code becomes part of the mandatory key for the partner profile if it is used. It must be present in the control segment field MESCOD to trigger this custom processing.

To get to the link editor, use Transaction WE57, select DEVELOPMENT • INBOUND PROCESSING SETTINGS • MESSAGE/APPLICATION OBJECT from the WEDI area menu,

or follow SAP menu path **TOOLS • ALE • ALE DEVELOPMENT • IDOC • INBOUND PROCESSING • FUNCTION MODULE • ASSIGN IDOC TYPE AND MESSAGE TYPE**.

Table EDIFCT stores these links. Table TOJTB adds a link to the corresponding object in the Business Object Repository (BOR).

8.1.11 Transaction BD51—Define IDoc Attributes

Attributes determine how inbound IDoc functions are processed from the following options:

- ▶ **0**
Mass processing for functions that use direct input to the database to post to a document or other data object.
- ▶ **1**
Individual input for functions that use call transaction to post.
- ▶ **2**
Individual input with a call transaction that locks the IDoc.
- ▶ **Dialog allowed**
Allows screen display during call transactions when IDocs are processed in foreground mode.

To define IDoc attributes, choose **DEVELOPMENT • INBOUND PROCESSING SETTINGS • ALE ATTRIBUTES** in the WEDI area menu, use Transaction BD51, or follow menu path **TOOLS • ALE • ALE DEVELOPMENT • IDOC • INBOUND PROCESSING • FUNCTION MODULE • MAINTAIN ATTRIBUTES**.

Attributes for functions are stored in table TBD51.

8.1.12 Transaction WE42—Inbound Process Code

The inbound process code links one IDoc processing function to one or more message types. The process code is required for the inbound partner profile.

Standard process code REMA, for example, is linked to function module `IDOC_INPUT_REMADV` and to logical messages `CREADV`, `DEBADV`, and `REMADV`.

If a message code and/or message function has been added to the link between the logical message, basic type, and processing function in WE57, then the inbound process code must also contain a record with the link between the message type and the message code and/or message function. This is illustrated in Figure 8.8.

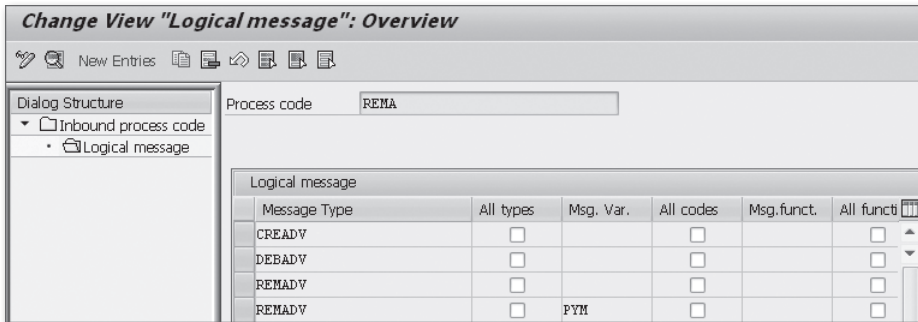


Figure 8.8 Multiple Use Cases Driven by the Process Code

This allows the same function and message to be used in different use cases. You would need to write custom code in a user exit to take advantage of this flexibility.

Use Transaction WE42 to configure the inbound process code or follow menu paths DEVELOPMENT • INBOUND PROCESSING SETTINGS • INBOUND PROCESS CODE in the WEDI area menu or TOOLS • ALE • ALE DEVELOPMENT • IDOC • INBOUND PROCESSING • DEFINE PROCESS CODE IN THE SAP MENU.

The inbound process code is stored in table EDE2T. Its link to the message type key is stored in TMSG2 and to the function module in TEDE2.

8.1.13 Transaction WE41—Outbound Process Code

The outbound process code links one IDoc processing function to one or more message types. Like the inbound, the outbound process code supports multiple use cases through the addition of message codes and/or message functions to the logical message. The configuration screens for the outbound process code are identical to WE41.

The outbound process code is used in the message control screen of the partner profile. It triggers the linked function that reads data from an SAP business document to build and write an IDoc to the database.

Use Transaction WE41 to configure the outbound process code or go to DEVELOPMENT • OUTBOUND PROCESSING SETTINGS/MC • OUTBOUND PROCESS CODE in the WEDI area menu or TOOLS • ALE • ALE DEVELOPMENT • IDOC • OUTBOUND PROCESSING • DEFINE PROCESS CODE in the SAP menu.

The outbound process code is stored in table EDE1T. Its link to the message type key is in TMSG1 and to the message and function module in TEDE1.

8.2 Building a Custom IDoc: Inbound Inventory Report

Now it's time to have a little fun. We're going to build a custom IDoc that maps to an inbound X12 846 inventory report from Acme's vendor Disk Services International (DSI). Before we begin, we'll outline the workflow for creating a custom IDoc from scratch.

8.2.1 Custom IDoc Development Workflow

Figure 8.9 outlines the three steps for building a custom IDoc: develop the IDoc, code the IDoc function, and configure the interface.

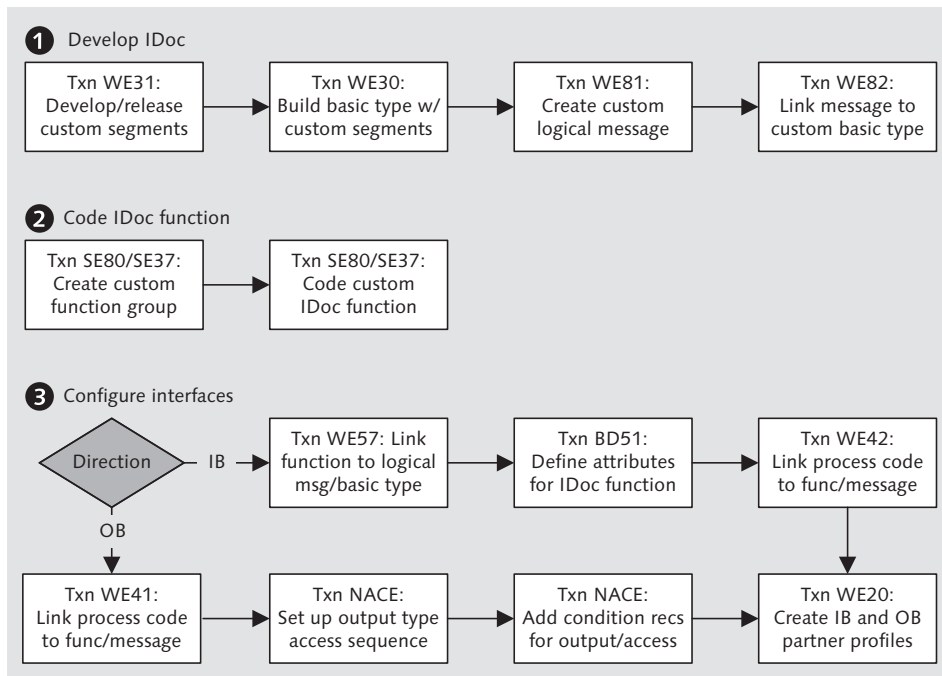


Figure 8.9 Custom IDoc Development Workflow

We'll get into the gritty details as we build our 846 interface. For now, we'll provide a high-level checklist of development tasks:

1. If required, first create domains and data elements in the data dictionary with Transaction SE11.
2. Create segments in the segment editor with Transaction WE31 using standard or custom data elements to create fields.
3. Assemble custom and standard segments into a basic type in the IDoc type editor using Transaction WE30.
4. Create a custom logical message with Transaction WE81.
5. Link the message to the custom basic type with Transaction WE82.
6. Create a new function group for the IDoc processing function with Transaction SE37.
7. Create and code a function module to process the custom IDoc within the new function group with Transaction SE37.
 - ▶ Inbound functions post the IDoc to a document or record.
 - ▶ Outbound functions extract data to build and distribute an IDoc.
8. For an inbound IDoc, do the following configuration:
 - ▶ Link the custom IDoc function to the logical message and basic type with Transaction WE57.
 - ▶ Define attributes for the function with Transaction BD51.
 - ▶ Create a process code linking the logical message to the custom IDoc function with Transaction WE42.
 - ▶ Create an inbound partner profile for the customer, message, and process code with Transaction WE20.
9. For an outbound IDoc, do the following configuration:
 - ▶ Create a process code linking the logical message to the custom IDoc function with Transaction WE41.
 - ▶ Set up message control: output type, access sequence, and condition record for the output and access with Transaction NACE.
 - ▶ Create an outbound partner profile for the customer, message, XML file port, basic type, and message control, including the process code with Transaction WE20.

8.2.2 Building the IDoc Interface

The inbound X12 846 inventory report from DSI carries end-of-day inventory summaries by material for finished movies on DVD and components such as packaging, blank disks, labeling, inserts, stickers, and so on.

Our IDoc will insert 846 data into a custom table (ZEDINVRPT) that will be used for daily inventory balancing reports. Table 8.1 lists its fields; these are labeled as *M* for mandatory and *O* for optional.

Field	Data Element	Description	Req.
MANDT	MANDT	SAP client	M
INVRPTNO	CHAR10	Unique ID for inventory report rec	M
MATNR	CHAR18	Material number	M
WERKS	CHAR4	Plant/warehouse	M
LGORT	CHAR4	Storage location	O
MENGE	QUAN13	Inventory quantity	M
MEINS	UNIT3	Unit of measure	M
CREDAT	DATS8	Create date	O
CRETIM	TIMS6	Create time	O
UPDAT	DATS8	Date record last changed	O
UPTIM	TIMS6	Time record last changed	O

Table 8.1 Structure of Custom Table ZEDINVRPT

We'll name our message ZINVRPT, which follows the SAP convention of using EDIFACT message names for similar objects. We'll name our basic type ZINVRPT01 and it will have the two segments described in Table 8.2.

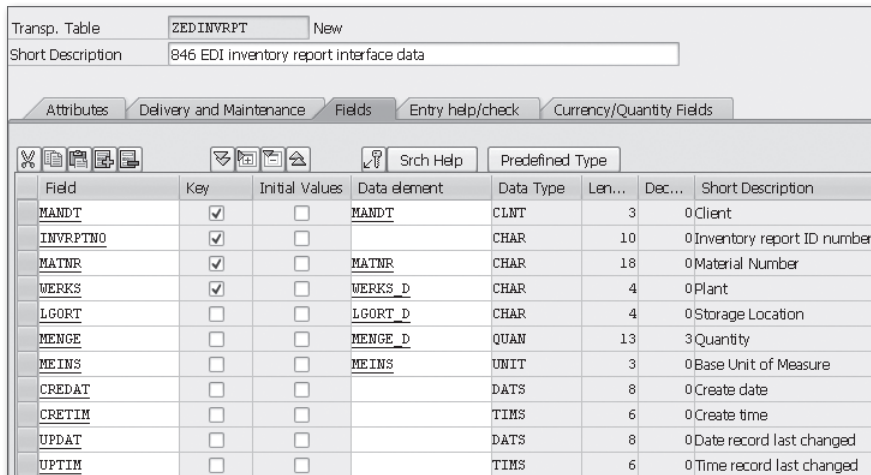
Segment	Description	Usage	Repeat
ZIVRPH	Header-level data	M	1
ZIVRPD	Item-level detail	M	N

Table 8.2 Structure of IDoc Basic Type ZINVRPT01

Creating Custom Table ZEDINVRPT

First we create the custom table ZEDINVRPT. The IDoc will post its data to this table and it is a mandatory part of the development cycle.

1. Run Transaction SE11 and enter table name ZEDINVRPT into the DATABASE TABLE field.
2. Click CREATE to open the DICTIONARY: CHANGE TABLE screen in the DELIVERY AND MAINTENANCE tab. Enter a table description in the SHORT DESCRIPTION field and select DELIVERY CLASS A for application data (master and transaction data).
3. In the FIELDS tab, enter the values shown in Figure 8.10.



Field	Key	Initial Values	Data element	Data Type	Len...	Dec...	Short Description
HANDT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	HANDT	CLNT	3	0	Client
INVRPTNO	<input checked="" type="checkbox"/>	<input type="checkbox"/>		CHAR	10	0	Inventory report ID number
MATNR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	MATNR	CHAR	18	0	Material Number
WERKS	<input checked="" type="checkbox"/>	<input type="checkbox"/>	WERKS_D	CHAR	4	0	Plant
LGORT	<input type="checkbox"/>	<input type="checkbox"/>	LGORT_D	CHAR	4	0	Storage Location
MENGE	<input type="checkbox"/>	<input type="checkbox"/>	MENGE_D	QUAN	13	3	Quantity
MEINS	<input type="checkbox"/>	<input type="checkbox"/>	MEINS	UNIT	3	0	Base Unit of Measure
CREDAT	<input type="checkbox"/>	<input type="checkbox"/>		DATS	8	0	Create date
CRETIM	<input type="checkbox"/>	<input type="checkbox"/>		TIMS	6	0	Create time
UPDAT	<input type="checkbox"/>	<input type="checkbox"/>		DATS	8	0	Date record last changed
UPTIM	<input type="checkbox"/>	<input type="checkbox"/>		TIMS	6	0	Time record last changed

Figure 8.10 Field Structure of Inventory Report Table ZEDINVRPT

4. In the CURRENCY/QUANTITY FIELDS tab a link must be made between the quantity field MENGE and a corresponding unit of measure field in an existing table. Since this is an inventory report, we'll use MSEG-MEINS.
5. Save the table and assign it to a transport. Click TECHNICAL SETTINGS and set DATA CLASS to APPLI (Transaction Data, Transparent Tables) and SIZE CATEGORY to 0 (Data records expected: 0 to 4,300).
6. Save and click REVISED <-> ACTIVATE.
7. Return to the FIELDS tab and click ACTIVATE ($\text{[Ctrl]} + \text{[F3]}$) in the toolbar above the table name field.

Our custom is ready to accept data.

Creating the Segments

We'll begin by creating the segments. Table 8.3 lists the structure of the header segment ZIVRPH.

Pos	Field	Data Element	Description
01	CREDAT	EDI_CCRDAT	Date IDoc created
02	CRETIM	EDI_CCRTIM	Time IDoc created

Table 8.3 Field Structure of Segment ZIVRPH

Table 8.4 lists the field structure of details segment ZIVRPD.

Pos	Field	Data Element	Description
01	MATNR	MATNR	SAP material number
02	WERKS	WERKS_D	Plant/warehouse
03	LGORT	LGORT_D	Storage location
04	MENGE	MENGE_D	Inventory quantity
05	MEINS	MEINS	Unit of measure

Table 8.4 Field Structure of Segment ZIVRPD

We'll first create ZIVRPH in the segment editor (see Figure 7.13). Go to Transaction WE31, enter "ZIVRPH" into the SEGMENT TYPE field, and click CREATE.

1. The CREATE SEGMENT DEFINITION screen opens. Enter a description in the SHORT DESCRIPTION field:
2. Create the following fields:
 - ▶ Enter "CREDAT" in FIELD NAME and "EDI_CCRDAT" in DATA ELEMENT.
 - ▶ Enter "CRETIM" in FIELD NAME and "EDI_CCRTIM" in DATA ELEMENT.
3. Click SAVE. Assign the segment to a package and a change request. It should like Figure 8.11.

P...	Field Name	Data element	IS...	Export leng
1	CREDAT	EDI_CCRDAT	<input type="checkbox"/>	8
2	CRETIM	EDI_CCRTIM	<input type="checkbox"/>	6

Figure 8.11 Fields Added to the Custom Segment

- Press **[F3]** to back out to the segment editor's opening screen. Release the segment by selecting menu option **EDIT • SET RELEASE**. Once released, the initial screen will look like Figure 8.12.

Version	Segm. definition	Release	Release	A.	No. of fields	Lgth	Date of l...	Time ...
000	ZIVRPH000	<input checked="" type="checkbox"/>	702	<input type="checkbox"/>	2	14	04/13/2013	21:33:0.

Figure 8.12 Segment ZIVRPH After It's Been Released

- Follow the same steps to create and release details segment ZIVRPD using the field and data element names in Table 8.4.

Building an IDoc Basic Type

We'll create our custom IDoc basic type with Transaction WE30.

- Enter "ZIVRPT01" into the **OBJ. NAME** field, select **BASIC TYPE**, and click **CREATE**. The **CREATE BASIC TYPE** dialog opens after the system informs you that the name is longer than eight characters. The following are radio buttons under **New basic IDoc type**:
 - ▶ **CREATE NEW** builds a new custom IDoc type.
 - ▶ **CREATE AS COPY** copies an existing IDoc type that we can change.

- ▶ CREATE SUCCESSOR creates a new release version of an existing custom IDoc type.
2. Select CREATE NEW, and click OK to open the IDoc type editor. We'll assemble our basic type from segments in this screen. To add segments, put the cursor on the IDoc type root name and click CREATE segment, press **Shift + F6**, or follow the menu path EDIT • CREATE SEGMENT.
 3. The MAINTAIN ATTRIBUTES dialog opens, as in Figure 8.13. To add header segment ZIVPRH, do the following:
 - ▶ Enter "ZIVRPH" in the SEGM.TYPE field.
 - ▶ Select the MANDATORY SEG. checkbox.
 - ▶ Enter "1" in MINIMUM NUMBER and "1" in MAXIMUM NUMBER.

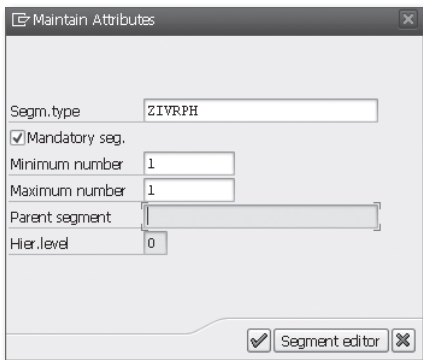


Figure 8.13 Adding a New Segment to the IDoc Type

4. Click OK. ZIVRPH is inserted as a child of the basic type root. Select ZIVRPH and click CREATE SEGMENT to add the details segment.
5. The SEGMENT HIERARCHY dialog opens. Select ADD SEGMENT TYPE AS CHILD.
6. Click OK to open the MAINTAIN ATTRIBUTES dialog. Do the following for the next segment:
 - ▶ Enter "ZIVRPD" in the SEGM.TYPE field.
 - ▶ Select the MANDATORY SEG. checkbox.
 - ▶ Enter "1" in MINIMUM NUMBER and "999999" in MAXIMUM NUMBER.
7. ZIVRPD is added as a child segment to ZIVRPH, as shown in the basic type editor in Figure 8.14.

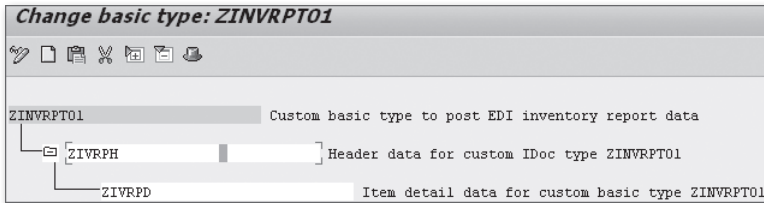


Figure 8.14 Our New IDoc Basic Type ZINVRPT01

8. Double-click ZIVRPD to view the MAINTAIN ATTRIBUTES dialog. The parent segment is ZIVRPH, and the hierarchy level is 2.
9. Save the basic type, and assign it to a package and a change request. Back out of the edit window. Release the IDoc basic type by selecting menu option EDIT • SET RELEASE.

Create a Custom Message Type

Next we create the logical message. Use Transaction WE81, and click DISPLAY CHANGE (menu option TABLE VIEW • DISPLAY -> CHANGE). Click NEW ENTRIES (menu option EDIT • NEW ENTRIES) to open the OVERVIEW OF ADDED ENTRIES screen.

Enter "ZINVRPT" in MESSAGE TYPE and a description of the message in the SHORT TEXT field, as shown in Figure 8.15. Save the message type and assign it to a customizing request.

EDI: Logical Message Types	
Message Type	Short text
ZINVRPT	EDI inventory report

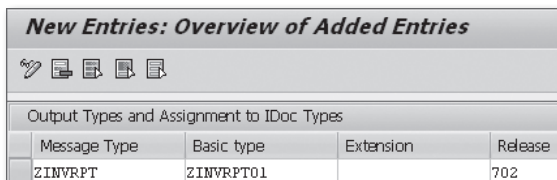
Figure 8.15 Create a Custom Message Type for the IDoc

Link Message to Basic Type

Now we link the logical message to the basic type with Transaction WE82. Click Display Change (menu option TABLE VIEW • DISPLAY -> CHANGE) and then click New Entries (menu option EDIT • NEW ENTRIES) to open the OVERVIEW OF ADDED ENTRIES screen.

Enter the following values into the table control, as illustrated in Figure 8.16:

- ▶ Enter "ZINVRPT" in the MESSAGE TYPE field.
- ▶ Enter "ZINVRPT01" in the BASIC TYPE field.
- ▶ Enter version "702" (your current SAP system release) in the RELEASE field.



New Entries: Overview of Added Entries			
Output Types and Assignment to IDoc Types			
Message Type	Basic type	Extension	Release
ZINVRPT	ZINVRPT01		702

Figure 8.16 Linking the Message Type to the IDoc Basic Type

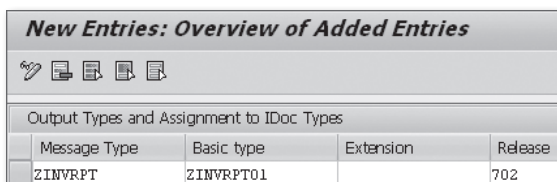
Save the entry and assign it to a customizing request.

Create the Function Group

We'll create the function group in the object navigator.

1. Run Transaction SE80 and click EDIT OBJECT at the top of the OBJECT NAVIGATOR window.
2. Select the FUNCTION GROUP tab, enter "ZEDINVRP" in the GROUP NAME field, and click CREATE.
3. The CREATE FUNCTION GROUP dialog opens. Enter a short description in the SHORT TEXT field, and click SAVE.
4. Assign the function group to a package and a change request.

Function group ZEDINVRP is now ready for coding. It should look like Figure 8.17 in the object navigator.



New Entries: Overview of Added Entries			
Output Types and Assignment to IDoc Types			
Message Type	Basic type	Extension	Release
ZINVRPT	ZINVRPT01		702

Figure 8.17 Function Group ZEDINVRP is Primed for Coding

Coding the Inbound IDoc Processing Function

We're still in the object navigator:

1. Click EDIT OBJECT, select FUNCTION MODULE in the FUNCTION GROUP tab, and enter the function name "ZIDOC_INPUT_ZINVRPT".
2. Click CREATE. The CREATE FUNCTION MODULE dialog opens.
3. The IMPORT PARAMETERS screen of the Function Builder opens. We'll use standard import parameters that will be passed to the function by the IDoc interface at runtime.
4. Enter the following values illustrated in Figure 8.18.

Hint

Copy and paste standard values for all parameter tabs from any standard inbound processing IDoc function.

Parameter Name	Typ...	Associated Type	Default ...	O...	Pass Va...	Short text
INPUT_METHOD	TYPE	BDWFAP_PAR-INPUTMETHD		<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inbound method for the IDoc inbound function module
MASS_PROCESSING	TYPE	BDWFAP_PAR-MASS_PROC		<input type="checkbox"/>	<input checked="" type="checkbox"/>	Flag: Mass processing

Figure 8.18 Standard Import Parameters for Inbound IDoc Functions

5. Export parameters return workflow and other information to the IDoc interface at runtime. You can also return document numbers and other application data for reporting.

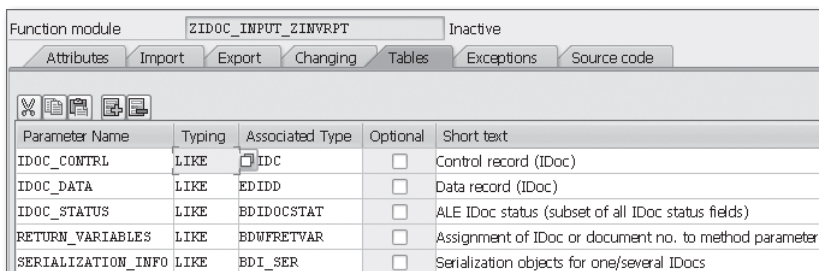
Click the EXPORT tab and enter the values illustrated in Figure 8.19.

Parameter Name	Typing	Associated Type	Pass V...	Short text
WORKFLOW_RESULT	TYPE	BDWFAP_PAR-RESULT	<input checked="" type="checkbox"/>	Final value of method
APPLICATION_VARIABLE	TYPE	BDWFAP_PAR-APPL_VAR	<input checked="" type="checkbox"/>	Variable to be used by application as required
IM_UPDATE_TASK	TYPE	BDWFAP_PAR-UPDATETASK	<input checked="" type="checkbox"/>	Flag: Application has triggered update task
CALL_TRANSACTION_DONE	TYPE	BDWFAP_PAR-CALLTRANS	<input checked="" type="checkbox"/>	Flag: Application has actually performed call transaction

Figure 8.19 Standard Export Parameters for Inbound IDoc Functions

- Click the TABLES tab and enter the values for the internal tables listed in Figure 8.20.

The tables will pass data in and out of the function. We're most interested in the IDoc control, data, and status records. We won't add any exceptions for this example.



Parameter Name	Typing	Associated Type	Optional	Short text
IDOC_CONTRL	LIKE	<input checked="" type="checkbox"/> IDC	<input type="checkbox"/>	Control record (IDoc)
IDOC_DATA	LIKE	<input checked="" type="checkbox"/> EDIDD	<input type="checkbox"/>	Data record (IDoc)
IDOC_STATUS	LIKE	<input checked="" type="checkbox"/> BDIDOCSTAT	<input type="checkbox"/>	ALE IDoc status (subset of all IDoc status fields)
RETURN_VARIABLES	LIKE	<input checked="" type="checkbox"/> BDWFRETVAR	<input type="checkbox"/>	Assignment of IDoc or document no. to method parameter
SERIALIZATION_INFO	LIKE	<input checked="" type="checkbox"/> BDI_SER	<input type="checkbox"/>	Serialization objects for one/several IDocs

Figure 8.20 Standard Table Parameters for Inbound IDoc Functions

- Click the SOURCE CODE tab and enter the function's code as it appears in Listing 8.1.

```

*-----
FUNCTION ZIDOC_INPUT_ZINVRPT.
*-----
***Local Interface:
*  IMPORTING
*    VALUE(INPUT_METHOD) TYPE  BDWFAP_PAR-INPUTMETHD
*    VALUE(MASS_PROCESSING) TYPE  BDWFAP_PAR-MASS_PROC
*  EXPORTING
*    VALUE(WORKFLOW_RESULT) TYPE  BDWFAP_PAR-RESULT
*    VALUE(APPLICATION_VARIABLE) TYPE  BDWFAP_PAR-APPL_VAR
*    VALUE(IN_UPDATE_TASK) TYPE  BDWFAP_PAR-UPDATETASK
*    VALUE(CALL_TRANSACTION_DONE) TYPE  BDWFAP_PAR-CALLTRANS
*  TABLES
*    IDOC_CONTRL STRUCTURE  EDIDC
*    IDOC_DATA STRUCTURE  EDIDD
*    IDOC_STATUS STRUCTURE  BDIDOCSTAT
*    RETURN_VARIABLES STRUCTURE  BDWFRETVAR
*    SERIALIZATION_INFO STRUCTURE  BDI_SER
*-----
*Data declarations
data: izedinvrpt type standard table of
      zedinvrpt with header line.

```

```

data: gs_zivrph type zivrph,
      gs_zivrpd type zivrpd.
data: gs_last_no like zedinvrpt-invrptno.
*Get last used ID number
clear gs_last_no.
select max( invrptno ) into gs_last_no
      from zedinvrpt.
gs_last_no = gs_last_no + 1.
*process IDoc records and format insert for izedinvrpt
loop at idoc_contrl.
  refresh izedinvrpt. clear izedinvrpt.
  loop at idoc_data where docnum = idoc_contrl-docnum.
    case idoc_data-segnam.
*process header record
      when 'zivrph'.
        gs_zivrph = idoc_data-sdata.
*process detail record
      when 'zivrpd'.
        gs_zivrpd = idoc_data-sdata.
        izedinvrpt-mandt = sy-mandt.
        izedinvrpt-invrptno = gs_last_no.
        izedinvrpt-matnr = gs_zivrpd-matnr.
        izedinvrpt-werks = gs_zivrpd-werks.
        izedinvrpt-lgort = gs_zivrpd-lgort.
        izedinvrpt-menge = gs_zivrpd-menge.
        izedinvrpt-meins = gs_zivrpd-meins.
        izedinvrpt-credat = gs_zivrph-credat.
        izedinvrpt-cretim = gs_zivrph-cretim.
        izedinvrpt-upddat = sy-datum.
        izedinvrpt-updtim = sy-uzeit.
        append izedinvrpt.
        gs_last_no = gs_last_no + 1.
    endcase.
  endloop.
*insert IDoc records to table zedinvrpt
  modify zedinvrpt from table izedinvrpt.
  if sy-subrc = 0.
*success message to status record
  clear idoc_status.
  idoc_status-docnum = idoc_contrl-docnum.
  idoc_status-msgty = 's'.
  idoc_status-msgid = 'ZEDI01'.
  idoc_status-msgno = '001'.

```

```

        idoc_status-status = '53'.
        append idoc_status.
    else.
*verify record count and total before updating.
        clear idoc_status.
        idoc_status-docnum = idoc_cntrl-docnum.
        idoc_status-msgty = 'e'.
        idoc_status-msgid = 'ZEDI01'.
        idoc_status-msgno = '002'.
        idoc_status-status = '51'.
        append idoc_status.
    endif.
endloop.

ENDFUNCTION.

```

Listing 8.1 Source Code for Inventory Report IDoc Function

This code is stripped down to its essentials. The control and data records are imported into the function at runtime through the `IDOC_CNTRL` and `IDOC_DATA` internal tables. The program logic follows:

1. Gets the last used invoice report ID from `ZEDINVRPT-INVVRPTNO` and increments it for use by the incoming records.
2. Loops through `IDOC_CNTRL`.
3. Loops through `IDOC_DATA` at the current IDoc number.
4. Evaluates the `SEGNAM` field for the current segment name.
5. Moves the `SDATA` field in the data record to a string structured by the segment type.
6. Moves data from the structured string to an internal table structured by our target database table `ZEDINVRPT`.
7. When the loops on `IDOC_DATA` and `IDOC_CNTRL` are done and internal table `IZED-INVVRPT` is fully populated with all IDoc data, then database table `ZEDINVRPT` is updated from the internal table.
8. If the update succeeds, the status record is updated with status 53, and a success message pulled from custom message class `ZEDI01`.
9. If the update fails, the status record is updated with status 51, and an error message pulled from custom message class `ZEDI01`.

10. Status and message values are passed to internal table `IDOC_STATUS` and returned to the calling IDoc interface function `IDOC_INPUT`, which passes them to routines that update the status records in the database.

All inbound IDoc processing functions work in essentially the same way, with varying levels of complexity for data processing and checks depending on transactional requirements. If you understand this processing approach, you'll understand a lot about the IDoc interface.

Now that the code is written, we'll step through the configuration we need to plug all of our custom objects into the standard IDoc interface.

Link the Function to Message and Basic Type

First up is to link the custom function to our logical message and basic type.

1. Go to Transaction WE57, and click `DISPLAY -> CHANGE` (menu path `TABLE VIEW • DISPLAY -> CHANGE`)
2. Click `NEW ENTRIES` (or press `F5`) to open the `DETAILS OF ADDED ENTRIES` screen and enter the following values into it:
 - ▶ Enter `"ZIDOC_INPUT_ZINVRPT"` into the `FUNCTION Module` field.
 - ▶ Select `FUNCTION MODULE` from the `FUNCTION TYPE` dropdown.
 - ▶ Enter `"ZINVRPT01"` in the `BASIC TYPE` field.
 - ▶ Enter `"ZINVRPT"` in the `MESSAGE TYPE` field.
 - ▶ Select `INBOUND` from `DIRECTION` dropdown.

Click `SAVE` and assign the changes to a customizing request. The screen should look like Figure 8.21 after you're done.

Set Attributes for the Function

To set attributes for the function, follow these steps:

1. Go to Transaction BD51 and click `NEW ENTRIES`.
2. Enter `"ZIDOC_INPUT_ZINVRPT"` in the `FUNCTION MODULE (INBOUND)` field and `"0"` in `INPUT T. column` for direct input. The function directly inserts IDoc data into the custom table.

New Entries: Details of Added Entries

Function module: ZIDOC_INPUT_ZINVRPT

Function type: Function module

Basic type: ZINVRPT01

Extension:

Message Type: ZINVRPT

Message Variant:

Mess. function:

Object Type:

IDoc: Assignment of FM to Log, Message and IDoc Type

Direction: Inbound

Description: EDI inventory report

Name:

Figure 8.21 Linking the Function to the IDoc Basic and Message Types

3. Save and assign the attributes to a transport. The screen should look like Figure 8.22.

New Entries: Overview of Added Entries

Characteristics of Inbound Function Modules

Function module (inbound)	Input t.	Dialog allowed
ZIDOC_INPUT_ZINVRPT	0	<input type="checkbox"/>

Figure 8.22 Function Attributes for Direct Input

Create a Custom Process Code

The process code ties all the processing pieces together.

1. Go to Transaction WE42, and switch to change mode.
2. Click NEW ENTRIES to open the DETAILS OF ADDED ENTRIES screen, and do the following:
 - ▶ Enter "ZINRP" in the PROCESS CODE field.
 - ▶ Enter a text description in the DESCRIPTION field. Begin the description with the message name.
 - ▶ Set PROCESSING TYPE as the function module.
3. Save the entry and assign it to a transport.

4. A more detailed view of the ADDED ENTRIES screen opens. Select the function "ZIDOC_INPUT_ZINVRPT" from the FUNCTION MODULE dropdown list.
5. Save and add to a transport. After the function has been assigned, back out via **F3** to the added entries screen. The function name will be linked to the process code. The screen should look like Figure 8.23.

Figure 8.23 The Process Code and Function Module Linked

6. Double-click the LOGICAL MESSAGE folder in the DIALOG STRUCTURE navigation display to open the Logical message overview screen. Click NEW ENTRIES and enter "ZINVRPT" into the MESSAGE TYPE field, as shown in Figure 8.24.

Figure 8.24 Link the Logical Message to the Process Code

Don't forget to save. This completes the link between the process code, the logical message, and the custom function module. All these objects are now ready to be used in a partner profile.

Define the Partner Profile

All that's left is the inbound partner profile. The inventory report is coming from the supplier Disc Services International, entered in Acme's SAP vendor master as DISK01.

So we'll create a partner profile with partner type LI for vendor.

1. Run Transaction WE20, select the PARTNER TYPE LI folder, and create a vendor partner profile header for DSI with partner number DISK01.
2. Save the general view and create inbound parameters to add message type ZINVRPT. Enter the following values into the INBOUND PARAMETERS screen:
 - ▶ Enter "VN" in the PARTNER ROLE field.
 - ▶ Enter "ZINVRPT" in the MESSAGE TYPE field.
 - ▶ Enter "ZINRP" in the PROCESS CODE field.
 - ▶ Set TRIGGER IMMEDIATELY as the processing mode. This is a simple table insert that comes in once a day and doesn't involve a lot of processing, so we won't schedule a background job.
3. Save the partner profile. The finished product should look like the inbound partner profile in Figure 8.25.

Partner profiles: Inbound parameters

Partner No. DISK01 Disc Services International

Partn. Type LI

Partner Role VN Vendor

Message type ZINVRPT EDI inventory report

Message code

Message function Test

Inbound options | Post processing: permitted agent | Telephony

Process code ZINRP 846 EDI inventory report proce...

Cancel Processing After Syntax Error

Processing by Function Module

Trigger by background program

Trigger Immediately

Figure 8.25 Inbound Parameters for DSI Message Type ZINVRP

This interface is now ready to roll—after it goes through the full testing cycle, of course.

8.3 Extending an IDoc: Outbound PO with BOMs

Next we'll try our hand at adding a custom segment to a standard IDoc. Our scenario is pretty simple. DSI needs to receive the bill of materials when Acme sends them a purchase order. They need to know what components Acme expects them to use in manufacturing.

DSI may have the components in inventory or they may need to order from a third-party supplier. The purchase order BOM is key to their manufacturing process.

A custom segment, Z1EDP01, will be created to hold the components just below E1EDP01, the parent for all line item segments, in an extension of IDoc basic type ORDERS05 that we will name ZORDRS01.

The code will be written in a CMOD modification project using enhancement SDEDI001 in component `EXIT_SAPLVEDC_002`.

8.3.1 IDoc Outbound Development Workflow

Extended IDocs are built on existing standard basic types. They are created to send or receive data that are not accommodated in a standard IDoc. Figure 8.26 outlines the workflow for extending an IDoc, which is broken into three main steps: extend the IDoc, code the IDoc enhancement, and configure the interfaces.

To create an extended IDoc type, you insert one or more segments into a standard basic type. The insertion point should make sense in terms of the data the extended segment contains and its semantic context within the IDoc.

We also need custom code to populate and process the extended segments. The code is usually written in a user exit, but sometimes you need a custom IDoc function. For example, message type MBGMCR doesn't have any customer exits, so it would need a custom version of its processing function if it were extended or if you needed to do any non-standard processing.

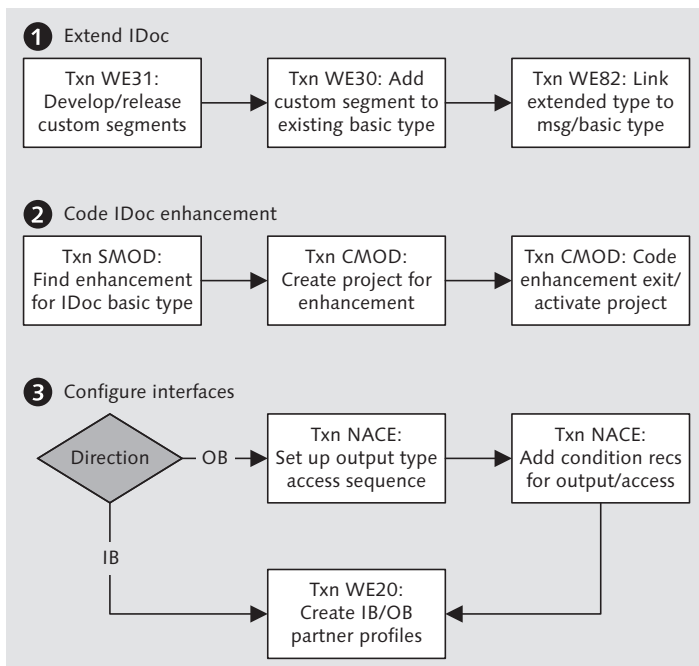


Figure 8.26 Basic Workflow for Extending IDocs

The process for creating and configuring a custom function for an extended IDoc type is the same as for a custom IDoc with the exception that we don't need to create a new message type.

To extend an IDoc by coding a customer exit, follow these steps:

1. Create custom segments in the segment editor with Transaction WE31.
2. Copy an existing standard basic type into an extended type in the IDoc type editor with Transaction WE30.
3. Insert custom segments into the extended type with Transaction WE30.
4. Link the extended type to the logical message and basic type with Transaction WE82.
5. Identify an enhancement with user exits for the extended type with Transaction SMOD.
6. Create a modification project to write and manage the custom code for the exit with Transaction CMOD.

7. Code the customer exit enhancement component in the CMOD project.
8. For an inbound extended IDoc, create inbound partner profile for the customer, message, and process code with Transaction WE20.
9. For an outbound extended IDoc, there are two additional steps:
 - ▶ Set up message control (output type, access sequence, and condition record for the output and access with Transaction NACE).
 - ▶ Create outbound partner profile for the customer, message, XML file port, basic and extended types, and message control with Transaction WE20.

Identifying Customer Exits

Let's root around the Data Dictionary and consider a simple backend way to identify user exits that we can use in modification projects. First we need to look at the standard frontend approach.

Modification projects organize our user exit work. We can add more than one enhancement to a project allowing it to encompass an entire business process. But at Acme, we'll only create one project per enhancement.

The following process is the typical frontend approach to finding exits:

1. Identify the function for the IDoc that will be extended.
2. Go to the Function Builder in Transaction SE37, enter the function name in the FUNCTION MODULE field, and click DISPLAY.
3. Click ATTRIBUTES to get the package name: VED for all SD EDI development objects and ME for purchasing documents, including EDI and ALE exits.
4. Go to Transaction SMOD, select the input help dropdown (or press) and click INFORMATION SYSTEM to open the REPOSITORY INFO SYSTEM: FIND EXITS dialog.
5. Enter the package name in the PACKAGE field of the search help dialog and click EXECUTE.
6. The REPOSITORY INFO SYSTEM: FIND EXITS dialog opens, listing all enhancements in the package.

For SD package VED, nine hits are returned; for purchasing package ME, 36 hits are returned in ERP 6.0, as shown in Figure 8.27.

Exit name	Short text
AMPL0001	User subscreen for additional data on AMPL
LMDR001	Enhancements to print program
LMEIA002	Adopt batch no. from shipping notification when posting
LMEIA010	Inbound shipping notification: Transfer item data from
LMEQR001	User exit for source determination
LMEXF001	Conditions in Purchasing Documents Without Invoice Rece
LWSUS001	Customer-Specific Source Determination in Retail
MM06B0001	Role determination for purchase requisition release
MM06B0002	Changes to comm. structure for purchase requisition rel
MM06B0003	Number range and document number
MM06B0004	Number range and document number
MM06B0005	Changes to comm. structure for overall release of requi
MM06E0004	Changes to communication structure for release purch. d
MM06E0005	Role determination for release of purchasing documents
ME590001	Grouping of requisitions for PO split in ME59
MEETA001	Define schedule line type (backlog, immed. req., previe
MEFLD004	Determine earliest delivery date f. check w. GR (only P
MELAB001	Gen. forecast delivery schedules: Transfer schedule imp
MEQUERY1	Enhancement to Document Overview ME2LN/ME5LN
MEVME001	ME default quantity calc. and over/ underdelivery toler
MM06E0001	User exits for EDI inbound and outbound purchasing docu
MM06E0003	Number range and document number
MM06E0004	Control import data screens in purchase order
MM06E0005	Customer fields in purchasing document
MM06E0007	Change document for requisitions upon conversion into P

Figure 8.27 SMOD Search Help Results for Package ME

Descriptions of enhancements don't always reveal their purpose or identify their message types. For example, if you search package ME for user exits in PO message ORDERS, you'd have to carefully scan the list of 36 enhancements to learn that MM06E001 is the one you need.

We can use the Data Dictionary backend to refine our search. It takes a couple of steps however. First replicate the SMOD search help results by following these steps:

1. Go to the Data Browser with Transaction SE16.
2. Enter "TADIR" into the TABLE NAME field and press . TADIR contains all development repository objects in the system.
3. Enter the following values into the selection screen for TADIR.
 - ▶ Enter "R3TR" in the PGMID field.
 - ▶ Enter "SMOD" in the OBJECT field.
 - ▶ Enter "ME" in the DEVCLASS field.
4. Click EXECUTE or press .

A list of every enhancement stored in package VED is returned in the field OBJNAME. So what do we need to know to make our job of identifying exits easier?

Focusing on Enhancements

Each enhancement contains all the user exits available to any IDoc function in components that hold a function module with the naming convention EXIT_<PROGRAM>_00X where:

- ▶ PROGRAM is the name of the function pool program that contains the IDoc processing function module.
- ▶ 00X is the number of the function within the exit function group.

For example, enhancement SIDOC001, which is used to change the control segment just before an IDoc is written to the database, has only one component: function EXIT_SAPLEDI1_001.

Enhancement MM06E001, with purchasing document exits, has 20 components; they are all called during various processing stages of more than one message type, including outbound purchase orders, inbound purchase order change and acknowledgments, and inbound ship notifications.

For processing data in the outbound PO, the key components include the following:

1. EXIT_SAPLEINM_001: Control record changes
2. EXIT_SAPLEINM_002: Data record changes while IDoc is being built
3. EXIT_SAPLEINM_011: Final data changes after the IDoc has been built

Note that EXIT_SAPLEINM_011 is also called for outbound message types ORDCHG (PO change), REQOTE (request for quotation), BLAORD (purchasing contracts), and BLAOCH (purchasing contract change).

The exit names are in function IDOC_OUTPUT_ORDERS. You can find them by going into the code of the function in Transaction SE37 and searching the string CUSTOMER-FUNCTION using the binocular FIND icon at the top of the FUNCTION BUILDER screen. Select IN MAIN PROGRAM.

Every call to a customer function in every message type processed by the main program is listed. The call syntax is CALL CUSTOMER-FUNCTION '00X' where 00X is

the number of the exit being called. We're interested in `CUSTOMER-FUNCTION '002'` for `EXIT_SAPLEINM_011`.

Click the instance of the customer function call in the `GLOBAL SEARCH IN PROGRAMS` window, and the system will go to the call point in the code. The customer function is called after each segment has been appended to `INT_EDIDD`, the internal table used to build the IDoc.

The `'002'` in single quotes following `CALL CUSTOMER-FUNCTION` is the function name. Double-click `'002'` within the single quotes and the system navigates to the exit function. The source code for the function is listed in Listing 8.2.

```

*-----
FUNCTION EXIT_SAPLEINM_002.
*-----
***Global Interface:
*  IMPORTING
*    VALUE(XEKKO) LIKE EKKO STRUCTURE  EKKO
*    VALUE(XLFA1) LIKE LFA1 STRUCTURE  LFA1
*    VALUE(XLFB1) LIKE LFB1 STRUCTURE  LFB1
*    VALUE(DOBJECT) LIKE NAST STRUCTURE  NAST OPTIONAL
*  TABLES
*    INT_EDIDD STRUCTURE EDIDD
*    XEKPO STRUCTURE UEKPO OPTIONAL
*    XEKET STRUCTURE UEKET OPTIONAL
*    DEKEK_X STRUCTURE EKEK_X OPTIONAL
*    DEKEH STRUCTURE IEKEH OPTIONAL
*    DSADR STRUCTURE SADR OPTIONAL
*    DVBAK STRUCTURE MMVBAK OPTIONAL
*    DVBAK STRUCTURE MMVBAK OPTIONAL
*    DVBAK STRUCTURE MMVBAK OPTIONAL
*    DVBAK STRUCTURE MMVBAK OPTIONAL
*    DVBAK STRUCTURE MMVBAK OPTIONAL
*  CHANGING
*    VALUE(ISC_ENHANCEMENT) TYPE ISC_EXIT_SAPLEINM_002 OPTIONAL
*  EXCEPTIONS
*    ERROR_MESSAGE_RECEIVED
*    DATA_NOT_RELEVANT_FOR_SENDING
*-----

  INCLUDE ZXM06U02.

ENDFUNCTION.

```

Listing 8.2 Call to Enhancement Component for Outbound PO

The `IMPORTING` parameters of the exit are identical to the `EXPORTING` parameters of the customer function, which is a shell that calls the exit.

Note include program `ZXM06U02`, which is where the exit code goes. The program doesn't exist until the system creates it when we double-click its name. We can write our code here through the IDoc function, but it is a best practice to code and manage exits in a modification project.

Shortcut to Identifying Enhancements

Let's go back to the `ATTRIBUTES` screen of `IDOC_OUTPUT_ORDERS` in Transaction `SE37`. There are values here that can help us identify our enhancement in the Data Dictionary. Copy the program name `SAPLEINM`. We can use it to identify which enhancement we need through a table read.

1. Go back to the Data Browser with Transaction `SE16`.
2. Enter "MODSAP" in the `TABLE NAME` field, and press . `MODSAP` stores SAP enhancements and their components.
3. Enter the program name bracketed by asterisks (`*SAPLEINM*`) in the `MEMBER` field. Or, better yet, get the exit name from the IDoc processing function (`EXIT_SAPLEINM_002`). Click `EXECUTE`.

The enhancement `MM06E001` is in the `NAME` field.

Between the IDoc processing function and table `MODSAP`, you can quickly identify the enhancement you need to add to your modification project.

Exit Function Groups

The names of exit function groups begin with an X, which can narrow a search for exit functions in Transaction `SE37`. An open-ended search for function groups that begin with X in ECC 6.0 returns 5,899 exits for all applications.

Function group `XM06` contains all 61 user exits for package `ME`, which contains all development objects for purchasing.

One of the joys of working with SAP is that a little educated poking around goes a long way. And it's fun, too!

It's time to build our extended IDoc. We'll begin with the custom segment.

8.3.2 Create Segment Z1EDP01

The structure of custom segment Z1EDP01 is described in Table 8.5.

Pos	Field	Data Element	Description
01	MATNR	MATNR	SAP component material number
02	MAKTX	MAKTX	Component description

Table 8.5 Structure of BOM Segment Z1EDP01

Build the custom segment in the segment editor with Transaction WE31. Enter "Z1EDP01" in the NAME field and click CREATE to open the CREATE SEGMENT DEFINITION screen.

Enter a description in the SHORT DESCRIPTION field. Add the field and data elements from Table 8.5 into the FIELD NAME and DATA ELEMENT fields in the table control.

Click SAVE and assign the segment to a change request. Press **[F3]** to back out to the segment editor's initial screen. Release the segment by selecting menu option Edit • SET RELEASE.

8.3.3 Build Extension ZORDRS01

Next we extend the standard basic type:

1. Go to the IDoc type editor with Transaction WE30. Enter the name of the IDoc Extension ("ZORDRS01") in the OBJ. NAME field, select EXTENSION, and click CREATE. The CREATE EXTENSION DIALOG opens, as shown in Figure 8.28.
2. Select CREATE NEW and enter "ORDERS05" in the LINKED BASIC TYPE field. Click OK to open the IDoc type editor.

The structure of basic type ORDERS05 is displayed in the editor with the root name ZORDRS01.

3. Expand the E1EDP01 item group by clicking on the folder icon next to the segment name.
4. To add the custom segment, put the cursor on the E1EDP01 segment name and click CREATE SEGMENT, or press **[Shift]+[F6]**, or choose menu option EDIT • CREATE SEGMENT. A pop-up informs you that the custom segment will be added as a child to E1EDP01. Click OK.

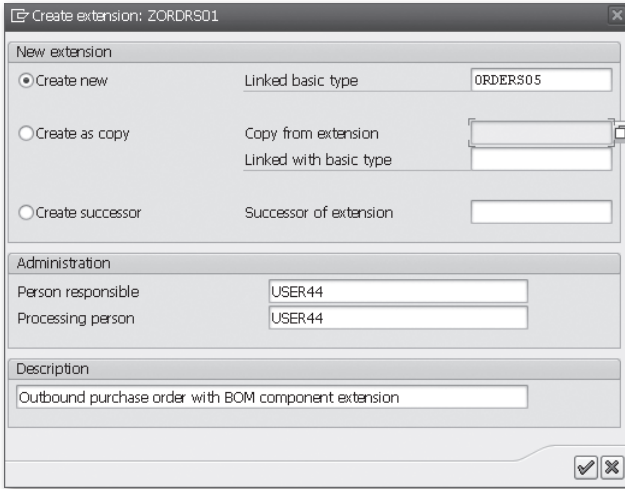


Figure 8.28 Select a Linked IDoc Basic Type to Create an Extension

5. The MAINTAIN ATTRIBUTES dialog opens. Enter the following values to add the Z1EDP01 segment:

- ▶ Enter "Z1EDP01" in the SEGM.TYPE field.
- ▶ Enter "1" in the MINIMUM NUMBER field.
- ▶ Enter "999" in the MAXIMUM NUMBER field.

6. Click OK. Z1EDP01 appears as a child of E1EDP01 (Figure 8.29). Extended type ZORDRS01 now has its custom segment.

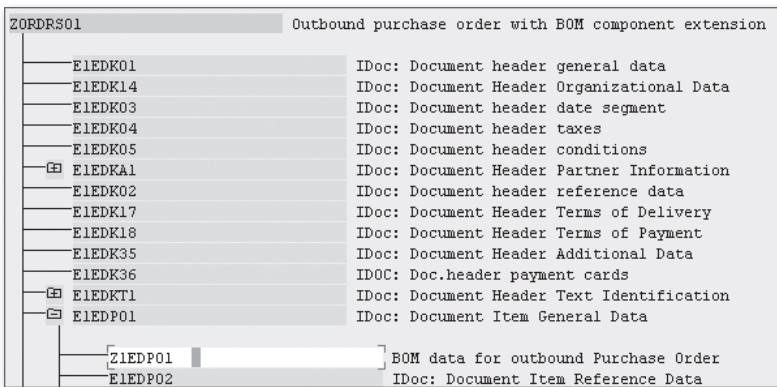


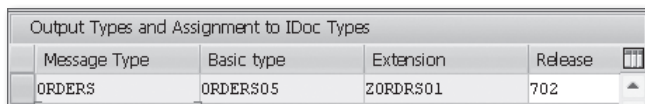
Figure 8.29 Z1EDP01 Is a Child of E1EDP01

7. Save the extension, assign it to a change request, and back out of the edit window.
8. Release the extended type in the initial screen by selecting menu option EDIT • SET RELEASE.

8.3.4 Link Message to Basic and Extended Types

To link the message and basic type to the extended type, follow these steps:

1. Go to Transaction WE82, and click CHANGE • DISPLAY, press **Ctrl** + **F4**, or follow menu path TABLE VIEW • DISPLAY -> CHANGE.
2. Click NEW ENTRIES (or press **F5**) to open the OVERVIEW OF ADDED ENTRIES screen. Enter the following values in the table control (see Figure 8.30):
 - ▶ Enter "ORDERS" into the MESSAGE TYPE field.
 - ▶ Enter "ORDERS05" into the BASIC TYPE field.
 - ▶ Enter "ZORDRS01" into the EXTENSION field.
 - ▶ Enter version "702" in the RELEASE field.



Message Type	Basic type	Extension	Release
ORDERS	ORDERS05	ZORDRS01	702

Figure 8.30 Link between the Extension and the Message and Basic Types

3. Save the entry and assign it to a transport request.

8.3.5 Create the Modification Project

Next we'll create the modification project with Transaction CMOD, as shown in Figure 7.35.

1. Enter the project name "ZEDIMPO1" in the PROJECT field and click CREATE. The project attributes screen opens. Enter a description in the SHORT TEXT field (see Figure 8.31).

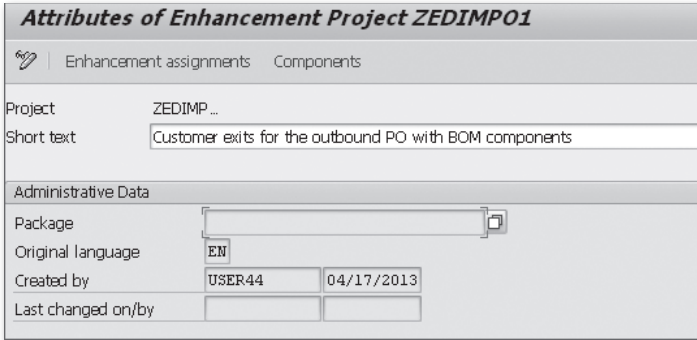


Figure 8.31 Modification Project ZEDIMP01 Attributes

2. Save the project and assign it to a change request.
3. Click the ENHANCEMENT ASSIGNMENTS button. Enter “MM06E001” into the ENHANCEMENT field (see Figure 8.32). Save.

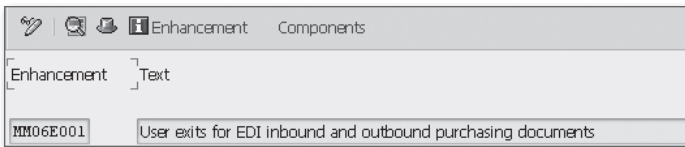


Figure 8.32 Our User Exit Is in Enhancement MM06E001

4. Click COMPONENTS to load a list of all exit functions assigned to the enhancements. A partial list is displayed in Figure 8.33.

The checkmark next to the enhancement means that there is already active code in the exit.

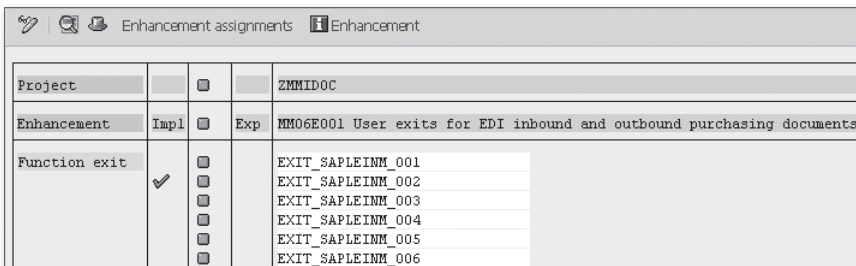


Figure 8.33 Exit Functions Are Components of Enhancements

- We'll write the code in component EXIT_SAPLEINM_002. Double-click the exit to open the source code window of the Function Builder.

Note the import and table parameters of the exit function in Figure 8.34. We'll read our inputs from the application tables and append our new segment to INT_EDIDD when we write our code.

INT_EDIDD is used to assemble the IDoc segments in the order that they will appear in the finished IDoc.

```

Function module EXIT_SAPLEINM_002 Active
Attributes Import Export Changing Tables Exceptions Source code
1 FUNCTION EXIT_SAPLEINM_002.
2  *"--
3  *"--Global Interface:
4  *"-- IMPORTING
5  *"-- VALUE(XEKKO) LIKE EKKO STRUCTURE EKKO
6  *"-- VALUE(XLFA1) LIKE LFA1 STRUCTURE LFA1
7  *"-- VALUE(XLFB1) LIKE LFB1 STRUCTURE LFB1
8  *"-- VALUE(DOBJECT) LIKE NAST STRUCTURE NAST OPTIONAL
9  *"-- TABLES
10 *"-- INT_EDIDD STRUCTURE EDIDD
11 *"-- XEKPO STRUCTURE UEKPO OPTIONAL
12 *"-- XEKET STRUCTURE UEKET OPTIONAL
13 *"-- DEKEK_X STRUCTURE EKEK_X OPTIONAL
14 *"-- DEKEH STRUCTURE IEKEH OPTIONAL
15 *"-- DSADR STRUCTURE SADR OPTIONAL
16 *"-- DVBK STRUCTURE MMVBK OPTIONAL
17 *"-- DVBAP STRUCTURE MMVBAP OPTIONAL
18 *"-- DVBKD STRUCTURE MMVBKD OPTIONAL
19 *"-- CHANGING
20 *"-- VALUE(ISC_ENHANCEMENT) TYPE ISC_EXIT_SAPLEINM_002 OPTIONAL
21 *"-- EXCEPTIONS
22 *"-- ERROR_MESSAGE_RECEIVED
23 *"-- DATA_NOT_RELEVANT_FOR_SENDING
24 *"--
25
26
27 INCLUDE ZXM06U02.
28
29
30 ENDFUNCTION.

```

Figure 8.34 Exit Parameters in the Source Code of the Customer Exit

We first need to create the include program ZXM06U02, if it doesn't already exist. When you double-click the include name, SAP returns the following information message:

Program names ZX... are reserved for includes of exit function groups.

Press **Enter** to get past the message and open the CREATE OBJECT dialog informing you of the following:

Include ZSM06U02 does not exist. Create object?

Click YES and assign the include program to a change request. When the ABAP editor loads, we're ready to code.

8.3.6 Coding the Exit

The code for our exit will be called after each instance of segment E1EDP01 is processed and added to the IDoc.

The exit will read the BOM for the purchase order line item material from table MDSB, which stores the material number for the item components against the PO and line item numbers. We can then get the description for each component from table MAKT.

The material number and description for each component will be passed to a separate instance of our custom segment Z1EDP01 and appended to internal table INT_EDIDD, which is used to build the IDoc.

The data that we need to identify the BOMs for the finished DVDs in the PO are in the table parameters of EXIT_SAPLEINM_002.

The IDoc is built from data in a number of structures and internal tables, including XEKKO (purchase order header) and XEKPO (line item details), which will give us the PO number and line item number to read MDSB. The IDoc build for each segment—and the customer function call for each segment—is in form FUELLEN_IDOC_INTTAB.

All line item segments are built within a loop on XEKPO. E1EDP01, the parent segment at the item level, is the first appended to the IDoc within the XEKPO loop. Our code will be called by the exit after E1EDP01 is written to internal table INT_EDIDD.

The same exit is called for every segment after it has been written to the IDoc, regardless of level. So we need to check that the current segment name is E1EDP01. This is in field SEGNAM in internal table INT_EDIDD.

In addition, we're only doing this for DSI. So we also need to check that the PO is for vendor DISK01. This value will be available to the exit in the import parameter LFA1-LIFNR.

This, and the custom segment name, is all we need to get the BOM and append it to the IDoc as a child of E1EDP01. The code is shown in Listing 8.3. Once again,

remember that the code is stripped down to its essentials and is really only a starting point for this solution.

```

*&-----*
*& Include          ZXM06U02                      *
*&-----*
** Type declarations                               *
** Component material numbers from MDSB
TYPES: BEGIN OF t_bomrec,
        matnr type matnr,
        END OF t_bomrec.
** Internal table declaration                       *
** BOM data for custom segment data:
it_pobom type table of t_bomrec.
** Data Declarations                               *
** Work areas
data: wa_pobom like line of it_pobom,
        ls_z1edp01 type Z1EDP01,
        ls_maktx type maktx.
** Begin processing                               *
** Restrict processing to segment E1EDP01 and vendor DSI
if x1fa1-lifnr = 'DISK01' and int_edidd-segnam = 'E1EDP01'.
** Get all BOM components and descriptions for finished good
** in current purchase order item.
        select matnr into corresponding fields
                of table it_pobom from mdsb
                where ebeln = xekpo-ebeln
                and ebelp = xekpo-ebelp.
** Get material description by looping on it_pobom and
** reading MAKT by material number
        loop at it_pobom into wa_pobom.
                clear ls_maktx.
                select single maktx into ls_maktx
                        from maktx where matnr = wa_pobom-matnr.
                if sy-subrc = 0.
** Populate custom segment string
                ls_z1edp01-matnr = wa_pobom-matnr.
                ls_z1edp01-maktx = ls_maktx.
                endif.
** Pass and append data to INT_EDIDD
        int_edidd-segnam = 'Z1EDP01'.
        int_edidd-sdata = ls_z1edp01.

```

```

        append int_edidd.
    endloop.
endif.

```

Listing 8.3 Exit Code Passes BOM Components to Segment Z1EDP01

8.3.7 Customize Message Control

Now we move on to outbound configuration: message control and a partner profile. We'll need a custom output type so that we can enter unique values into the message control screen of our outbound partner profile.

Message type ORDERS will be used in more than one outbound interface for DSI and will require more than one outbound partner profile. We need a unique output type for our BOM ORDERS interface that we will not use in any other partner profile.

Each partner profile must be unique, as we have seen from our discussions of partner profile and message control keys in tables EDP13 and EDP12 in Chapter 7, Section 7.2, Outbound Configuration Generates IDocs.

To create our custom output type ZNEU, we'll copy standard type NEU in application EF (purchase order). We'll only use this custom type to generate an outbound ORDERS IDoc from a PO with the BOM extension.

1. Call Transaction NACE.
2. Select application EF (purchase order) and click OUTPUT TYPES to open the OUTPUT TYPES: OVERVIEW in display mode.
3. Select the menu path TABLE VIEW • DISPLAY • CHANGE (or press **Ctrl** + **F1**) and then select standard output NEU. Click COPY AS or press **F6**.
4. Change the name of the output type to "ZNEU" and enter a description for the extended PO with BOM. Make sure the ACCESS TO CONDITIONS and MULTIPLE ISSUING checkboxes are both set, as in Figure 8.35.
5. Access sequence 0001 (document type, purchasing organization, and vendor) gives us enough granularity to restrict who gets a PO with BOMs from which purchase order.

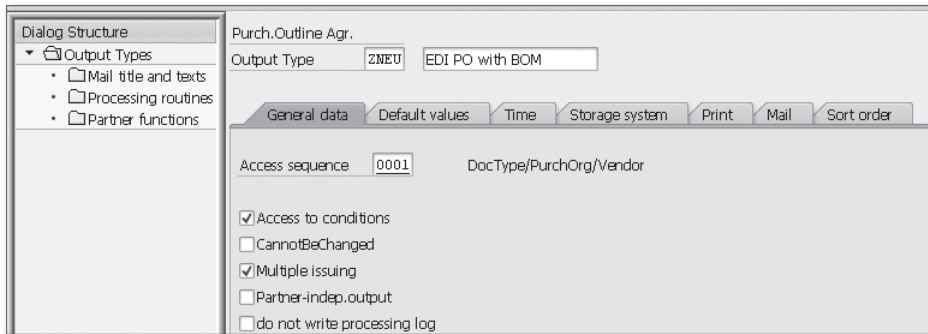


Figure 8.35 CustomOutput Type ZNEU with Access Sequence 0001

- We'll use a custom PO document type (ZNB) copied for standard PO document type NB to make sure that only some POs are sent to DSI with an extended BOM segment. We'll use ZNB to create purchase orders that will be sent to DSI with the BOM in the extended IDoc type.

Creating a Custom Purchasing Document Type

Not all purchase order IDocs will be sent with a custom BOM segment. We need to distinguish between orders that will generate a BOM segment and those that won't. To do this, we'll create a custom purchase order document type and use this in our condition record.

Functional consultants and business users define custom document types after thorough analysis of how these documents will be used by the business and what data they need to contain in their different use cases.

It's not really the job of the EDI team, but it doesn't hurt to know.

Purchasing document types are configured in the IMG (Transaction SPRO). We'll take the easy road and copy the existing standard PO document type to a custom code.

- In the IMG, follow menu path SAP CUSTOMIZING IMPLEMENTATION GUIDE • MATERIALS MANAGEMENT • PURCHASING • PURCHASE ORDER • DEFINE DOCUMENT TYPES.
- In the table control of the DOCUMENT TYPES PURCHASE ORDER CHANGE screen, find and select document type NB (standard PO).
- Click the COPY As button (or press **F6**).
- Change the document type code to ZNB and add a short description.
- Press **Enter**. The SPECIFY OBJECT TO BE COPIED dialog opens with the message that the document type has dependent entries.
- Click COPY ALL to get an exact copy of the standard document type.
- Save and assign the custom document type to a transport request.

Our custom purchasing document type can now be used to create purchase orders for whoever needs an EDI PO with the BOM attached.

7. We will select access number 10 in access sequence 0001, which points to condition table 25 (B025) and communications structure KOMKBEA with key fields:
 - ▶ BSART: Purchasing document type
 - ▶ EKORG: Purchasing organization
 - ▶ LIFNR: Vendor
8. Press . The SPECIFY OBJECT TO BE COPIED dialog opens with the observation that the output type has dependent entries. Click COPY ALL. Another dialog opens with the number of dependent entries.
9. The system returns us to the OUTPUT TYPES: OVERVIEW screen in change view. Select output ZNEU, and double-click the PROCESSING ROUTINES folder to open the PROCESSING ROUTINES: OVERVIEW screen.
10. If there is no program name for medium EDI, add one. Click NEW ENTRIES to open the DETAILS OF ADDED ENTRIES screen.
11. Enter the following values into the added entries details screen:
 - ▶ Enter "EDI" in the TRANSM.MEDIUM field.
 - ▶ Enter "RSNASTED" in the PROGRAM (Processing 1) field; this is the standard SAP output program. You can also use a custom ABAP output program. Use RSNASTED as a model.
 - ▶ Enter "EDI_PROCESSING" in the Form ROUTINE field. This form identifies and calls the function that will build the IDoc, writes it to the database, sends it to the outbound IDoc processing function, and updates its status record.
12. Press and double-click the PARTNER FUNCTIONS folder to open the PARTNER FUNCTIONS: OVERVIEW screen.
13. Make sure that there's an entry for medium EDI and partner type VN for vendor. If not, click NEW ENTRIES and select EDI in the MEDIUM field and VN in the FUNCT field.
14. Save and assign the new output to a change request. You may need to cycle through several objects to complete the assignment to the request.

Assign ZNEU to a Procedure

1. Back out to the initial output control screen in Transaction NACE. Select application EF and click the PROCEDURES button.
2. There's only one procedure in application EF, and that is RMBEF1. We'll need to assign our custom output type to it.
3. Select the procedure and double-click the CONTROL folder in the navigation pane.
4. Select output type NEU and click the COPY AS button (or press **F6**). The entry is copied into the CHANGE VIEW CONTROL: OVERVIEW screen.
5. Change the step number to "15" and the output type name to "ZNEU", but leave everything else the same.
6. Press **Enter** to return to the overview screen. Save the entry and assign it to a change request. The entry should look like Figure 8.36.

Step	C...	C...	Description	Requirement	Manual on
10	1	NEU	New PO printout		<input type="checkbox"/>
15	1	ZNEU	EDI PO with BOM		<input type="checkbox"/>

Figure 8.36 Assigning ZNEU to the Purchasing Procedure

Create Condition Records

Condition records drive generation of the IDoc for Acme's vendor, DSI. In this case, the IDoc will only be generated for PO document type. Follow these steps:

1. In Transaction NACE, select application EF and click CONDITION RECORDS.
2. Select OUTPUT TYPE "ZNEU". Click the CONDITIONS RECORDS button. The KEY COMBINATION dialog opens. Select the second key combination, as shown in Figure 8.37.

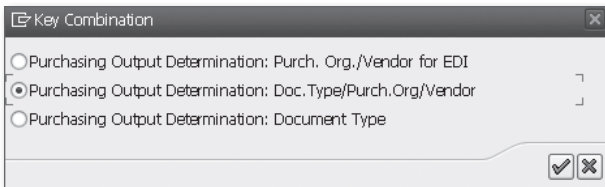


Figure 8.37 Select the Access Sequence in the Key Combination Dialog

3. Press or click the green checkmark. The selection screen for the condition record opens. Enter the following values and click EXECUTE to open the condition record table control (see Figure 8.38):
 - ▶ Enter "ZNB" for EDI PO with BOM enhancement in the PURCHASE DOC. TYPE field.
 - ▶ Enter "3000" (sample) in the PURCH. ORGANIZATION field.
4. In the condition records table control, enter the following:
 - ▶ "DISK01" for Acme's vendor Disc Services International
 - ▶ "VN" for vendor as the function
 - ▶ "6" for EDI as the medium
 - ▶ "4" for immediate when the document is saved as the dispatch time
 - ▶ "EN" as the language

Purchasing Doc. Type	ZNB	EDI PO with BOM				
Purch. Organization	3000	IDES USA				
Condition Recs.						
Vendor	Name	Fu...	Partner	Medium	Date/Time	Language
DISK01	Disc Services Inter...	VN		6	4	EN

Figure 8.38 Condition Record for Output Type ZNEU in Application EF

Note that the full access key for the condition record is present in the header and table control area of the entry.

8.3.8 Build Outbound Partner Profile

All that's left is to set up a unique outbound partner profile for DSI using our logical message, extended type, and custom message control.

Message code BOM will create the unique key in table EDP12 for our partner profile. The message code links to the logical message and process code. The standard process code for the PO is ME10, which links message type ORDERS to IDoc function IDOC_OUTPUT_ORDERS. And this is where we coded our customer exit to populate the BOM extension.

We have two choices: either create a custom process code or extend the standard to include message code BOM. We'll extend the standard ME10.

Extending the Process Code

We can extend process code ME10 with Transaction WE41 or by following WEDI area menu DEVELOPMENT • OUTBOUND PROCESSING SETTINGS/MC • OUTBOUND PROCESS CODE.

1. Select process code ME10 and double-click the LOGICAL MESSAGE folder in the navigation pane to open the LOGICAL MESSAGE DETAILS screen.
2. Click NEW ENTRIES (or press **F5**) and add the following values to the added entries screen:

- ▶ MESSAGE TYPE: ORDERS
- ▶ MESSAGE CODE: BOM

We could select the ALL CODES radio button under MESSAGE CODE and the system would pass any value found in the message code field, but being specific gives us better control over our partner profile keys.

3. Save and assign the entry to a change request. The screen should look like Figure 8.39.

The screenshot shows a configuration window for 'Assignment to logical message'. At the top, the 'Process code' is 'ME10' and the description is 'ORDERS: Purchase order'. Below this, there are three sections, each with a radio button and a text field:

- Message type:** The radio button is selected, and the text field contains 'ORDERS'. To the right, the description 'Purchase order / order' is visible.
- Message code:** The radio button is selected, and the text field contains 'BOM'.
- Message function:** The radio button is selected, and the text field is empty.

Figure 8.39 Linking Logical Message ORDERS to Message Code BOM

4. There should now be two entries for message ORDERS in the LOGICAL MESSAGE table control in the LOGICAL MESSAGE OVERVIEW screen, as in Figure 8.40.

Process code	ME10					ORDERS: Purchase order
Logical message						
	Message Type	All types	Msg. Var.	All codes	Msg.funct.	All functions
	ORDERS	<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>
	ORDERS	<input type="checkbox"/>	BOM	<input type="checkbox"/>		<input type="checkbox"/>

Figure 8.40 Process Code Extended with Multiple Instances of ORDERS

Creating the Partner Profile

Go to Transaction WE20 and create the header level partner profile for DSI in the PARTNER TYPE LI folder in the navigation pane, if one doesn't already exist.

1. Click CREATE OUTBOUND PARAMETER beneath the outbound parameters table control and enter the following values that are shown in Figure 8.41:
 - ▶ PARTNER ROLE: "VN"
 - ▶ MESSAGE TYPE: "ORDERS"
 - ▶ MESSAGE CODE: "BOM"

Partner No.	DISK01	Disc Services International
Partn. Type	LI	Vendor
Partner Role	VN	Vendor
Message Type	ORDERS	Purchase order / order
Message code	BOM	
Message function	<input type="checkbox"/>	<input type="checkbox"/> Test
Outbound Options Message Control Post Processing: Permitted Agent Telep...		
Receiver port	XML_IDOC	XML File Test port for XML IDoc output
Output Mode		
<input type="radio"/> Transfer IDoc Immed.	<input checked="" type="radio"/> Start subsystem	Output Mode 3
<input checked="" type="radio"/> Collect IDocs	<input type="radio"/> Do not start subsystem	
IDoc Type		
Basic type	ORDERS05	Purchasing/Sales
Extension	ZORDRS01	Outbound purchase order wit...
View		
<input checked="" type="checkbox"/> Cancel Processing After Syntax Error		

Figure 8.41 Outbound Parameters for the BOM PO IDoc

- ▶ RECEIVER PORT: "XML_IDOC"
 - ▶ OUTPUT MODE: The radio buttons for COLLECT IDocs and START SUBSYSTEM
 - ▶ BASIC TYPE: "ORDERS05"
 - ▶ EXTENSION: "ZORDRS01"
2. Click on MESSAGE CONTROL and add two entries with the following values that are shown in Figure 8.42:
- ▶ APPLICATION: "EF"
 - ▶ MESSAGE TYPE: "ZNEU"
 - ▶ PROCESS CODE: "ME10"
 - ▶ CHANGE MESSAGE: One checkbox entry null and one checked

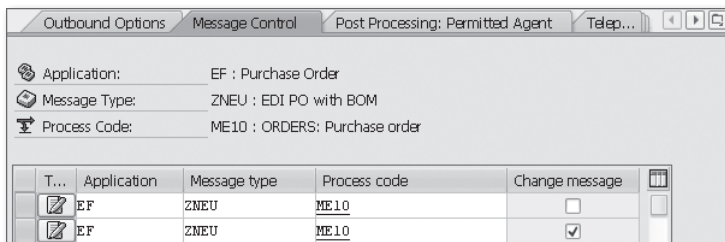


Figure 8.42 Message Control Values for the BOM ORDERS

3. We'll also need EDI information for the IDoc control record. All of Acme's IDocs will include this data. Select the EDI STANDARD tab, and enter the following values:
- ▶ EDI STANDARD: "X" for X12
 - ▶ MESSAGE TYPE: "850" for outbound X12 850
 - ▶ VERSION: "005010"
4. Save the partner profile.

This extended IDoc is now ready to test. Feel free to play with it, tweak it, extend it, and so on. This is only the starting point.

8.4 Summary

Now we've had a brief introduction to building custom and extended IDocs. We've toured the key IDoc development and configuration tools and have seen how they can be used to build a simple custom inbound and an extended outbound IDoc. We've also touched on enhancements and modification projects and have seen how simple it can be to find a specific user exit and enhancement using the Data Dictionary.

But there's so much more to know, so little time, and so few pages to learn it all, which is a major problem in any implementation project. We need to learn enough quickly enough to do the work that needs to be done. So our next step is to put our brief introduction to this material to work. Our legs may still be wobbly, but we'll take the plunge and begin building Acme's new SAP EDI system, at least the order-to-cash cycle of interfaces, beginning with the inbound customer purchase order.

Besides, the great Darryl Q. Fernhausen often began a new picture knowing even less about the plot, the actors, or the writers. So let's open the curtain on Act III and prepare to take the stage.

Index

A

- ABAP Data Dictionary, Data objects
 - Data elements*, 85, 231, 232
 - Domains*, 85, 231, 232
 - Fields*, 85, 231, 232, 233
 - Segments*, 85, 233
- ABAP Data Dictionary, IDoc architecture, 230
- ABAP Programs, Business application
 - RV60SBAT Batch billing jobs*, 642, 655
 - RVV50R10C Delivery due list*, 549, 562, 565
 - SAPMV45A Create sales order*, 496
 - SAPMV50A Delivery module pool*, 562
 - SDBILLDL Billing due list*, 656
 - SPLIT_PAYMENT_ADVICE*, 709, 711, 716, 717, 724
- ABAP Programs, Custom IDoc
 - ZEDI_ORDRSPSDQ Create SDQ ORDRSP*, 482, 483, 488, 498, 500, 509–533, 541, 543
 - ZEDI_TRNSFIDOCs ALE IDoc transfer*, 764, 765, 766, 771
 - ZEDI_UPLDPP Load partner profiles*, 749–756, 762, 763, 768
 - ZSDCHINVOIC ALV grid update report*, 641, 665, 667, 668, 688
- ABAP Programs, IDoc functions
 - SAPLEINM Purchasing messages*, 314, 347
- ABAP Programs, IDoc processing
 - RBDAPP01 Post inbound IDocs*, 251, 268, 270, 369, 383, 397, 405, 407, 431, 443, 452, 456, 577, 585–589, 705, 715, 718, 882
 - RSEOUT00 Output batched IDocs*, 306, 369, 382, 383, 431, 484, 498, 500, 549, 563, 565, 607, 624, 626, 642, 643, 662, 664, 792, 856, 879, 885, 887
 - RSNAST00 Selection program for issuing output*, 380, 792
- ABAP Programs, IDoc processing (Cont.)
 - RSNASTED Output IDocs from table NAST*, 282, 303, 305, 317, 318, 357, 380, 492, 563, 616, 653, 659
- ABAP Programs, IDoc utilities
 - MSEIDOC0 Turnaround utility*, 794
 - MSEIDOC1 Status test file*, 793
 - RBDINPUT Inbound processing all*, 866
 - RBDMON00 IDoc status monitor*, 852
 - RBDOUTPU Outbound processing all*, 866
 - RBDSTATE Send ALEAUD confirmations*, 772–777
 - RSEIDOC2 IDoc list*, 868
 - RSEIDOC9 Search IDoc by field content*, 872
 - RSEIDOC9 Search IDocs by field contents*, 204
 - RSEXARCA Archive IDocs*, 201
 - RSEXARCD Delete IDocs from Archive*, 203
 - RSEXARCL Search archive for IDoc*, 204
 - RSEXARCR Archive read*, 204
 - RSWWHIDE Delete work item history*, 204
 - RSWWHIDE Mass delete work items*, 204
 - SAPMSSEDIPARTNER Maintain partner profiles*, 750
- ABAP Programs, Load master data
 - RCSBI010 Bill of Materials*, 93
 - RFBIDE00 Customers*, 89
 - RFBIKR00 Vendors*, 90
 - RMDATIND Materials*, 91
 - RV14BTCL Pricing conditions*, 94
- Acme legacy systems, 63–69
- ALE custom IDoc transfer program logic, 765–771
- ALV Grid List Report Logic, 666–693
- ALV List Report Logic, 513, 516, 517, 531, 532
- American National Standards Institute (ANSI), 119
 - ANSI ASC X12*, 37
- Application errors inbound, 848, 849
- Application errors outbound, 849, 850

Application Link Enabling (ALE), 156
 Archiving
 Deleting workflow tasks in SAP, 204, 205, 206, 207
 EDI system strategy, 169, 171
 IDoc archive configuration, 198, 199, 201
 IDocs in SAP, 196, 198, 199, 201, 202, 203, 204
 IDoc status codes that can be archived, 197
 Workflow tasks in SAP, 196
 AS2 (Applicability Statement 2), 37
 AS2 server services, 162
 AS2 text header, 163

B

Billing due list, 106, 111, 640, 643, 655, 656, 700
 Blueprint questions, 51
 BPML, 83, 160, 162
 BPML code sample
 SAPAdapter, 44
 Business document
 ASN (Advanced Ship Notification), 59
 Billing, 107, 277, 640, 643, 644, 645, 649, 650, 651, 658, 700
 Customer invoice, 40, 59
 Customer purchase Order, 38
 Delivery, 277, 549–598, 606–615, 620, 636, 656, 658
 Material document, 397–406
 Sales order, 274, 277, 302, 442, 443, 446, 450, 457, 471, 479, 550, 561–577, 597, 606, 610, 620, 636–646, 648, 658, 659, 700
 Ship confirmation, 59
 Supplier invoice, 38, 413–420
 Supplier purchase order, 103, 150, 277, 367–378
 Supplier purchase Order, 37
 Business Object Repository (BOR), 322
 Business process
 Billing, 95, 106–108
 Catalog planning, 95, 100, 101
 Delivery, 95, 104–106
 Payment, 95, 108–110

Business process (Cont.)
 Purchasing, 37, 52, 95, 102–104
 Purchasing, catalog release, 55, 56
 Purchasing, new release, 53
 Replication, 70
 Sales, 38
 Sales and distribution, 57
 Business Process Execution Language (BPEL), 43
 Business Process Modeling Language (BPML), 43
 Business process workflow
 Inbound IDoc processing steps, 43, 44, 45

C

Communications IDoc, 225
 Communications protocols
 AS2, 69, 72, 73, 77, 83, 124, 150, 154, 162, 195
 FTP, 71, 74, 77, 83
 FTP/S, 83
 HTTP, 83
 HTTP/S, 72, 73, 83, 163
 VAN, 69, 86
 Configuring ALE send of status confirmation IDoc, 773, 774, 775, 776, 777
 Creating a custom IDoc segment, 348
 Creating a modification project, 350, 351, 353
 Custom 846 Inventory Report, 324–341
 Custom ABAP program ZSDCHINVOIC logic flow, 665–694
 Customer exits, 98
 Customer order lifecycle, 94–110
 Custom table ZEDIXREF, structure, 262

D

Data conversion strategy, 87, 88, 89, 90, 92, 93, 94
 Delivery due list, 105
 Development workflow
 Create custom IDoc, 324, 325

Development workflow (Cont.)

Create table ZEDIXREF, 741–749

Extend IDoc basic type, 341, 343

Domain values, qualifiers, 231

E

EAN (European Article Number), 91

EDI

Architecture, 47

Batch processing, 46

Codes and qualifiers, 136

Implementation guidelines, 130

Metadata formats, 160

EDI correlation data, 160, 161, 168

EDI correlation data inserts/reads, 179, 180, 181

EDI envelope

GS group, 71

ISA interchange, 71

ST transaction set, 71

EDIFACT compared to IDocs, 126, 127

EDIFACT compared to X12, 129, 130

EDIFACT messages

CONTRL Syntax and service report, 186

DESADV Despatch advice, 126

INVOIC Invoice, 121, 126

ORDERS Purchase order, 126

REMAADV Payment advice, 126

EDIFACT messages anatomy, 125, 126, 127

EDI history

Accredited Standards Committee (ASC)

X12, 119

Berlin Airlift, 117, 118

DISA (Data Interchange Standards Association), 118, 119

Electronic Data Interchange For Administration, Commerce and Transport, 43

FTD (Florists' Telegraph Delivery), 117

Overview, 116, 117, 118, 119, 120, 121

Transportation Data Coordinating Committee (TDCC), 118

Uniform Communication Standard (UCS), 119

EDI interface

De-enveloping, 73, 146, 147, 165

Enveloping, 71, 147, 148, 160, 161

Legacy inbound process flow, 72–75

Legacy outbound process flow, 70–72

Process flow, 150

SDQ segment, 41, 62, 97, 99, 111, 143–145, 443–479

Status code, 72, 167, 168

Trading partner, receiver, 72, 73, 146, 159, 161, 165

Trading partner, sender, 72, 73, 146, 159, 161, 165

Transaction set identifier code, 72

EDI mapping strategy, 48

EDI reference data, SAP

Inbound, 252–263

Outbound, 296–299

EDI standards

ASC X12, 86, 118–120

EDIFACT, 86, 120, 121

ODETTE, 122

TRADACOMS, 86, 122

UN/GTDI guidelines, 121, 122

VDA, 123

EDI system

Adapters, 43

Business process modeling tool, 43

SAP IDoc adapter, 44

Services, 43

EDI trading partner management, 86, 87

Electronic Data Interchange (EDI), 33

Enhancements described, 319, 320

Extending an IDoc basic type, 348, 350, 501–506

Extensible Markup Language (XML), 40

F

Field symbols, 525

File Transfer Protocol (FTP), 71

Functional acknowledgment data, 161, 166, 168

Functional specs

ASN, 608–615

Customer invoice, 642–651

- Functional specs (Cont.)
 - Customer PO*, 443–448
 - Goods receipt/goods issue*, 398–402
 - Payment advice*, 706–712
 - PO confirmation*, 483–489
 - Ship confirm*, 577–585
 - Ship order*, 549–557
 - Supplier invoice*, 416–426
 - Supplier PO*, 370–374
- Function groups, ALV processing
 - SALV ALV list viewer*, 532
- Function groups, Application processing
 - FRAD Payment advice*, 109
 - MEGUI PO user interface*, 380
 - MEPO Purchase order business logic*, 380
 - V50K Delivery BAPIs*, 562
 - V50R_VIEW Delivery due list*, 562
 - V60A Billing*, 658
- Function groups, Custom
 - ZEDINVRP Inventory report*, 332
 - ZVEDI ZEDIXREF maintenance screen*, 741, 743, 744
- Function groups, IDoc output
 - EDI7 EDI communications*, 381
 - EINM Output purchasing messaging*, 314
 - EINV On commit calls*, 380
 - V61B Output control conditions*, 286, 562
 - V61Z Conditions general determination*, 379
 - V70A Processing output list*, 380
- Function groups, IDoc processing
 - EDFI2 Maintain table EDFI2*, 266
 - EDI1*, 522
 - EDI5*, 522
 - EDI6 Maintain partner profiles*, 750
 - EDIN Inbound processing*, 266
- Function groups, IDoc utilities
 - EDIMEXT IDoc type API*, 84
- Function modules, ALV processing
 - REUSE_ALV_HIERSEQ_LIST_DISPLAY*, 517, 532
 - REUSE_ALV_LIST_DISPLAY*, 757, 761
- Function modules, Application processing
 - BAPI_GOODSMVT_CREATE*, 104, 405
 - BAPI_REQUIREMENTS_CHANGE*, 101
 - BAPI_REQUIREMENTS_CREATE*, 101
- Function modules, Application processing (Cont.)
 - GUI_UPLOAD*, 758
 - IDOC_INPUT_DELVRY*, 577, 586, 593, 602
 - IDOC_INPUT_INVOIC_FI*, 425
 - IDOC_INPUT_INVOIC_MRM*, 414, 425, 428, 437
 - IDOC_INPUT_MBGMCRCR*, 397, 405, 406
 - IDOC_INPUT_ORDERS*, 250–273, 316, 319, 321, 442, 453, 471–478
 - IDOC_INPUT_REMADV*, 322, 705, 715, 735
 - IDOC_INPUT_REMADV_CTR*, 735
 - MB_CREATE_GOODS_MOVEMENT*, 405
 - MRM_INVOICE_CHECK*, 429
 - MRM_INVOICE_CREATE*, 425, 429
 - MRM_INVOICE_PARK*, 429
 - MRM_INVOICE_POST*, 429
 - NREL_GET_NEIGHBOURHOOD*, 454
 - PARTNER_CONVERSION_INT_TO_EXT*, 623
 - REMADV_INSERT*, 716, 731
 - RV_CUSTOMER_MATERIAL_READ*, 260
 - RV_INVOICE_ADD*, 658
 - RV_INVOICE_CREATE*, 658
 - RV_PRICE_PRINT_ITEM*, 660
 - RV_READ_INVOICE_INDEX*, 657
 - RV_SALES_DOCUMENT_ADD*, 302, 496
 - SD_COLLECTIVE_RUN_EXECUTE*, 658
 - SD_OBJECT_TYPE_DETERMINE*, 305
 - SHP_BAPI_DELIVERY_REPLICA*, 562
 - SHP_EXTENDED_DUE_LIST_VIEW*, 562
 - SPLIT_PAYMENT_ADVICE*, 109
 - VIEW_KNVV*, 298
 - WS_DELIVERY_UPDATE_2*, 587
- Function modules, Custom development
 - RV_CUSTOMER_MATERIAL_UPDATE*, 92
- Function modules, IDoc development
 - EDI_AGREE_IN_MESSTYPE_INSERT*, 316, 750, 761
 - EDI_AGREE_IN_MESSTYPE_UPDATE*, 750
 - EDI_AGREE_OUT_IDOC_INSERT*, 316, 750, 761
 - EDI_AGREE_OUT_IDOC_UPDATE*, 750, 761
 - EDI_AGREE_OUT_MESSTYPE_INSERT*, 315, 750, 760

- Function modules, IDoc development (Cont.)
- EDI_AGREE_OUT_MESSTYPE_UPDATE*, 750, 760
 - EDI_AGREE_PARTNER_INSERT*, 315, 750, 759
 - EDI_PARTNER_CREATE_SYPART01*, 762
 - EDI_PARTNER_SEND_IDOC*, 762
 - ZIDOC_INPUT_ZINVRPT* Post inventory report, 333–339
- Function modules, IDoc output
- COMMUNICATION_AREA_KOMKBEA*, 378
 - COMMUNICATION_AREA_KOMKBV1*, 302, 496
 - COMMUNICATION_AREA_KOMKBV2*, 562, 620
 - COMMUNICATION_IDOC_CREATE*, 305
 - EDI_PARTNER_READ_OUTGOING*, 304, 497, 563, 621, 659
 - IDOC_OUTPUT_DELVRY*, 549, 563, 572, 607, 613, 621, 623, 633, 636
 - IDOC_OUTPUT_INVOIC*, 316, 641, 659, 662, 700
 - IDOC_OUTPUT_ORDERS*, 300, 314, 345, 347, 359, 368, 375, 381, 482, 563
 - IDOC_OUTPUT_ORDRSP*, 294, 304, 368, 392, 497
 - KOMKBV1_FILL*, 286, 302
 - MASTER_IDOC_CREATE_BOMMAT*, 316
 - MESSAGING*, 302, 496, 562, 620
 - RV_MESSAGES_UPDATE*, 303
 - SD_COND_ACCESS*, 302, 379
- Function modules, IDoc processing
- ALE_FTCH_DATA_SEGMENTS_OF_IDOC*, 270
 - APPLICATION_IDOC_POST_IMMEDIAT*, 270
 - BAPI_IDOC_INPUT1*, 411
 - EDI_CHANGE_DATA_SEGMENT*, 692
 - EDI_CONTROL_RECORD_MODIFY*, 267
 - EDI_DATA_INCOMING*, 45, 159, 169, 186, 193, 229, 266, 268, 404, 452, 586, 715, 771, 873, 877
 - EDI_DOCUMENT_CLOSE_CREATE*, 268
 - EDI_DOCUMENT_CLOSE_EDIT*, 692
 - EDI_DOCUMENT_CLOSE_READ*, 520, 677, 770
- Function modules, IDoc processing (Cont.)
- EDI_DOCUMENT_OPEN_FOR_CREATE*, 268, 305, 747
 - EDI_DOCUMENT_OPEN_FOR_EDIT*, 692
 - EDI_DOCUMENT_OPEN_FOR_READ*, 519, 676, 770
 - EDI_DOCUMENT_STATUS_SET*, 268
 - EDI_DOCUMENT_TREE_DISPLAY*, 693, 856
 - EDI_EDFI2_DELETE*, 266, 268
 - EDI_EDFI2_READ*, 266
 - EDI_OUTPUT_NEW*, 156, 230, 307, 381, 624, 662
 - EDI_PATH_CREATE_MESTYP_DOCNUM*, 239
 - EDI_PORT_READ*, 230, 266, 307
 - EDI_SEGMENT_GET*, 520, 522
 - EDI_SEGMENTS_ADD_BLOCK*, 268
 - EDI_SEGMENTS_GET_ALL*, 677, 770
 - EDI_STATUS_INCOMING*, 793
 - IDOC_CCMS_OPEN*, 381
 - IDOC_CONTROL_OUTBOUND_CONVERT*, 226
 - IDOC_CREATE_ON_DATABASE*, 305, 747
 - IDOC_CTRL_INBOUND_CONVERT*, 226, 267
 - IDOC_DATA_INBOUND_CONVERT*, 226
 - IDOC_DATA_OUTBOUND_CONVERT*, 226
 - IDOC_INBOUND_ASYNCHRONOUS*, 771
 - IDOC_INBOUND_PROCESS_DATA_GET*, 267, 268, 271
 - IDOC_INBOUND_WRITE_TO_DB*, 267
 - IDOC_INPUT*, 271, 318
 - IDOC_READ_COMPLETELY*, 522, 523
 - IDOCS_OUTPUT_IN_XML_FORMAT*, 230, 238, 307
 - IDOCS_OUTPUT_TO_FILE*, 238
 - IDOC_START_INBOUND*, 270
 - IDOC_WORKITEM_INBOUND_CREATE*, 271
 - IDOC_XML_FROM_FILE*, 230, 266
 - MASTER_IDOC_DISTRIBUTE*, 156, 230, 498, 528, 692, 762, 766, 771
 - ME_MESSAGES_UPDATE*, 380
 - NUMBER_GET_NEXT*, 268
 - RFC_REMOTE_EXEC*, 238, 307

Function modules, IDoc processing (Cont.)

RV_MESSAGES_UPDATE, 380
SD_INT_TO_EXT_PARTNER_NUMBER,
 297, 298, 660
WFMC_MESSAGE_SINGLE, 380

Function modules, IDoc utilities

EDI_IDOC_SYNTAX_GET Get IDoc syntax,
 224
IDOC_RECORD_READ Get IDoc record
 types, 84
IDOCTYPE_READ_COMPLETE Get IDoc
 segments, 84, 224, 235, 872

Functions

Inbound IDoc API, 318
Outbound IDoc API, 317, 318
Use in IDoc interface, 315, 316

G

Generating IDocs with message control

ASN, 615–625
Invoice, 651–663
PO confirmation, 489–498
Ship order, 557–565
Supplier purchase order, 378–382

GLN (Global Location Number), 147, 446,
 452–476, 485, 502, 538

Goods receipt, 37

GTIN (Global Trade Item Number), 91

H

Hypertext Transfer Protocol (HTTP), 72

I

IDoc anatomy, 210–230

IDoc basic type, 83

ALEAUD01 ALE confirmations, 772,
 775
COND_A02 Pricing conditions, 94
DELVRY03 Delivery document, 548, 550,
 570, 576, 578, 586, 589, 590, 599,
 600, 607, 608, 632, 634, 636

IDoc basic type (Cont.)

INVOIC02 Invoice, 194, 414, 415, 418,
 428–436, 641, 669
MBGMCR03 Inventory movements, 397–410

IDoc basic type

ORDERS05 PO confirmation, 275, 294,
 299–306, 368, 369, 371, 382–389,
 482–487, 498, 501, 503, 506, 534
ORDERS05 Purchase order, 166, 223, 224,
 231, 263, 264, 266, 309, 312, 321,
 341, 348, 350, 362, 367–390, 441–443,
 445, 450, 455–471, 778, 779, 780,
 783, 784
PEXR2003 Payment advice, 705, 707, 717,
 718, 725, 735
SYPART01 Transport partner profiles, 248,
 762
SYSTAT01 IDoc status, 171
ZINVRPT01 Custom inventory report, 326,
 329, 331, 332, 337

IDoc basic type DELVRY03, Structure, 599,
 600

IDoc benefits, 45, 46

IDoc configuration

Condition records, 286
EDI user name, 235
Logical system, 236, 237
Partner profile, 48, 88, 97
Partner profile, header level, 246, 249
Partner profile, inbound, 246–252
Partner profile, outbound, 292–296, 359,
 361, 362
Partner type, 245, 246
Process code, 277, 435
RFC destination, 157, 237, 238, 239, 274
XML file port, 47, 158, 239, 240, 241

IDoc control record key fields, inbound, 244,
 263

IDoc control record key fields, outbound, 274

IDoc definitions

Basic type, 223, 224
Extended type, 225
IDoc file, 226
IDoc instance, 225

IDoc development

Identifying exits, 343, 344, 345, 347
Work Flow, 341

- IDoc development tools
 - ABAP Data Dictionary*, 310
 - Enhancement projects*, 320, 341, 342, 350
 - Enhancements*, 319, 320, 342, 344
 - Function editor*, 313–319, 325, 345, 347
 - IDoc attributes*, 322, 325
 - Inbound process code*, 322–325, 338
 - Link functions to IDocs*, 321–325, 337
 - Link message to basic type*, 313, 325, 331, 342, 350
 - Logical message type*, 313, 325, 331
 - Outbound process code*, 323, 324
 - Segment editor*, 101, 311, 312
 - Type editor*, 312, 313, 325, 329, 342, 348
 - IDoc EDI envelope key fields, 159, 194
 - IDoc enhancements, 98, 297
 - Bundling sales orders*, 99, 100
 - IDoc extended type, 225
 - Outbound PO with BOM segment*, 341–362
 - ZORDRS01 PO with BOM segment*, 341, 348, 350
 - ZORSQ01 PO confirm with SDQ segment*, 482–487, 498, 499, 501–510, 518, 523, 534, 541, 543
 - IDoc, Generate from business object, 101, 102
 - IDoc interface
 - Process flow inbound*, 264
 - IDoc interface development
 - ABAP ALV grid classes*, 666
 - IDoc message type, 94
 - List for Acme Pictures*, 211
 - IDoc metadata
 - EDIFECs gXML*, 84
 - Parser format*, 83
 - XML format*, 83
 - IDoc monitoring and recovery tools, 850
 - BD87*, 850–887
 - WE05*, 867–871, 877, 883
 - WE07*, 870, 871
 - WE08*, 873, 877
 - WE09*, 871, 872, 873
 - IDoc processing programs by status code
 - Inbound*, 864
 - Outbound*, 865
 - IDoc record types
 - ASCII external control*, 212
 - ASCII external data*, 215
 - ASCII external representation*, 212–216
 - ASCII external status*, 216
 - Control*, 84, 97, 159, 177, 212, 218, 219, 226, 244
 - Data*, 84, 212, 219, 220
 - Status*, 84, 212, 220–222
 - IDocs as intelligent messages, 127, 210, 211
 - IDocs inbound process flow, 264–273
 - IDocs outbound process flow, 300–307
 - IDoc status codes, 171, 172, 173
 - IDoc structures
 - Data element*, 84
 - Segment*, 84
 - IDoc test tools
 - startRFC*, 796–798
 - WE12 Turnaround utility*, 794, 795
 - WE14 Output batched IDocs*, 792, 814
 - WE15 Output from message control*, 791, 792
 - WE16 Inbound file processing*, 795, 796
 - WE17 Built test status file*, 793
 - WE18 Receive status file*, 793
 - WE19 Main test tool*, 788–796, 809
 - IDoc view
 - ZORSQ01_BAS PO confirm with SDQ segment*, 507, 509, 510, 543
 - Inbound processing requirements
 - ORDERS PO*, 264, 265
 - Intermediate Documents (IDocs), 34
 - Inventory of EDI interfaces, 79
-
- ## J
-
- Java, 42, 83
 - JDBC (Java Database Connectivity), 83
-
- ## L
-
- Legacy systems
 - DVD Repository*, 65, 82
 - EDI*, 69, 77
 - Finance*, 67, 81

Legacy systems (Cont.)

Manufacturing, 67, 74
Release planning and promotions, 66
Release Planning and Promotions, 80
Sales and Distribution, 66, 67, 76, 82
StoreData, 68, 75, 80, 97, 101, 153
Title Master, 65, 66, 80
VMI, 68, 75, 80, 110, 153

Links between SAP IDocs and EDI, 34

Link standard IDoc to custom function, 321

Logical message as abstraction of business document, 210

Logical message type

ADRMAS Partner addresses, 104
BLAOCH Purchasing contract change, 345
BLAORD Purchasing contracts, 345
BOMMAT BOM master, 97, 103, 316
COND_A Pricing conditions, 94
CREADV Credit advice, 322
DEBADV Debit advice, 322
DEBMAS Customer master, 97, 104
DESADV Outbound delivery/ASN, 97, 106, 111, 277, 578, 595, 607–637, 643, 832
INVOIC Customer invoice, 107, 111, 147, 181, 182, 194, 277, 641–652, 659, 662–665, 674, 694, 700, 853, 861, 885, 887
INVOIC Supplier invoice, 110, 414–436
MATMAS Material master, 97, 103
MBGMCR Inventory posting, 97, 103, 110, 397–410
ORDCHG Change purchase order, 345
ORDERS Customer purchase order, 97, 98, 110, 146, 164–168, 225, 226, 244–277, 302, 308, 441–478, 778–784, 809, 812, 817, 824, 843, 848, 852, 871, 882
ORDERS Supplier purchase order, 103, 110, 150, 309, 321, 344, 350, 355–375, 382, 383, 384, 388, 390
ORDRSP PO confirmation, 99, 103, 111, 155, 159, 274–277, 281, 287, 292–299, 300, 304–308, 368–392, 482–489, 498, 500, 506, 510, 521, 522, 529, 534, 538
REMAADV Payment advice, 109, 111, 322, 703–736, 817, 824, 835

Logical message type (Cont.)

REQOTE Request quotation, 345
SHPCON Ship confirm, 106, 111, 576, 578–590, 605, 609, 620, 636, 817, 824, 832, 849, 853
SHPMNT Outbound shipment, 595, 600
SHPORD Ship order, 105, 111, 549–570, 605, 610, 615
STATUS IDoc status in EDI system, 171, 177, 180, 184, 271, 382, 412, 772
SYPART Partner profiles, 248, 762, 763
WMMBXY Stock movements from external systems, 397, 405
ZINVRPT Custom invoice report, 326, 331, 332, 339, 340
ZREQTS Custom requirements feed, 101
 Logistics invoice verification, 413, 416, 417, 421–426, 437, 438
 Logistics invoice verification configuration, 421–426, 437

M

Mapping specs

ASN, 627–632
Customer invoice, 694–696
Customer PO, 457–465
Goods receipt/goods issue, 408–409
Payment advice, 725–731
PO confirmation, 534–537
Ship confirm, 590–593
Ship order, 568–570
Supplier invoice, 431–434
Supplier PO, 385–388

Mapping structures, overview, 83–86

Master data

Bill of materials, 82, 87, 93
Customer material info record, 87, 92
Customers, 82, 87
Finished goods, 82
Finished goods attributes, 82
GL accounts, 88
Materials, 87, 90, 92
Pricing conditions, 87, 93, 94
Raw materials, 82

Master data (Cont.)
Ship-to customer, 89, 90
Sold-to customer, 88
Suppliers, 82, 87, 90

MDN (message disposition notification), 72, 73, 124, 163–171, 192, 195, 624, 842, 874, 881

Message control
Access sequence, 280–302
Application, 276, 279, 288, 302
Condition records, 280, 286
Key fields, 278, 288, 289, 659
Output types, 277–284, 302
Procedures, 283, 284, 302
Requirements, 284, 286

Message control configuration
Supplier purchase order, 374–377

Message control configuration, condition records
Shipping, 654

Message control, create custom, 278–292, 355, 356, 357, 358

Message control, overview, 276, 277

Message type ORDERS as customer and supplier order, 211

Modifications and enhancements
MM06E001 EDI exits for purchasing, 344–353
SDEDI001 EDI exits for SD, 341
SIDOC001 Control record edits, 262, 267, 299, 305, 319, 345, 740, 741, 747
VEDA0001 Incoming SD orders, 442, 472

Modifications projects
ZEDITPXR Control record edits, 741, 742, 746, 747

Multi-pack, Sales BOM explosion, 552

Multiple message types to one basic type, 224

O

Order-to-cash cycle, 75, 76, 78, 153
Interfaces, 110, 111

Outbound processing requirements
ORDRSP PO confirm, 274, 300

Output determination structure
KOMKBEA Purchase header, 375, 378

Output determination structure (Cont.)
KOMKBV1 Header application V1, 286, 287, 288, 290, 302
KOMKBV2 Header application shipping, 621
KOMKBV3 Header application billing, 653, 654, 658

Output types, 277

Outsourced production, 70

P

Partner number conversion
External to SAP, 253–257

Partner profile, 86

Partner profile configuration
Inbound customer payment, 734, 735
Inbound customer purchase order, 478, 479
Inbound goods receipt/goods issue, 410, 411
Inbound PO confirmation, 392
Inbound ship confirmation, 601, 602
Inbound supplier invoice, 436, 437
Outbound ASN, 634–636
Outbound customer invoice, 698–700
Outbound PO confirmation, 539–543
Outbound ship order, 571–573
Outbound supplier PO, 390, 391

Partner profile custom load program logic, 754–762

Partner profiles, Inbound
Key values, 247
Parameters, 249–252, 340, 341

Partner profiles, Message control
Parameters, 294, 295, 300

Partner profiles, Outbound
Parameters, 294, 300

Partner profiles, Transporting, 248

Partner types, Standard, 245

Picking, 579

Plan Q from Outer Space, 40–42, 80, 111

Point of sales (POS), 37

Points of failure
Inbound technical, 844, 846
Outbound technical, 846, 847

POS (point of sales), 60, 95

Post Goods Issue (PGI), 576–584, 591, 838

Posting inbound goods receipt/goods issue, 402–406
 Posting inbound payment advice, 713–716
 Posting inbound sales orders, 449–455
 Posting inbound ship confirmation, 585–588
 Posting inbound supplier invoice, 426, 427, 428, 429
 Procedures, 283
 Process code, Link to processing function, 250

R

Reconciliation process, X12 820 payment advice, 712, 730, 731
 Recovery, Data issues
 Inbound, 881, 882, 883, 884, 885
 Outbound, 885, 886, 887, 888
 Recovery, Technical
 Inbound, 874, 875, 877
 Outbound, 878, 879, 880, 881
 Resource Integration Manager, 80
 RFC (remote function call), 47, 83, 156, 194, 300, 498
 RFC registered program, 157, 158

RIM

Business process workflow, 83
EDI process flow inbound, 191–193
EDI process flow outbound, 193–195
Inbound services, 164–169
Message routing, 162–165
Outbound services, 155–164
Overview, 80–87
Reporting status to SAP, 171–189
Role in Acme architecture, 155–171
SAP IDoc adapter, 83, 156, 158, 168
System design philosophy, 190, 191
Trading partner management, 151–154
Trading partner management, 151

S

Sales documents
 Document flow, 562
 Index tables, 453

SAP data feeds, 82
 SAP EDI system testing
 Integration, 822–835
 Interface, 815–822
 Strategy, 798–805
 Stress, 835–839
 String, 808–814
 Unit, 805–808
 SAP JCo classes, 169
 SAP JCo functionality, 158
 SAP JCo (Java Connector), 45, 83, 156, 169, 812, 814, 876, 877, 880
 SAP JCo system files, 157, 158
 SAP job scheduler, 47
 SAP NetWeaver PI, 43
 SDQ array, 468–471
 SDQ mapping, 457–471
 startRFC call parameters, 797
 STATUS IDoc interface, 182–186, 220
 STATUS IDoc interface, estimated data volumes, 196
 STATUS IDoc structure, 175, 177
 STATUS IDoc translation map, 175–180
 STATUS IDoc updates, 178

T

Tables, ABAP utilities
 TADIR Directory of repository objects, 344
 TFDIR Function modules, 381, 563
 TOJTB Business object repository, 322
 Tables, Business application
 AVIK Payment advice header, 109, 710–716
 AVIP Payment line item details, 109, 710–716
 AVIR Payment subitem details, 109, 711
 BKPF Accounting document header, 419, 620
 BSEG Accounting document item details, 419, 421, 620
 EKBE Purchase order history, 405
 EKET PO delivery schedule lines, 373
 EKKO Purchase order header, 372, 373, 379
 EKPO Purchase order item, 373

Tables, Business application (Cont.)

LIKP Delivery header, 554, 582, 583, 611, 621, 622, 647

LIPS Delivery item details, 555, 556, 582, 583, 612, 613, 622

MKPF Material documents header, 401

MSEG Material documents item details, 401

RBKP Invoice receipt header, 419, 420, 429

RSEG Invoice receipt item details, 419, 420, 429

T053E Reason code conversion, 729–733

T053R Reason codes, 733

T053S Reason code texts, 733

T076B Invoice company code, 731

T158G BAPI goods movement codes, 104

VAKPA Sales doc index, 474

VAKPA Sales orders by sold-to partner, 454

VAPMA Sales orders by material number, 454

VBAK Sales order header, 272, 304, 446–453, 474, 486, 487, 514, 622, 647

VBAP Sales order item, 260, 261, 272, 304, 446, 447, 453, 474, 475, 486, 487, 514, 622, 648

VBEP Delivery schedule lines, 304

VBFA Sales document flow, 550, 579, 581, 582, 583, 584, 620, 622

VBKD Sales doc business data, 304, 486, 556, 583, 612, 622

VBPA Sales doc partner data, 304, 446, 447, 453, 474, 476, 486, 555, 582, 612, 622, 623, 647

VBRK Billing header, 646, 659, 669

VBRP Billing item details, 647, 669

VBUK Sales doc status header, 582, 583, 622, 658

VBUP Sales doc status item, 582

VEKP Handling units header, 612, 622

VEPO Handling units items, 613, 622

VKDFS Billing index, 657

VLKPA Deliveries by ship-to partner, 454

VLPMA Deliveries by material number, 454

VRKPA Billing docs by payer partner, 454

VRPMA Billing docs by material number, 454

Tables, Custom development

ZEDINVRPT EDI inventory report, 326, 327, 328, 336

ZEDIXREF EDI partner mapping, 253, 255, 262–265, 267, 274, 296–308, 383, 388, 389, 399, 407–410, 418, 431, 436, 445, 456, 477, 478, 500, 538, 539, 565, 570, 589, 600, 626, 632, 634, 664, 697, 718, 733, 734, 741, 743, 744, 746, 748

Tables, Data dictionary objects

DD01L Domains, 232, 310

DD01T Domain descriptions, 232, 310

DD02L Tables, 233, 310

DD02T Table descriptions, 233, 310

DD03L Table fields, 85, 233

DD03T Table field descriptions, 85, 233

DD04L Data elements, 85, 232, 310

DD04T Data element descriptions, 85, 232, 310

DD07L Domain values, 85, 232

DD07T Domain value descriptions, 85, 232

Tables, EDI reference data

EDPAR SAP-external partner mapping, 253–265, 272, 274, 296, 297, 298, 308, 445, 476, 538, 645, 660, 661, 697

EDSDC Sales organization by vendor, 253–258, 265, 273, 445

PUMA SAP-external partner mapping, 610, 623, 626, 630, 632, 634

Tables, Enhancements

EDIFCT Link function to IDoc, 322

MODACT Modification projects, 320

MODSAP Enhancements, 320, 347

MODSAPT Enhancement descriptions, 320

MODTEXT Modification project descriptions, 320

Tables, IDoc configuration

EDE1T Outbound process code description, 324

EDE2T Inbound process codes, 323

EDIPOD File port definitions, 238, 241

EDIPORT Port types for IDoc processing, 266

EDIPOX XML file port definitions, 238, 241, 307

Tables, IDoc configuration (Cont.)

EDP12 Partner profile outbound message control, 278, 304, 316, 355, 359, 750, 751, 752, 753, 754, 756, 763
EDP13 Partner profile outbound, 278, 293, 315, 355, 750, 751, 753, 754, 756, 763
EDP21 Partner profile inbound, 247, 250, 270, 316, 750–756, 763
EDPAR SAP-external partner mapping, 48
EDPP1 Partner profile header, 247, 315, 750, 756, 758
RFCDES RFC destinations, 238
T000 SAP Clients, 237
TBDLS Logical system, 237
TBDLST Logical system texts, 237
TEDE1 Function link to outbound process code, 324
TEDE2 Function link to inbound process code, 323
TMSG1 Message link to outbound process code, 324
TMSG2 Message link to inbound process code, 323
TPAR Business partners, 253, 255

Tables, IDoc database

EDID4 Data record, 217, 219, 225, 514, 529, 669, 852, 861
EDIDC Control record, 217, 218, 225, 306, 514, 519, 669, 671, 674, 675, 852, 861, 867
EDIDD Data record structure, 270, 659, 671, 675, 677
EDIDS Status record, 217, 220, 225, 514, 669, 674, 675, 852, 862

Tables, IDoc processing

EDFI2 Last IDoc read from file, 266, 268, 873
EDIQO IDoc outbound RFC queue, 307
EDK21 Control record key structure, 267
T158G Goods movement codes, 404, 405, 408
TBD51 IDoc function attributes, 271
TBD52 Inbound IDoc function modules, 271

Tables, IDoc structure

CIMSYN IDoc extensions syntax description, 85

Tables, IDoc structure (Cont.)

EDBAS IDoc basic types, 234
EDCIM IDoc extended types, 234
EDIMSG IDoc message basic type link, 85
EDIMSGT IDoc message basic type link text, 234
EDISDEF Segment definitions, 233
EDISEGMENT IDoc segments, 85, 233
EDMSG Logical message type, 234
EDSAPPL Segment application structure, 233
IDOCSYN IDoc syntax, 234
IDOCSYN IDoc syntax description, 84, 224, 313

Tables, Master data

KNA1 Customer main, 89, 753
KNB1 Customer company code, 89
KNKA Customer credit management, 89
KNMT CMIR mapping table, 92, 253, 259, 260, 261, 274, 445, 454
KNMTK CMIR header, 92, 259
KNVP Customer partner functions, 48, 89
KNVV Customer sales area, 89, 298
KONKH Pricing conditions header, 94
KONKP Pricing conditions details, 94
LFA1 Vendor main, 90, 612, 623, 753
LFB1 Vendor company code, 90
LFM1 Vendor purchasing organization, 90
MARA Materials main, 91, 97
MARC Materials plant data, 91
MARD Materials storage location, 92
MARM Materials units of measure, 91
MAST BOM to material link, 93
MBEW Materials valuation area, 91
MVKE Materials sales data, 91
STAS BOM header to item link, 93
STKO BOM header, 93
STPO BOM item level raw material details, 93
T001 Company codes, 732

Tables, Message control

B000 Output control template, 288
B001 Output by sales org/customer, 288, 290, 302, 303, 616, 620, 621
B006 Output for billing, 653, 658, 659
B021 Delivery type/shipping pt, 558, 562

- Tables, Message control (Cont.)
- B025 Purchasing output*, 375, 377, 379
 - B150 Output by doc type/sales org/customer*, 491, 494, 497, 509
 - NACH Detailed output data*, 289, 292, 621, 659
 - NAST IDoc output status*, 303, 304, 379, 380, 381, 496, 498, 562, 563, 620, 622, 623, 658, 659, 792
 - T681 Conditions structures*, 379
 - T681Z Conditions dependent data*, 379
 - T682I Conditions access sequence*, 302
 - TEDE1 EDI process types outbound*, 304, 381
 - TNAPR Output processing programs*, 303, 380, 563
- Technical specs
- ASN*, 626, 627
 - Customer invoice*, 663–694
 - Customer PO*, 455, 456
 - Goods receipt/goods issue*, 406, 407
 - Payment advice*, 717, 718, 719, 720, 721, 722, 723, 724
 - PO confirmation*, 499–533
 - Ship confirm*, 588, 589
 - Ship order*, 565, 566
 - Supplier PO*, 382, 383, 384
- Title, definition, 53, 65
- Transaction codes, ABAP Data Dictionary
- SE11 ABAP Dictionary*, 215, 219, 222, 232, 233, 310, 325, 327, 667, 742
 - SE16 Data browser*, 310, 344, 347, 377, 494
- Transaction codes, ABAP utilities
- CMOD Enhancement projects*, 320, 341, 342, 350, 472, 747
 - SARA Archive and delete*, 199, 203
 - SBWP SAP Business Workplace*, 205
 - SCAT CATT test tool*, 836
 - SE37 Function builder*, 84
 - SE37 Function editor*, 266, 313, 315, 316, 317, 318, 319, 325, 343, 345, 347, 743
 - SE38 ABAP Editor*, 513, 756, 767
 - SE41 Menu painter*, 684, 686
 - SE51 Screen painter*, 680
 - SE80 Repository Info Center*, 266, 380, 472, 513, 532, 668, 680, 684, 742, 743, 756, 767
- Transaction codes, ABAP utilities (Cont.)
- SE93 Maintain transaction code*, 532, 724, 762, 771
 - SM30 Maintain tables*, 741–744
 - SM36 SAP Job scheduler*, 47, 383, 431, 500, 565, 586, 715, 718
 - SM37 View scheduled jobs*, 202, 270, 589
 - SMOD Enhancements*, 267
 - SU01 User maintenance*, 236
 - WE60 IDoc documentation*, 83
- Transaction codes, Business application
- F-28 Clearing*, 109, 706, 707, 716, 735
 - FBE1 Create payment advice*, 109, 705, 706, 707, 709, 716
 - FBE3 Display payment advice*, 711
 - MB1A Goods issue*, 397, 398, 400, 404, 405, 408
 - MB1B Transfer posting*, 404
 - MB1C Goods receipt other*, 404
 - MB01 Goods receipt for PO*, 103, 397, 400, 404, 405, 408, 427
 - MB03 Display material document*, 397, 400, 832
 - MB04 Subsequent adjustment*, 404
 - MB11 Goods movement*, 404
 - MB31 Goods receipt for order*, 404
 - ME21N Create purchase order*, 103, 110, 370, 378, 380
 - ME28 Release purchase orders*, 370
 - ME29N Release individual PO*, 370, 372, 378
 - ME51N Create purchase requisition*, 370
 - ME54N Release purchase requisition*, 370
 - ME58 Release purchase orders*, 370
 - ME59N Release individual PO*, 370
 - MIGO Goods movements*, 405
 - MIR4 Display invoice*, 421
 - MIRO Post incoming invoice*, 415, 416, 419, 425
 - VA01 Create sales order*, 257, 273, 302, 442, 446, 455, 482, 486, 496, 838
 - VA02 Change sales order*, 551
 - VF01 Create billing*, 641, 658, 813
 - VF02 Change billing*, 813
 - VF04 Billing due list*, 106, 641, 656, 834, 838
 - VF31 Output from billing*, 645, 813, 887

- Transaction codes, Business application (Cont.)
- VL01 Create outbound delivery legacy*, 562
 - VL01N Create outbound delivery*, 549, 550, 551
 - VL02N Change outbound delivery*, 106, 577, 581, 607, 838
 - VL03N Display outbound delivery*, 556, 614
 - VL10 Delivery due list*, 105, 549, 550, 551, 561, 562, 838
- Transaction codes, Custom
- ZEDIPP Load partner profiles*, 762
 - ZEDIXFR IDoc ALE transfer program*, 764, 771
 - ZSPLIT Payment split program*, 711, 716, 724
- Transaction codes, Custom programs
- ZEDINV INVOIC IDoc edit program*, 641, 648, 667, 693
 - ZSDQ Build SDQ ORDRSP*, 488, 532, 533
- Transaction codes, IDoc configuration
- BD54 Logical system*, 236
 - OBCA Company code mapping*, 731, 732
 - SALE ALE IMG*, 235, 773
 - SM59 RFC destination*, 194, 238, 880
 - VNPU PUMA entries*, 633, 634
 - VOE2 ESDC entries*, 259, 477, 883
 - VOE4 EDPAR entries*, 48, 256, 476, 477, 538, 697
 - WE20 Partner profile*, 88, 90, 248, 249, 294, 315, 325, 340, 343, 361, 390, 410, 436, 478, 539, 571, 601, 615, 634, 651, 698, 734, 749, 762, 775
 - WE21 Maintain ports*, 239
 - WE44 Partner types*, 245
 - WE47 Maintain status codes*, 197, 222
 - WEDI EDI area menu*, 309, 312, 313, 321, 322, 323, 360, 435, 494, 501, 503, 506, 507
- Transaction codes, IDoc development
- BD51 IDoc attributes*, 322, 325, 337
 - SMOD Enhancements*, 319, 320, 342, 343, 344
- Transaction codes, IDoc development (Cont.)
- WE30 IDoc basic type editor*, 234, 312, 313, 325, 329, 342, 348, 503
 - WE31 IDoc segment editor*, 101, 233, 311, 312, 325, 328, 342, 348, 501
 - WE32 Maintain view*, 507
 - WE41 Outbound process codes*, 323–325, 360, 494
 - WE42 Inbound process codes*, 322, 323, 325, 338, 435
 - WE57 Link function to IDoc*, 321, 322, 325
 - WE81 Maintain logical message types*, 234, 313, 325, 331
 - WE82 Link message to basic type*, 313, 325, 331, 342, 350, 506
- Transaction codes, IDoc test tools
- WE12 Turnaround utility*, 795
 - WE15 Output from message control*, 887
 - WE19 Main IDoc test tool*, 223
- Transaction codes, IDoc utilities
- BD87 Status monitor for messages*, 374, 401, 447, 448, 489, 557, 583, 627, 650, 693, 712, 796, 814
 - BDBG Generate IDoc for BAPI*, 101
 - BDM8 Send ALEAUD IDoc confirmation*, 772–777
 - SWWH Delete work item history*, 204, 206
 - SWWL Mass delete work items*, 204, 205, 206, 207
 - WE05 IDoc list*, 374, 401, 447, 448, 489, 557, 583, 650, 664, 665, 693, 712, 796
 - WE09 IDoc search by field contents*, 204, 627, 665
 - WE10 IDoc search by field contents*, 204
 - WE14 Output batched IDocs*, 306, 832
 - WE60 IDoc basic type documentation*, 212, 229, 779
- Transaction codes, Master data
- CS01 Create BOM*, 93
 - MM01 Create material*, 91
 - VD51 Create customer material info record*, 92, 261, 850
 - VD52 Change customer material info record*, 92, 261

Transaction codes, Master data (Cont.)

- VD53 Display customer material info record*, 261
- XD01 Create customer*, 89
- XD02 Change customer*, 90
- XK01 Create vendor*, 90

Transaction codes, Message control

- MN04 Condition records for PO output*, 376
- NACE Conditions for output control*, 279–290, 375, 376, 490, 492, 493, 558, 559, 560, 616, 617, 618, 650, 652, 653
- V/27 Maintain requirements*, 286
- VV11 Condition records for sales*, 290, 493
- VV21 Condition records for shipping*, 560, 562, 618, 621
- VV23 Condition records shipping*, 614
- VV31 Condition records for billing*, 654
- VV33 Display billing condition records*, 651

Transaction codes, System

- SPRO IMG configuration*, 286, 421, 732

Trouble shooting and recovery

- Defining failure*, 844, 845, 846, 848, 849, 850
- Defining success*, 842, 843, 844
- Functional and business issues*, 881–888
- Technical issues*, 874–881

U

UN/CEFACT, 121

United States Census Bureau EDI statistics, 116

UPC (Universal product code), 61, 65, 91, 259

User exits, 98

V

VMI, 42, 55–63

Data feeds, 58, 68

VMI as relationship of trust, 154, 155

VMI data

- Inventory*, 58, 153
- POS (point of sale)*, 101, 153

VMI data (Cont.)

- POS (point of sale) data*, 58
 - Reserved PO numbers*, 58, 61, 153
- VMI orders, 58, 95, 96, 98, 99, 444, 446, 450, 451, 452

W

Workflow, 43

X

X12

Anatomy, 128–145

X12 820 split large payment, 714–724

X12 820 structure issues in split, 720–724

X12 850 PO, deconstructed, 134–145

X12 856, HL hierarchy, 594, 595, 596, 597, 599

X12 997-STATUS interface, 173, 186, 187, 188, 189, 195

X12 envelopes

GS functional group, 129, 133, 146, 147

ISA interchange, 132, 146, 148

ST transaction set, 129, 133, 134, 146, 147

X12 grammar, 130, 131, 132

X12 industry subcommittees, 119

X12 ST envelope, Link to translation map, 159, 194

X12 transaction sets

810 Customer invoice, 78, 107, 111, 147, 181–194, 640–650, 703, 717, 809, 813, 815, 816, 819, 824, 835, 843, 849, 885, 887

810 Supplier invoice, 75, 104, 110, 415–436

812 Debit/credit memo, 78, 704–708, 716–721

816 Organizational relationships, 104

820 Payment advice, 47, 78, 108, 111, 703–717, 725, 731, 735, 835

824 Error report, 106, 607, 611, 614, 625, 627, 637, 639, 642, 655, 849, 850

830 Ship order, 105, 111, 548–567, 586, 602, 605, 610, 816, 823, 829

X12 transaction sets (Cont.)

- 832 *Price/sales catalog*, 103
- 846 *Inventory report*, 309, 324, 325, 326
- 850 *Customer purchase order*, 71, 98, 128, 146, 164, 166, 168, 192, 254–264, 426, 441, 445, 448–469, 483, 484, 489, 519, 552, 809, 811, 818, 828, 842, 843, 848, 881
- 850 *Supplier purchase order*, 73, 103, 110, 150, 309, 362, 369–374, 382, 383, 388, 389, 391, 887
- 852 *Point of sales/on-hand inventory*, 61, 63, 75, 96, 101, 110, 153, 155, 441–457, 477
- 855 *PO confirmation*, 73, 77, 103, 111, 155, 159, 161, 296, 369, 370, 371, 374, 382–389, 426, 457, 481, 483, 485, 489, 499, 500, 509, 510, 534–538, 553, 561
- 856 *Advanced ship notification*, 77, 78, 106, 111, 607–627, 637, 642, 643, 655, 816, 824, 832, 849
- 856 *Shipping confirmation*, 106, 111, 576–599, 832
- 864 *Error report*, 106, 108, 641, 643, 646, 663, 664, 849, 850, 887
- 864 *Text report*, 650, 703
- 864 *Text report as error report*, 78

X12 transaction sets (Cont.)

- 867 *Goods receipt/inventory adjustment*, 74, 397–410, 427
 - 888 *Item maintenance*, 103
 - 940 *Warehouse ship order*, 77
 - 944 *Warehouse stock transfer*, 74, 103, 110
 - 997 *Functional acknowledgment*, 72, 73, 78, 87, 107, 153, 161–195, 369, 382, 383, 415, 430, 431, 443, 455, 456, 483, 499, 500, 549, 553, 561, 564, 566, 577, 586, 588, 589, 607, 624–626, 639, 641, 650, 655, 663, 664, 703, 705, 714, 717, 817, 821, 829, 832, 842, 843, 844
- XML IDocs, 226, 227, 228, 229, 230
- Advantages*, 212
- XML runtime processing data, 83
- XML schema, 160
- XML Schema Definition (XSD), 228
- XPath, 42, 43, 160, 170
- XSD schema
- Adding enumerators*, 781, 782, 783, 784
 - As XML metadata dictionary*, 782
 - Enumerators*, 778
 - Getting from SAP*, 779, 780
 - IDoc*, 83, 228, 229, 778
 - Using enumerators in IDoc*, 784, 785