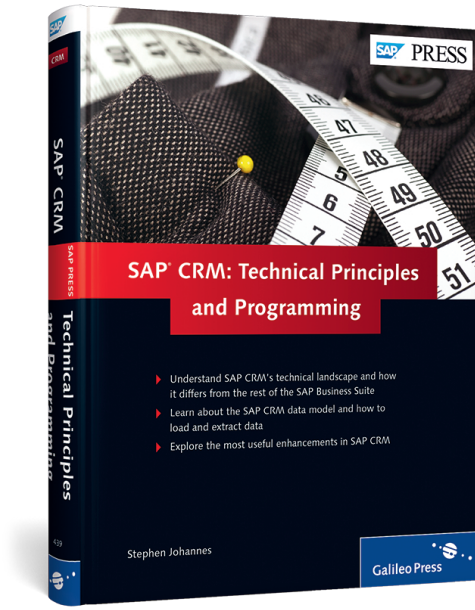
Stephen Johannes

SAP® CRM: Technical Principles and Programming



Galileo Press

Bonn • Boston

Contents at a Glance

1	Understanding the Basic Architecture of SAP CRM	25
2	The SAP CRM Data Model	53
3	Data Model Extension Techniques	105
4	Business Transaction Event Framework	155
5	Data Extraction and Loading with the XIF Adapter	201
6	The Post Processing Framework: Output and Actions	245
7	Common Enhancement Requests in Sales and Service	293
8	Common Enhancement Requests in Marketing	349
9	Common Enhancements in Analytics and Reporting	379
10	When All Else Fails	407
A	Common Mistakes When Setting Up an SAP CRM Development System	425

Contents

Acknowledgments 13
Introduction 17

1 Understanding the Basic Architecture of SAP CRM 25

1.1 Defining Customer Relationship Management 25
1.1.1 Sales 28
1.1.2 Service 31
1.1.3 Marketing 32
1.1.4 E-Commerce 34
1.1.5 Interaction Center 34
1.2 Reasons for a Separate SAP CRM System 35
1.2.1 Performance 35
1.2.2 Development Cycles 36
1.2.3 Mobility 36
1.2.4 Non-SAP Systems 37
1.2.5 Target User Base 38
1.3 Technical Landscape of SAP CRM 38
1.3.1 System Components 39
1.3.2 User Interface 41
1.3.3 Recommended Landscape 44
1.3.4 SAP CRM Middleware 45
1.3.5 Data Model 47
1.3.6 Development Tools 47
1.3.7 Reporting 49
1.3.8 Supporting SAP CRM 50
1.4 Summary 51

2 The SAP CRM Data Model 53

2.1 Data Model Background 53
2.1.1 SAP ERP Data Model for Sales 54
2.1.2 Segments and Tables 54
2.1.3 Major Objects of the SAP CRM Data Model 55
2.2 Business Partners 55
2.2.1 Business Definition 55

2.2.2 Technical Definition 56
2.3 Products 69
2.3.1 Business Definition 70
2.3.2 Technical Definition 70
2.4 One Order: SAP Business Transaction 77
2.4.1 General Design 78
2.4.2 Programming with One Order 85
2.5 Marketing Attributes 94
2.5.1 Business Definition 94
2.5.2 Technical Definition 95
2.5.3 API Function Modules 102
2.6 Summary 103

3 Data Model Extension Techniques 105

3.1 Easy Enhancement Workbench 105
3.1.1 Add a New Field 106
3.1.2 Create an Extension 109
3.1.3 Specify New Fields to Be Added 112
3.2 Application Enhancement Tool 118
3.2.1 Launching the AET 119
3.2.2 Adding a New Field 119
3.2.3 Adding a New Table 127
3.2.4 Conclusion 137
3.3 Manual Enhancements: Don't Try This at Home 137
3.4 Marketing Attributes 138
3.4.1 Creating New Marketing Attribute Sets 138
3.4.2 Create a New Attribute or Choose an Existing Attribute 139
3.5 Product Master Attribute Sets 144
3.5.1 Create a New Attribute for a Product 144
3.5.2 Create a New Set Type 147
3.5.3 Assign the Set Type to a Product Category 150
3.6 Summary 154

4 Business Transaction Event Framework 155

4.1 Introduction to the Business Transaction Event Framework 156

4.2 The Three Main Components of the Business Transaction Event Framework 157
4.2.1 Customizing Data 157
4.2.2 Framework Code 171
4.2.3 Event Handlers 174
4.3 Standard Delivered Events 176
4.3.1 Review Business Transaction Event Framework Customizing 177
4.3.2 Review Logic 177
4.3.3 Business Object Categories for Transactions 178
4.3.4 Code Field Updates 179
4.3.5 List SAP-Delivered Event Handlers 180
4.4 Event Trace 181
4.4.1 Prerequisites 182
4.4.2 Review the Trace 182
4.5 Creating an Event Module 185
4.5.1 Determining the Correct Event 186
4.5.2 Creating the Function Group 187
4.5.3 Building the Function Module 190
4.5.4 Common Code Structure 194
4.5.5 Registering the Function Module 196
4.6 Summary 200

5 Data Extraction and Loading with the XIF Adapter 201

5.1 Introduction to the XIF Adapter 201
5.1.1 Design Benefits 202
5.1.2 Structure of an XIF Remote-Enabled Function Module (RFC) 202
5.1.3 Finding an XIF Module 203
5.2 Loading Data via the XIF Adapter and the Legacy System Migration Workbench 204
5.2.1 Identifying the Target XIF Adapter 205
5.2.2 Creating a New LSMW Data Conversion Project 205
5.2.3 Setting Up the LSMW for Inbound IDoc Processing 207
5.2.4 Mapping the Flat File in LSMW 214
5.2.5 Running the Conversion Programs 223

5.2.6 Troubleshooting the Conversion Program 223
5.3 Loading Data via XIF and Custom Code 225
5.3.1 Required Data Segments 226
5.3.2 Putting It All Together 229
5.3.3 Example: Creating a Product 233
5.3.4 Example: Creating a Business Partner 236
5.4 Extracting Data via the XIF Adapter 238
5.4.1 Create a Logical System 239
5.4.2 Create Receiver Port 240
5.4.3 Create Partner Profile 241
5.4.4 Create an XIF Adapter Site 242
5.5 Summary 243

6 The Post Processing Framework: Output and Actions 245

6.1 Introduction to the Post Processing Framework (PPF) 245
6.2 Customizing 247
6.2.1 Standard Delivered Action Profiles 247
6.2.2 Actions 249
6.2.3 Condition Configuration for Actions 259
6.2.4 Creating a New Action Profile 267
6.2.5 Assigning the Profile to a Business Transaction ... 272
6.3 Action Scheduling 272
6.4 Using Actions for Nonoutput Tasks 278
6.4.1 Standard Delivered Method Calls 279
6.4.2 Creating a New Method Call 279
6.4.3 Implementing the IF_EX_EXEC_METHODCALL_PFF Interface 283
6.4.4 Method Action Customizing 285
6.5 Summary 290

7 Common Enhancement Requests in Sales and Service 293

7.1 Requiring Fields in a Business Transaction 294
7.1.1 Business Scenario 295
7.1.2 Incompletion Procedure 295
7.1.3 Using Segment BAdIs 301
7.1.4 ORDER_SAVE BAdI 307

7.2 Defaulting Values in a Business Transaction 313
7.2.1 Business Scenario 313
7.2.2 Segment BAdI 313
7.2.3 Business Transaction Events 315
7.3 Custom Date Rules 317
7.3.1 Business Scenario 317
7.3.2 Create New Date Rule 318
7.3.3 Store the Result 322
7.4 Partner Determination Access Rules 327
7.4.1 Business Scenario 328
7.4.2 Configuration Only Method 328
7.4.3 Configuration and BAdI Implementation Method 338
7.5 Organization Model Access Rules 340
7.5.1 Standard Delivered Rules 341
7.5.2 Creating a Custom Rule 342
7.6 Summary 347

8 Common Enhancement Requests in Marketing 349

8.1 External List Management BAdI 350
8.1.1 BAdI Overview 350
8.1.2 Method CREATE_BUSINESS_TRANSACTIONS 351
8.1.3 Method CREATE_BUSINESS_PARTNERS 364
8.2 Open Channel BAdI for Campaign Execution 365
8.2.1 Campaign Customizing for Open Channel Use 366
8.2.2 Implementing the Open Channel BAdI 368
8.2.3 Testing the BAdI 373
8.3 Summary 377

9 Common Enhancements in Analytics and Reporting ... 379

9.1 SAP NetWeaver BW Data Source Enhancements via BAdIs 380
9.1.1 Enhance the Data Source Structure 381
9.1.2 Extractor Program Logic 385
9.2 Interactive Reporting Enhancements 393

9.3 Displaying Custom Reports Using the Transaction Launcher 396
9.3.1 Setting Up the SAP GUI for HTML 397
9.3.2 Defining the URL for the SAP Web Client 397
9.3.3 Configuring the Transaction Launcher 398
9.4 Summary 406

10 When All Else Fails 407

10.1 Implicit Enhancements 407
10.1.1 Create an Implicit Enhancement 408
10.1.2 Common Uses of Implicit Enhancements 409
10.1.3 Upgrades 413
10.2 Core Modifications 414
10.3 Community Resources 416
10.3.1 SAP Community Network 416
10.3.2 SAP User Groups 421
10.3.3 Social Media 422
10.4 Summary 423

Appendices 425

A Common Mistakes When Setting Up an SAP CRM Development System 425
B The Author 429

Index 431

Introduction

SAP Customer Relationship Management (which we'll refer to throughout the book as SAP CRM) is a software solution delivered by SAP to aid businesses in their management of relationships between the company, their current customers, and their potential customers. A successful SAP CRM implementation provides tools that allow the business to manage relationships and the experience with the customer. This book will give you the technical foundation you need to be more effective on an SAP CRM project as a developer. By reading this book, you should gain the knowledge needed to implement enhancements that will allow you to meet many different business requirements and increase the success of your SAP CRM implementation.

The information gathered is based on multiple SAP CRM projects and years of experience supporting SAP CRM productively. We'll reveal information that was learned only by working on SAP CRM projects, or by searching and reading hundreds of articles on the Internet.

Who This Book Is For

This book is aimed at developers and consultants who are currently working with SAP CRM or who are new to SAP CRM. We recommend that you have a general working knowledge of ABAP, ABAP Objects, and general programing. You should be able to read and write ABAP code without difficulty in order to understand the examples in this book. Some basic experience in SAP configuration may be helpful in setting up parts of the system required for the examples; however, we do cover the basic configuration required.

What This Book Covers

This book provides an introduction to customer relationship management in general and to SAP CRM specifically. We introduce the business

processes supported by SAP CRM and cover the basic overall technical architecture.

Key concepts

Throughout the book, we examine key concepts such as the data model of SAP CRM. This knowledge will serve as the foundation in how to enhance SAP CRM to meet key business requirements. As we go through the book, we cover various tools provided by SAP CRM in the system that can be used to meet various requirements. In addition to examining the tools available, we provide several examples of business requirements and how to solve those business requirements using the available tools. As every project needs a fallback plan, we also explore options that should be considered only as a last resort, but kept on hand if necessary.

The book starts with an overview of the SAP CRM system. It then dives into the SAP CRM data model and explores how to enhance that data model. The next three chapters focus on three major technical areas of the business transaction: business transaction event framework (BTE framework), XIF adapter, and Post Processing Framework (PPF). These technical areas cover custom logic, data loads and extracts, and finally output for the business transaction. We then look at common enhancement requests in the Sales, Service, and Marketing modules of SAP CRM found in most SAP CRM projects. We then move to cover enhancements for analytics and reporting. We end the book by examining methods of last resort, when there are no other options left to meet your requirements.

Chapter breakdown

The chapters are organized as follows:

- **Chapter 1**
  This chapter provides an introduction to SAP CRM. We explain what SAP CRM is and why we have a separate SAP CRM system, instead of SAP CRM being a component in SAP ERP. We also take a look at the basic technical landscape of SAP CRM.
- **Chapter 2**
  This chapter discusses the SAP CRM data model for business partners, products, business transactions, and marketing attributes. If you only read one chapter in full in this book, we recommend you read this

chapter. The information in this chapter will be foundational for all other chapters.

- **Chapter 3**
  This chapter explains how to enhance the data model of SAP CRM. We'll look at four different tools: EEWB, AET, marketing attributes, and product attributes for data model enhancements. Enhancing the data model is the number one requirement for most SAP CRM projects.
- **Chapter 4**
  This chapter discusses the business transaction event framework (BTE framework). The BTE framework is a function module-based framework that allows for custom logic to be added within your SAP CRM system for business transactions. We look at the customizing of this framework and provide an example of how to build on the BTE function module.
- **Chapter 5**
  This chapter discusses data extraction and loading using the XIF adapter. We examine how to load data and extract into SAP CRM using the XIF adapter via two main methods: LSMW and custom code. If you need to load legacy data into SAP CRM or extract data from SAP CRM, we encourage you to review this chaper.
- **Chapter 6**
  This chapter discusses the Post Processing Framework (PPF), which is the SAP CRM version of output control for business transactions. In addition, it can be used for custom business logic. We cover the customizing of this logic and how to create custom logic within this framework.
- **Chapter 7**
  This chapter discusses common enhancements in sales and service. We show you how to make fields be required and values be default in the business transaction. Through configuration and coding, we examine how date rules can be used for complex time calculations in a business transaction. Finally, we look at how to extend partner and organizational model determination.

- **Chapter 8**
  This chapter discusses common enhancements in marketing. Our focus will be on two BAdIs. The first BAdI deals with External List Management (ELM) used in the import of list data into SAP CRM. The second BAdI deals with exporting data to the open channel when executing a marketing campaign in SAP CRM.
- **Chapter 9**
  This chapter discusses common enhancements in analytics and reporting. We show you how to extend data sources for SAP NetWeaver Business Warehouse (SAP NetWeaver BW). A brief look at enhancing interactive reporting is also provided. Finally, we illustrate how the Transaction Launcher can be used to display custom reports.
- **Chapter 10**
  This chapter discusses other techniques that can be used when all else fails. We look at implicit enhancements, core modifications, and community resources.

Technical specifications

All coding samples and screenshots were prepared using the initial SAP CRM 7.0 and SAP CRM 7.0 EHP1 available at the time of the writing of this book (January 2013). Unless noted, coding examples aren't specific to either release. Based on past experience, examples may work on SAP CRM 7.0 with the exception of examples concerning the Application Enhancement Tool. Using a sandbox system to practice or follow the examples is strongly encouraged.

Example Scenario: Developing a Social Call Report

Throughout the book we'll discuss the concept of a social call report as a business example for the enhancements. The business requirements are scattered through each chapter and organized by the technical nature of each enhancement, so we'd like to elaborate on how all of these enhancements fit within a typical SAP CRM implementation and explain which areas of the book cover those requirements. Here, we're assuming that you're an SAP CRM functional consultant who needs to verify the development effort of your project.

Business Background and Need

The example company has recently entered the world of social media through a corporate account on Facebook and Twitter. Up until this point, the company's only interactions with their customers was through phone, email, and the corporate website. However, the company started to notice that customers were talking about them on these social channels. The company already had a great SAP CRM system implemented that was providing a good view of their customer interactions; however, they were missing the social interactions and honest feedback and suggestions about the direction of the company.

Their sales representatives were already logging their interactions with customers in SAP CRM using an activity business transaction called a call report. The company decided that they'd like to have their sales force interact with customers on social media directly and record those interactions manually in the SAP CRM system until they could upgrade/purchase a social media integration package. While not the optimal solution, the modifications would allow the company to start listening, responding, and capturing these conversations to prevent them from losing some of their customer base.

Fit Gap Analysis

The development team determined that the standard SAP CRM system did not contain any specific tools out of the book for recording social media interactions. In reviewing the system, we noticed several gaps, which included business partner master data, transaction data and processing, loading data into the system, and reporting.

Add fields

The first major gap we noticed was that we didn't see any specific fields on the business partner where we could specify the Facebook or Twitter account of a business partner. The information contained in Chapter 2 will allowed us to confirm our findings. We will need to add these fields to the business partner in order to be able to understand who the company's customers are on Twitter. To resolve this, we'll use the Application Enhancement Tool to add two new fields for Facebook ID and Twitter account ID for business partners. You can use the information in Section 3.2 of this book to understand how the AET can be used to add those fields to the business partner in your system.

The second major gap we noticed is that the standard activity transaction did not contain all the fields we need to capture. Also, when creating the transaction, we noticed that it did not perform the necessary checks and validations to properly record a social media interaction. Accordingly, we'll need to create a new business transaction type called a *social call report* that will store all of our social media interactions in the SAP CRM system. The call report will need to be able to record notes and the applicable hashtag of the social media interaction to provide the company with a rich view of the conversation. The standard SAP CRM business activity does not contain a field to record a hashtag, so we will need to add the field using the AET tool as explained in Section 3.2.

Required fields

For all interactions, we would like to require that the hashtag is filled out. Now we understand that not all tweets or Facebook interactions will use a hashtag, so we will have our sales representative put a one or two word description of the primary subject of the interaction if there is no hashtag provided in the tweet. It's important that this field is always maintained by the company's sales representative in order to have accurate analytics on the type of conversations that are occurring. We can use the information in Section 7.1 to achieve this requirement. However, we may want to default a subject, as initial feedback from the company's sales representatives indicates that trying to pick a single appropriate word is next to impossible. Section 7.2 provides an example of how you can default this value within the business transaction. Chapter 4 also gives other options you can use for inserting custom business logic rules into the social call report business transaction.

Automatic field population

When we create the transaction, we want the dates of when it occurred to be populated automatically to the current date, so the sales representatives have less work to do. The information in Section 7.3 will allow you to achieve this requirement. We also want the correct contact person to be suggested when the transaction is created. If the system could figure out or suggest other people that we know that might have similar tweets or influence the interaction, we want it to provide a list of suggestions so that we can record that into the social call report. The information in Section 7.4 will help you to meet that requirement.

Formatted report output

Even though the company communicates electronically, the sales representatives would still like a formatted output of the social call report, similar to how they have a print output of their standard call report. The company also wants to send email notifications to their marketing group, as these social call reports are being created so they can make sure that their sales force is properly aligned with the branding message. The Post Processing Framework can be used to achieve this requirement; refer to Chapter 6.

Historical data

Now that the social media interactions can be stored in the SAP CRM system, we would like to have some historical data available upon launch. It will be hard for the team to have a complete view of the customer without some historical information that shows what happened in the last two years. We've hired some great web programmers to download and create a few flat files that contain interactions relevant to the company from Facebook and Twitter. The sales representatives have even agreed to clean and review the data so that it will end up in a structured tab delimited file. To meet this requirement, we could examine either using the XIF adapter (Chapter 5) or look at extending External List management (Section 8.1). Either option could work; it's just a matter of determining whether the data should be loaded on a recurring basis, or this will be a one-time conversion. If this is a one-time conversion, our development team suggests XIF, while if this is a recurring process they think the marketing group should own this and they are willing to extend ELM so that going forward, the marketing group can be self sufficient.

Reporting and analysis

Senior management has requested that they track and analyze how many social interactions the company is receiving, and if any correlation to sales volume can be determined. The company currently has an SAP NetWeaver BW system in place that contains the sales volume data. We would like to send over information about the social media interactions to the SAP NetWeaver BW system to create combine reports on the number of social media interactions for the customer versus the ordering volume. (In Section 9.1, you'll learn how to extend SAP CRM to provide this information to the SAP NetWeaver BW system.) The company also wants to do some simple reporting on a weekly basis of all social media interactions at a detailed level received based on the hashtag

entered. They believe that with those metrics, they'll have a better understanding of their customer base. In Section 9.3, you'll learn how you can launch some simple custom reports that you could develop to meet this requirement.

The company wants to accomplish all these requirements using the existing SAP CRM support staff within six to eight weeks. They're hoping to handle this all in-house, as they don't have the budget to pay for external consulting resources. Chapter 10 provides some resources to access when you've run out of solutions to solve the business requirements.

Summary

Although this scenario is purely fictional, the discussion of the requirements and how to meet those requirements is close what you will find on actual SAP CRM project when talking with your technical team. We hope that by matching the tools we've presented with this business scenario, your team can better use this book to solve your SAP CRM implementation requirements. In addition, we've chosen the social media example as it is simple to explain, but has several required technical components and is hopefully more relevant to your business than a flight reservation system. If for some reason you do actually end up productively implementing this social media scenario in your SAP CRM system, please let us know via Twitter *@sapcrmtpp*.

2.1.3 Major Objects of the SAP CRM Data Model

Data objects

SAP has several major data objects that are key to the application: *business partners*, *products*, *business transactions*, and *marketing attributes*. Every SAP CRM module uses at least one of these four objects to support the business process represented by that module.

Now that you know about the design of the data model, we'll discuss each of these major objects in SAP CRM in detail.

2.2 Business Partners

The primary focus of CRM is on managing the relationship with the customer. In SAP CRM, the business partner is used to model the customer in the software solution. You first need to learn the business definition of the business partner and then understand how this is technically translated into a database model, which we'll explain in the following sections. Finally, we'll discuss some common techniques used to retrieve business partner information from the database model.

2.2.1 Business Definition

As we've mentioned, the customer is the heart of CRM, and so the business partner is the heart of SAP CRM. It's impossible to execute any scenario in SAP CRM without a business partner. A *business partner* is any person, company, or organization that is part of a relationship between a company and its customers. A business partner can be a customer, vendor, contact person, employee, third party, or other entity that needs to be tracked or is involved in a customer interaction. In traditional SAP ERP, customers, vendors, and contact persons are stored separately and act as different entities. In SAP CRM, all business partners are stored centrally and function based on the role provided.

Everything Is a Partner

One of the hardest concepts to understand for new SAP CRM users is that a business partner isn't just a customer but rather can represent any party involved in an interaction. One complaint or result from this is that if your

master data in your SAP ERP system isn't clean or is set up in a confusing manner, it will be the same in SAP CRM. Keep this in mind when you get requests about eliminating duplicate data from your SAP CRM system.

2.2.2 Technical Definition

SAP Business Partner concept

The SAP CRM business partner is based on the generic SAP Business Partner concept. In creating the SAP CRM business partner, new extensions to the SAP Business Partner were created that are only found in SAP CRM. This data is normally referred to as SAP CRM-specific data. The SAP Business Partner is used in other applications such as Financial Supply Chain Management (FSCM) within the SAP ERP system. You'll also find the SAP Business Partner used in SAP Supplier Relationship Management (SAP SRM) due to the fact that SAP CRM and SAP SRM originally shared a common technical foundation.

Three technical divisions of the data model

The SAP CRM overall data model features three technical divisions:

- General data is primarily part of the Business Partner data model and was traditionally modeled via the Business Data Toolset (BDT).
- Address data is based on the generic address management services provided by SAP.
- Relationship data allows connecting and/or relating business partners in the system together.

The four tables illustrated in Table 2.1 (and discussed in the following subsections) are the primary tables of the business partner in SAP CRM that connect the business partner to each technical division of data.

Table Name	Description
BUT000	General data
BUT020	Address data
BUT050	Relationships
BUT100	Business partner roles

Table 2.1 Primary Business Partner Data Tables

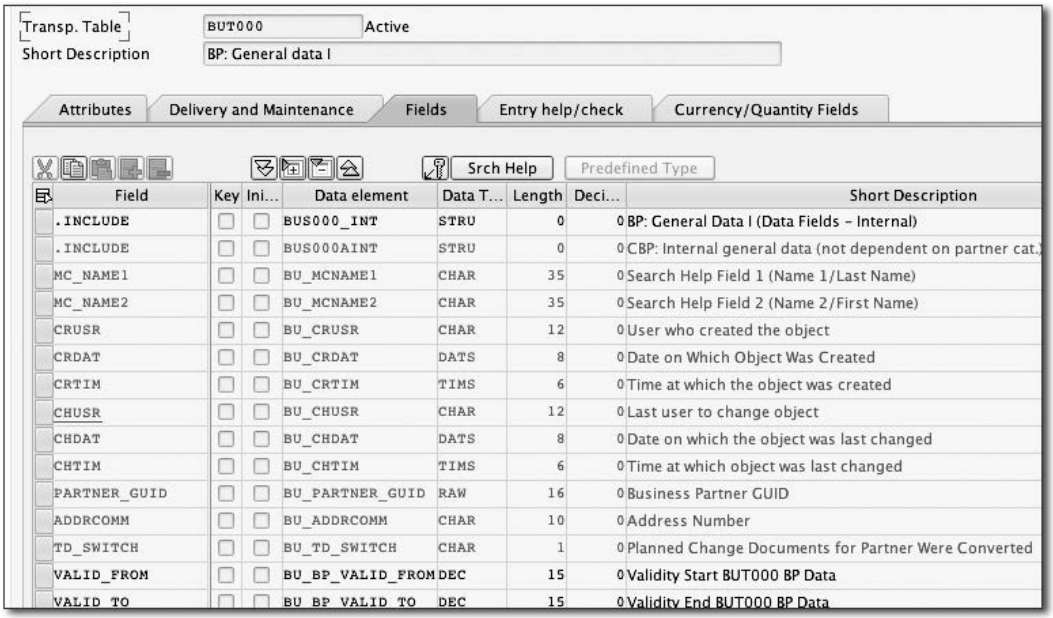
General Data

Because the business partner was designed to be an application service not unique to SAP CRM, the design is different in several aspects from other data objects. The SAP CRM business partner is the only major data object in SAP CRM not using a GUID as a primary key.

The business partner consists of several key tables, each starting with "BUT". **Key tables**

Table BUT000

Table BUT000 is considered the primary or header table for the business partner. The primary key is the PARTNER, which is a 10-digit alphanumeric field modeled by the data element BU_PARTNER. The table also has a secondary index based on the field PARTNER_GUID, which is used to link the standard business partner data to SAP CRM extension data and business transactions. The PARTNER_GUID field is found toward the end of the table as shown in Figure 2.1.



Field	Key	Ini...	Data element	Data T...	Length	Deci...	Short Description
.INCLUDE			BUS000_INT	STRU	0	0	BP: General Data I (Data Fields – Internal)
.INCLUDE			BUS000AINT	STRU	0	0	CBP: Internal general data (not dependent on partner cat.)
MC_NAME1			BU_MCNAME1	CHAR	35	0	Search Help Field 1 (Name 1/Last Name)
MC_NAME2			BU_MCNAME2	CHAR	35	0	Search Help Field 2 (Name 2/First Name)
CRUSR			BU_CRUSR	CHAR	12	0	User who created the object
CRDAT			BU_CRDAT	DATS	8	0	Date on Which Object Was Created
CRTIM			BU_CRTIM	TIMS	6	0	Time at which the object was created
CHUSR			BU_CHUSR	CHAR	12	0	Last user to change object
CHDAT			BU_CHDAT	DATS	8	0	Date on which the object was last changed
CHTIM			BU_CHTIM	TIMS	6	0	Time at which object was last changed
PARTNER_GUID			BU_PARTNER_GUID	RAW	16	0	Business Partner GUID
ADDRCOMM			BU_ADDRCOMM	CHAR	10	0	Address Number
TD_SWITCH			BU_TD_SWITCH	CHAR	1	0	Planned Change Documents for Partner Were Converted
VALID_FROM			BU_BP_VALID_FROM	DEC	15	0	Validity Start BUT000 BP Data
VALID_TO			BU_BP_VALID_TO	DEC	15	0	Validity End BUT000 BP Data

Figure 2.1 Partner GUID on Table BUT000

Other important fields include TYPE, BPKIND, and BU_GROUP, as shown in Figure 2.2.

Transp. Table BUT000 Active
Short Description BP: General data I

Attributes | Delivery and Maintenance | Fields | Entry help/check | Currency/Quantity Fields

Srch Help | Predefined Type

Field	Key	Ini...	Data element	Data T...	Length	Deci...	Short Description
CLIENT	☑	☑	MANDT	CLNT	3	0	Client
PARTNER	☑	☑	BU_PARTNER	CHAR	10	0	Business Partner ID
.INCLUDE	☐	☐	BUS000_DAT	STRU	0	0	BP: General Data I (Data Fields – External)
.INCLUDE	☐	☐	BUS000A	STRU	0	0	CBP: General data (independent of partner cat.)
.INCLUDE	☐	☐	BUS000AINI	STRU	0	0	CBP: General Data (not dep. on Part. Cat.), Initial Screen
TYPE	☐	☐	BU_TYPE	CHAR	1	0	Business partner category
BPKIND	☐	☐	BU_BPKIND	CHAR	4	0	Business partner type
BU_GROUP	☐	☐	BU_GROUP	CHAR	4	0	Business Partner Grouping
BPEXT	☐	☐	BU_BPEXT	CHAR	20	0	Business partner number in external system
.INCLUDE	☐	☐	BUS000ADAT	STRU	0	0	CBP: General Data (not dep. on Part. Cat.), Data Screens
BU_SORT1	☐	☐	BU_SORT1	CHAR	20	0	Search term 1 for business partner
BU_SORT2	☐	☐	BU_SORT2	CHAR	20	0	Search Term 2 for Business Partner
SOURCE	☐	☐	BU_SOURCE	CHAR	4	0	Data origin types
TITLE	☐	☐	AD_TITLE	CHAR	4	0	Form-of-Address Key
XDELE	☐	☐	BU_XDELE	CHAR	1	0	Central archiving flag
XBLCK	☐	☐	BU_XBLCK	CHAR	1	0	Central Block for Business Partner
AUGRP	☐	☐	BU_AUGRP	CHAR	4	0	Authorization Group

Figure 2.2 Table BUT000 Header Fields

TYPE field The TYPE field is one of the most misunderstood fields for the business partner. The sole purpose of this field is to determine the form of address for the business partner (i.e., whether the business partner is a person, organization, or group). You have the following options:

- A person is typically a contact person or employee in SAP CRM.
- An organization normally represents a company or other legal entity in SAP CRM.
- The group isn't normally used in most typical SAP installations.

BPKIND field BPKIND was designed for field control in the BDT. This field isn't normally used in newer installations, except as a way to classify SAP CRM partners. BU_GROUP is an important field because it's the technical

equivalent of the account group found in SAP ERP. This field controls the number range that will be created for a business partner.

Another interesting aspect of Table BUT000 is the storage of the business partner name. The name storage consists of many different fields and varies based on the type of partner (person, organization, or group). These fields are shown in Table 2.2.

Field Name	Field Description
NAME_LAST	Last name of person
NAME_FIRST	First name of person
NAME_ORG1	Name 1 of organization
NAME_ORG2	Name 2 of organization
NAME_ORG3	Name 3 of organization
NAME_ORG4	Name 4 of organization
NAME_GRP1	Name 1 of a group
NAME_GRP2	Name 2 of a group

Table 2.2 Multiple Name Fields in Table BUT000

It's important to note that NAME_ORG1 is equivalent to NAME_LAST, and NAME_ORG2 is equivalent to NAME_FIRST. In several applications where you search for business partners by name, these fields are typically combined for both search input and output.

CI include

The final feature of the table that we'll mention is the CI include for customer attributes. This include—called `CI_EEW_BUT000`—is delivered by SAP for enhancing the business partner with new attributes that are of a 1:1 relationship to the business partner. In Chapter 3, we'll explain how to extend the SAP CRM business partner.

Table BUT100

Business partner role

Table BUT100 describes the role of a business partner in SAP CRM. This originally controlled the tab order for the SAP GUI Transaction BP, and now it's used primarily for classification of business partners. This differs

from the partner function assignment of the business partner in the sales area data.

SAP CRM-specific data for a business partner consists of the segments detailed in Table 2.3, which are centered on the sales area data of the customer.

Data Segment	Table Name
Business hours	CRMM_BUT_FRG0060
Partner function	CRMM_BUT_FRG0081
Sales employee	CRMM_BUT_SEMPL00
Sales classification	CRMM_BUT_FRG0041
Status	CRMM_BUT_FRG0100
Sales Area Data (Sales, Billing, Shipping)	
Sales rule	CRMM_BUT_LNK0011
Sales data	CRMM_BUT_SET0010
Shipping rule	CRMM_BUT_LNK0021
Shipping data	CRMM_BUT_SET0020
Bill rule	CRMM_BUT_LNK0031
Billing data	CRMM_BUT_SET0030
Organization rule	CRMM_BUT_LNK0141

Table 2.3 Data Segment Tables of the Business Partner

Sales area data

The sales area data in SAP CRM corresponds to the sales area data normally found in SAP ERP. This data is normally used to control the creation of sales orders and assign the business partner to particular sales areas. There are two tables per sales area-related table. The tables that start with CRMM_BUT_LNK link a set of attribute data to a business partner. The tables that start with CRMM_BUT_SET contain a unique set of attributes for a given period of time. Each set has a set of Business Application Programming Interfaces (BAPIs) that follows the naming pattern `BAPI_BUPA_FRG<number>_*`, where `<number>` is the four-digit number of the attribute set.

The full data model for the SAP CRM business partner can be viewed via Transaction SD11 and looking at the data model PRM_BP.

Business Partner BAPIs

Normally when reading business partner data, you can use the delivered BAPIs. A BAPI is a remote-enabled function module delivered by SAP. These BAPIs normally start with `BAPI_BUPA`; some of the more commonly used BAPIs are listed here:

Common BAPIs

- `BAPI_BUPA_SEARCH`
- `BAPI_BUPA_CENTRAL_GETDETAIL`
- `BAPI_BUPA_CREATE_FROM_DATA`

As a general rule, ABAP programs in SAP CRM should not be created to directly read data from the tables unless performing a search. Instead, we recommend using the business partner BAPIs and API function modules to retrieve any data required. The primary benefit of this method is that you'll take advantage of the buffering logic built by SAP and the most optimized path to read this data.

We'll now look at three coding examples for business partner search, retrieving the details of a business partner, and creating a new business by using the business partner BAPIs.

Business partner data search

Listing 2.1 shows a common coding example to search for business partner data.

```
data: telephone type AD_TELNRCL,
      email type AD_SMTPADR,
      url type AD_URI2,
      addressdata type BAPIBUS1006_ADDR_SEARCH,
      centraldata type BAPIBUS1006_CENTRAL_SEARCH,
      businesspartnerrole type BU_ROLE,
      country_for_telephone type AD_COMCTRY,
      fax_data type BAPIBUS1006_FAX_DATA,
      others = BAPIBUS1006_OTHER_DATA ,
      searchresult type table of BAPIBUS1006_BP_ADDR,
      return type table of BAPIRET2.

 CALL FUNCTION 'BAPI_BUPA_SEARCH'
    EXPORTING
      TELEPHONE                  = telephone
```

```
      EMAIL                        = email
      URL                          = url
      ADDRESSDATA                  = addressdata
      CENTRALDATA                  = centraldata
      BUSINESSPARTNERROLE          = businesspartnerrole
      COUNTRY_FOR_TELEPHONE        = countryfortelephone
      FAX_DATA                     = fax_data
      OTHERS                       = others
    TABLES
      searchresult                 = searchresult
      return                       = return.
```

Listing 2.1 BAPI_BUPA_SEARCH

Find business partner details

Listing 2.2 shows a typical coding example to retrieve the central details for a business partner. Note we're using the function module `BUPA_CENTRAL_GET_DETAIL` instead of the BAPI because the BAPI won't return whether the business partner is a person, organization, or group. You only need to pass the partner number to retrieve the primary details of the business partner. Also keep in mind that the business partner number should be zero filled, so you may need to call `CONVERSION_EXIT_ALPHA_INPUT` to properly format the partner number.

```
DATA: lv_partner type bu_partner,
      ls_data type BAPIBUS1006_CENTRAL,
      ls_data_person type BAPIBUS1006_CENTRAL_PERSON,
      ls_data_organ type BAPIBUS1006_CENTRAL_ORGAN,
      ls_data_group type BAPIBUS1006_CENTRAL_GROUP,
      ls_data_info type BAPIBUS1006_CENTRAL_INFO,
      lv_category type BU_TYPE,
      lv_fullname_converted type AD_NAMCONV,
      ls_central_customer_ext type BUPA_CENTR_CUST_EXT,
      return TYPE table of BAPIRET2
* Retrieve the central data for a business partner(non-
address)
* Also includes any extension data we created
CALL FUNCTION 'BUPA_CENTRAL_GET_DETAIL'
 EXPORTING
   IV_PARTNER                    = iv_partner
  IMPORTING
    ES_DATA                       = ls_data
    ES_DATA_PERSON                = ls_data_person
    ES_DATA_ORGAN                 = ls_data_organ
```

```
    ES_DATA_GROUP                = ls_data_group
    ES_DATA_INFO                 = ls_data_info
    EV_CATEGORY                  = lv_category
    EV_FULLNAME_CONVERTED        = lv_fullname_converted
    ES_CENTRAL_CUSTOMER_EXT      = ls_central_customer_ext
  TABLES
    ET_RETURN                    = lt_return.

CASE LV_CATEGORY.
  WHEN '1'.
* Retrieve information from ls_data_person
  WHEN '2'.
* Retrieve information from ls_data_organ
  WHEN '3'.
* Retrieve information from ls_data_group
  WHEN OTHERS.
* Do Nothing
ENDCASE.
```

Listing 2.2 BUPA_CENTRAL_GET_DETAIL

Create new business partner

Listing 2.3 shows a common coding example to create a new business partner from scratch. A business partner in SAP CRM only requires that two fields are maintained to be created. The first field is the country of the business partner. The second field will be LAST_NAME for a person, NAME_ORG1 for an organization, or NAME_GRP1 for a group. All other fields are considered optional unless you've configured or extended the system to make a field required.

```
DATA:  lt_error         TYPE bapiret2_t,
       ls_error         TYPE bapiret2,
       lt_telephone     TYPE STANDARD TABLE OF bapiadtel,
       lt_fax           TYPE STANDARD TABLE OF bapiadfax,
       lt_uri           TYPE STANDARD TABLE OF bapiaduri,
       ls_mktlist_adr   TYPE crmt_mktlist_adr,
       ls_mktlist_org   TYPE crmt_mktlist_org,
       ls_bp_org        TYPE bapibus1006_central_organ,
       ls_telephone_org TYPE bapiadtel,
       ls_fax_org       TYPE bapiadfax,
       ls_uri_org       TYPE bapiaduri,
       ls_central       TYPE bapibus1006_central,
       ls_bp_add        TYPE bapibus1006_address.
```

```
data:  lv_no_duplicate_check
             TYPE BAPIBUS1006_HEAD-CONTROLDUPLICATEMESSAGE.
data: lv_bp             TYPE bu_partner,
      lv_address_num    TYPE AD_ADDRNUM,
      lv_address_guid   TYPE BU_ADDRESS_GUID,
      lv_partner_guid   TYPE BU_PARTNER_GUID.

CALL FUNCTION 'BUPA_CREATE_FROM_DATA'
   EXPORTING
      iv_category              = '2'
      is_data                  = ls_central
      is_data_organ            = ls_bp_org
      is_address               = ls_bp_add
      iv_duplicate_message_type = lv_no_duplicate_check
   IMPORTING
      ev_partner               = lv_bp
      ev_partner_guid          = lv_partner_guid
      ev_addrnumber            = lv_address_num
      ev_addrguid              = lv_address_guid
  TABLES
      it_adtel                 = lt_telephone
      it_adfax                 = lt_fax
      it_aduri                 = lt_uri
      et_return                = lt_error.
```

Listing 2.3 BUPA_CREATE_FROM_DATA

Address Data

Address data for a business partner includes the mailing address and communication data. The mailing address contains all of the necessary fields to support a postal or physical address. The communication data consists of contact information storage such as telephone numbers (including mobile), fax numbers, email address, and URI for web addresses or other Internet resources.

The address data for an SAP CRM business partner is maintained using the Business Address Service (BAS) or Central Address Management (CAM) of the SAP NetWeaver ABAP AS. The business partner data model maintains a link to the address management system via Table BUT020.

Call function modules

Normally, you don't read the address data directly from the database tables. Instead, you can use several function modules to handle this:

- `BUPA_ADDRESSES_GET`
  Gets the list of addresses for a business partner
- `BUPA_ADDRESS_GET_DETAIL`
  Gets the details for a particular address

For contact persons and the relationship address information, the function modules are different:

- `BUPR_CONTP_ADDRESSES_GET`
  Retrieves the addresses for a conact person relationship
- `BUPR_CONTP_ADDR_GET_DETAIL`
  Retrieves the details of a contract persoon relationship address

Listing 2.4 shows a typical example for retrieving the address data for a given business partner.

```
DATA: lv_partner TYPE bu_partner,
      lv_standard_addrnumber TYPE AD_ADDRNUM,
      lv_standard_addrguid TYPE BU_ADDRESS_GUID,
      lt_return TYPE TABLE OF BAPIRET2,
      ls_address TYPE BAPIBUS1006_ADDRESS,
      lt_adtel TYPE TABLE OF BAPIADTEL,
      lt_adfax TYPE TABLE OF BAPIADFAX,
      lt_adsmtp TYPE TABLE OF BAPIADSMTP,
      lt_aduri TYPE TABLE OF BAPIADURI.

CALL FUNCTION 'BUPA_ADDRESSES_GET'
  EXPORTING
    IV_PARTNER                    = lv_partner
  IMPORTING
    EV_STANDARD_ADDRNUMBER        = lv_standard_addrnumber
    EV_STANDARD_ADDRGUID          = lv_standard_addrguid
  TABLES
    ET_RETURN                     = lt_return.

REFRESH: lt_return.

CALL FUNCTION 'BUPA_ADDRESS_GET_DETAIL'
  EXPORTING
   IV_PARTNER             = lv_partner
```

```
    IV_ADDRNUMBER          = lv_standard_addrnumber
    IV_ADDRGUID            = lv_standard_addrguid
  IMPORTING
     ES_ADDRESS             = ls_address
  TABLES
     ET_ADTEL               = lt_adtel
     ET_ADFAX               = lt_adfax
     ET_ADSMTP              = lt_adsmtp
     ET_ADURI               = lt_aduri
     ET_RETURN              = lt_return.
```

Listing 2.4 Retrieve Address Data for a Business Partner

Relationship Data

Business partner interaction

Relationship data describes how two business partners in SAP CRM interact. Typical examples of relationships include contact person and employee responsible for a business partner such as a sales representative.

Table BUT050

Table BUT050 stores the relationships between business partners. This is the second-most important table in the SAP CRM business partner data model next to Table BUT000. The primary key consists of RELNR, PARTNER1, PARTNER2, and DATE_TO. PARTNER1 is the parent partner in a relationship that is expressed by PARTNER1 HAS RELATED PARTNER2. If you look at a contact person relationship, PARTNER1 is the company, and PARTNER2 is the contact person. The table contains a validity date, which means relationships can be time dependent. However, extra configuration work is required to activate this feature for end users of the system.

An important field not included in the primary key of this table is RELTYP, which defines the type of relationship. The most common relationships you'll see in your SAP CRM system are BUR001 (HAS CONTACT PERSON) and BUR010 (HAS THE EMPLOYEE RESPONSIBLE). SAP provides two secondary indexes for this table that allow lookup on PARTNER1 and PARTNER2 based on the relationship type.

Table BUT051

Table BUT051 contains specific information for a contact person relationship. In SAP CRM, a contact contains personal and work address information. Table BUT051 contains the work address, which is equivalent to the relationship address for the contact person.

Retrieve contacts for business partner

Listing 2.5 shows how to retrieve all of the contact persons for a given business partner. This is a common requirement for reporting within an SAP CRM system.

```
DATA: lv_partner TYPE bu_partner,
      lt_partner type table of bu_partner,
      lt_addresses type table of BAPIBUS1006002_ADDRESSES_I,
      lt_return type table of bapiret2.
field-symbols: <fs_partner> type bu_partner.
* LV_PARTNER is the main partner that you wish to find contact
* persons for
SELECT partner2 FROM but050 INTO TABLE lt_partner
            WHERE partner1 = lv_partner
            AND date_to le sy-datum
            AND reltyp = 'BUR001'.

LOOP AT lt_partner ASSIGNING <fs_partner>.
  refresh: lt_addresses, lt_return.
  CALL FUNCTION 'BUPR_CONTP_ADDRESSES_GET'
    EXPORTING
      IV_PARTNER                 = lv_partner
      IV_CONTACTPERSON           = <fs_partner>
    TABLES
      ET_ADDRESSES               = lt_addresses
      ET_RETURN                  = lt_return .
ENDLOOP.
```

Listing 2.5 Contact Person Retrieval for a Business Partner

Attachments

CL_CRM_DOCUMENTS API

Attachments for business partners are typically files such as pictures, Microsoft documents, or other types of files. They are stored in the SAP CRM Document Management system, and the best way to access them is using the `CL_CRM_DOCUMENTS` API. We don't recommend even trying to retrieve related documents outside of this API. The layers of abstraction

for this piece of SAP CRM normally befuddle the most senior developers working with the solution.

Listing 2.6 is an example to retrieve a list of documents for a business partner and then retrieve each individual document. This example calls the method `CL_CRM_DOCUMENTS_API=>GET_WITH_FILE` to download them to the end-user's PC directly via the SAP GUI. You can also call the method `GET_WITH_TABLE` to store the document into an internal table instead.

```
DATA: ls_business_object TYPE SIBFLPORB,
      lt_phioloios TYPE SKWF_LPIOS,
      lt_ios_properties_result TYPE CRM_KW_PROPST,
      ls_ios_properties_result type crm_kw_props.
DATA: ls_loio TYPE SKWF_IO,
      ls_phio TYPE SKWF_IO,
      lt_bad_ios TYPE SKWF_IOERRS,
      ls_properties TYPE SDOKPROPTL,
      lv_filename TYPE SDOK_FILNM,
      lv_directory TYPE SDOK_CHTRD.
FIELD-SYMBOLS: <fs_phioloios> TYPE SKWF_LPIO.
* The instance ID is the GUID of the partner
* The type ID is business object BUS1006, which is SAP Busine
ss Partner
lv_directory = 'C:'.
ls_business_object-instid = iv_guid.
ls_business_object-typeid = 'BUS1006'.
ls_business_object-catid = 'BO'.
CALL METHOD CL_CRM_DOCUMENTS=>GET_INFO
  EXPORTING
    BUSINESS_OBJECT = ls_business_object
  IMPORTING
    PHIOLOIOS = lt_phioloios
    IOS_PROPERTIES_RESULT    = lt_ios_properties_result.

LOOP AT lt_phioloios ASSIGNING <fs_phioloios>.
  CLEAR: ls_loio, ls_phio, ls_ios_properties_result,
         lv_filename, ls_properties.
  ls_loio-OBJTYPE = <fs_phioloios>-OBJTYPELO.
  ls_loio-class = <fs_phioloios>-CLASSLO.
  ls_loio-objid = <fs_phioloios>-OBJIDLO.
  ls_phio-OBJTYPE = <fs_phioloios>-OBJTYPEPH.
```

```
  ls_phio-class = <fs_phioloios>-CLASSPH.
  ls_phio-objid = <fs_phioloios>-OBJIDPH.
 READ TABLE lt_ios_properties_result INTO ls_ios_properties_
result
       WITH KEY objtype = ls_loio-objtype
                class = ls_loio-class
                objid = ls_loio-objid.
 READ TABLE ls_ios_properties_result-properties INTO ls_
properties
      WITH KEY name = 'KW_RELATIVE_URL'.
 lv_filename = ls_properties-value.
 CALL METHOD CL_CRM_DOCUMENTS=>GET_WITH_FILE
   EXPORTING
    LOIO = ls_loio
    PHIO = ls_phio
    FILE_NAME = lv_filename
    DIRECTORY = lv_directory
   IMPORTING
    BAD_IOS = lt_bad_ios
   EXCEPTIONS
    NOT_TRANSFERRED = 1
    others          = 2.
ENDLOOP.
```

Listing 2.6 Retrieve Document Attachments for a Business Partner

Now that we've reviewed the data model of the business partner, let's examine how the goods and services that are sold to the business partners are modeled within your SAP CRM system.

2.3 Products

Track goods and services

An important part of the relationship between a company and its customers are the goods and services sold. In SAP CRM, products are used to model and track those goods and services sold. To better understand the product in SAP CRM, we'll review the business definition and the underlying technical design of the product master in the following sections. Finally, we review some of the common APIs available in SAP CRM to access or update the product master data.

9.1 SAP NetWeaver BW Data Source Enhancements via BAdIs

Transfer information to BW

A common requirement in many SAP CRM implementations is to transfer information to the SAP NetWeaver BW system for analytics and reporting. For most SAP CRM customers, business transactions make up the primary information transferred for reporting. Until SAP made the interactive reports available in the newer releases of SAP CRM, there were no other methods to build reports on SAP CRM data without writing a custom ABAP program.

Manually enhance data source

As we discussed in Chapter 3, you can enhance the SAP CRM system to add custom attributes to the business transactions. If you use the standard SAP tools such as the Application Enhancement Tool (AET) or Easy Enhancement Workbench (EEWB), the custom fields can be automatically added to the BW data source in SAP CRM. However, there are many situations in which you might want to add information to the data source that is either not a custom field or for which you did not check the option in the EEWB or AET to add the custom attribute. To get around this limitation, you can manually enhance the data source. By enhancing the data source, the new attribute will be made available to your SAP NetWeaver BW system and can be used in reports that you create on that system.

Prerequisites

To manually enhance the data source, it must be installed on your SAP CRM system for use by SAP NetWeaver BW. This process typically involves installing the data source using Transaction RSA5 in the SAP CRM system.

Note

We won't cover the installation of the data source in this chapter. The SAP best practices available at *http://help.sap.com/bp-crm* contain guides that detail how to install the SAP NetWeaver BW data sources on your SAP CRM system.

After the data source has been installed, you're ready to enhance the data source. This process generally requires two different steps. The first step is to enhance the data source structure. The second is to implement

the BAdI CRM_BWA_MFLOW to enhance the extractor program for the data source. We'll discuss these steps in the next subsections.

9.1.1 Enhance the Data Source Structure

To enhance the structure of the data source, there are several steps that you'll need to follow.

Transaction RSA6

Go to Transaction RSA6, and locate the data source that needs to be enhanced as shown in Figure 9.1. For this chapter, we'll examine enhancing the data source 0CRM_SALES_ACT_1, which is used to transfer data from SAP CRM to BW for activity transactions. If you recall from Chapter 2, you created a new business transaction called a social media call report. For this social media call report, you added a new field called ZZHASHTAG via the EEWB. During that enhancement, you didn't choose to add that field to the BW source structure via the EEWB tool.

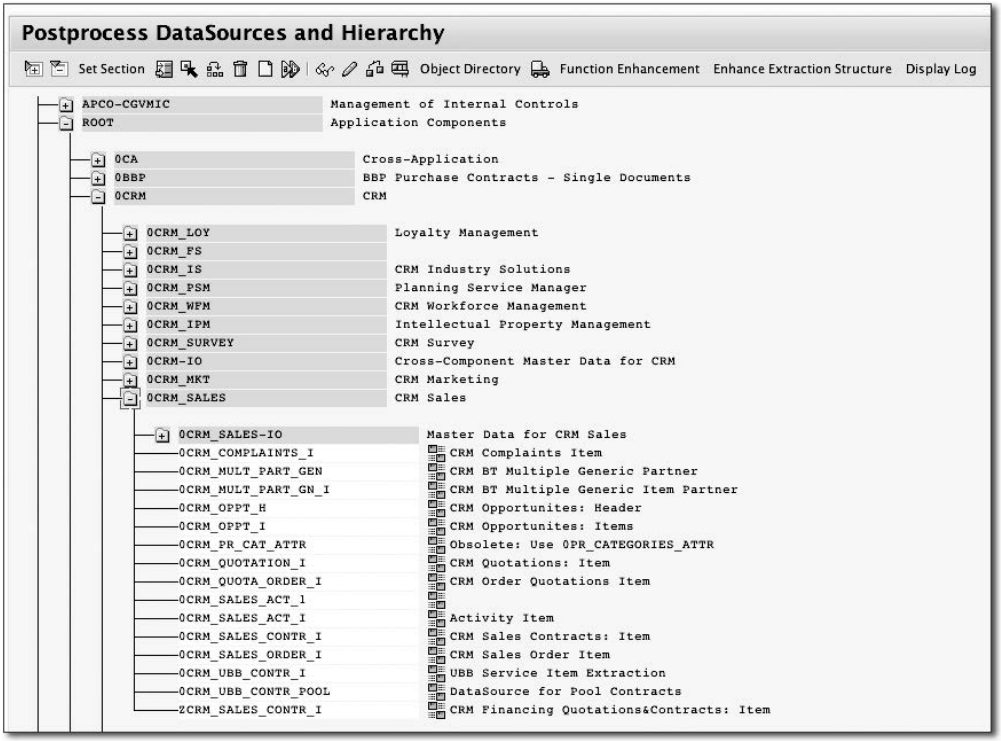


Figure 9.1 Transaction RSA6

Manually add ZZHASHTAG to the report

Now you'll need to manually add the field because you don't want to regenerate your EEWB extension. To do this, follow these steps:

1. Select the 0CRM_SALES_ACT_1 data source, and drill into the display.
2. Click on CRMT_BW_DS_ACTIVITY to pull up the data source structure in a Transaction SE11 view as shown in Figure 9.2.

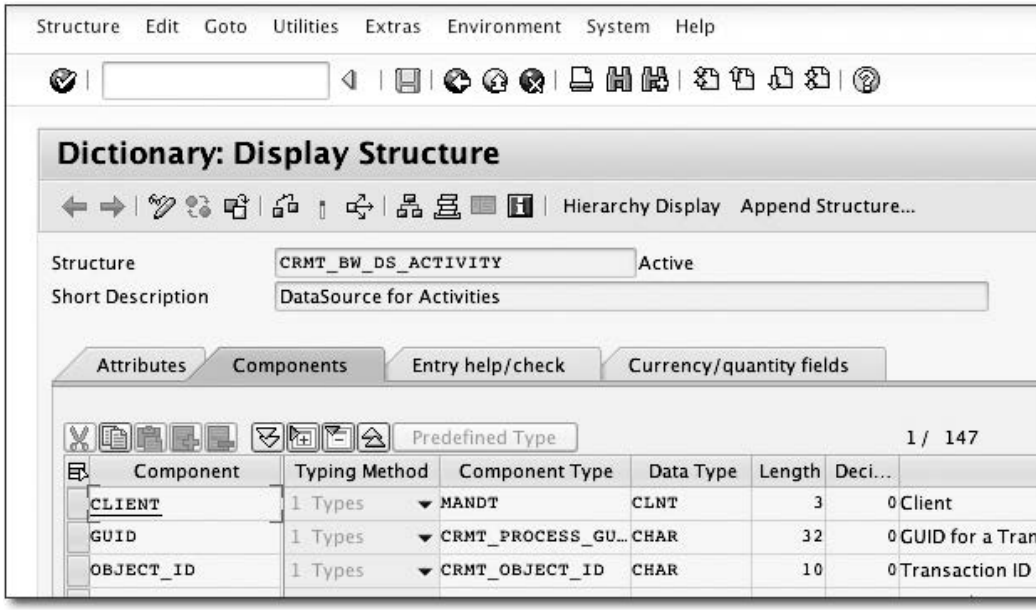


Figure 9.2 Data Structure CRMT_BW_DS_ACTIVITY

3. Click on the APPEND STRUCTURE button. Your structure doesn't contain any append structures, so you'll be prompted for the name of your new append structure as shown in Figure 9.3.

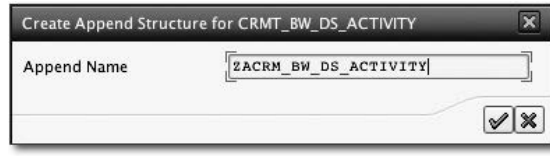


Figure 9.3 Create Append Popup Window

Add new append structure

4. Call the append "ZACRM_BW_DS_ACTIVITY".

5. Click on the checkmark icon.
6. Define fields in the append structure on the next screen. For this example, enter the description of the structure as "Activity BW Data Source Extension". In the COMPONENT section, add a new field called "ZZHASHTAG" that is of type ZZHASHTAG.
7. Save your work and assign your append structure to a transportable package.
8. You'll be prompted for a transport request. After you provide that information, the append structure will be in inactive status.
9. Activate your structure. Your new append structure will contain a single field called ZZHASHTAG and appear as active (see Figure 9.4).

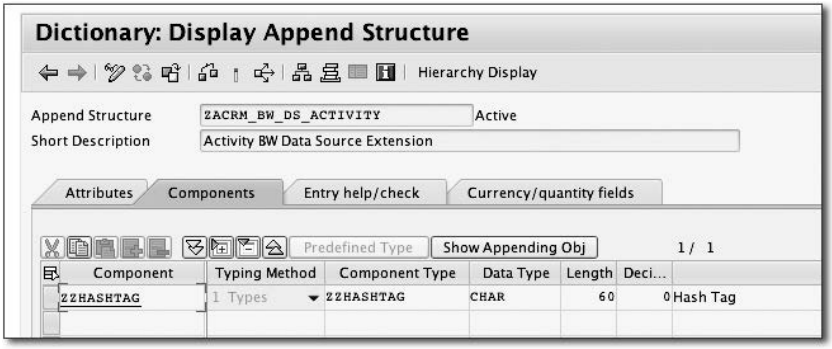


Figure 9.4 Fields of the New Append Structure

After the structure is activated, the fields show up in Transaction RSA6. By default, all custom fields added to a data source via the append structure are hidden in the data source definition as shown in Figure 9.5 (at the bottom of the screen, the new field has the HIDE FIELD checkbox selected).

Enable data to be transferred

To allow that data to be transferred to SAP NetWeaver BW, remove the HIDE FIELD flag, and the SAP NetWeaver BW system will be able to see that those fields are part of your data source definition (see Figure 9.6). Now that you've extended the structure of the data source, you'll need to build logic that will enhance the extractor programs to put values in your fields.

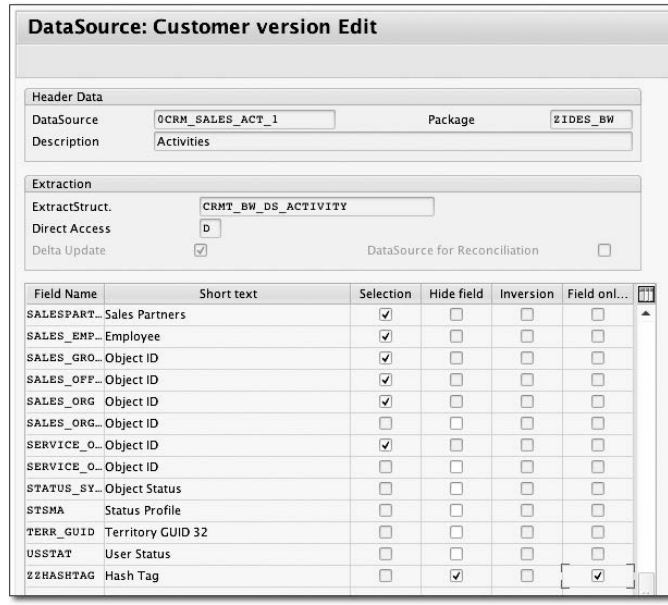


Figure 9.5 New Field Hidden

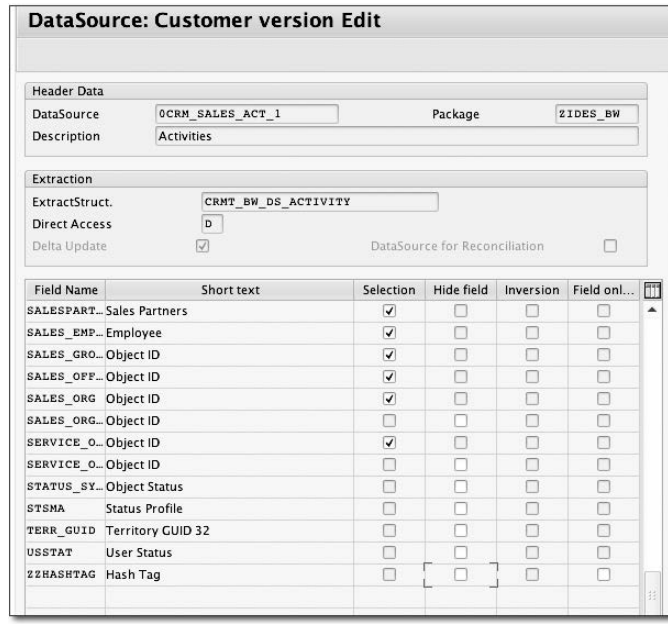


Figure 9.6 New Fields Unhidden

9.1.2 Extractor Program Logic

CRM_BWA_MFLOW

Now you're ready for the second step of the process. For SAP NetWeaver BW data sources, the traditional method of enhancing the extractors has been through user exits. In the SAP CRM system, those user exits are available but not used. Instead, SAP CRM has a specific BAdI for enhancements of the extractors for SAP CRM-specific data sources. This BAdI is called `CRM_BWA_MFLOW`. If you were using the EEWB or AET, an implementation of this BAdI would be automatically generated in your system.

Manual implementation

However, in this example, because you decided not to use the tools provided by SAP and instead chose to work manually, you must implement this BAdI manually. A benefit of manually implementing this BAdI is that you can implement virtual attributes that don't correspond to physical values stored on the SAP CRM database or attributes that cross business objects. A great example is that you might want to transfer the country of the sold-to party in the data source; however, it isn't stored on your business transaction. You can then instead populate the attribute virtually by using the `CRM_BWA_MFLOW` BAdI.

In the following subsections, we'll walk you through the steps to create the BAdI implementation and test your work. We'll also provide a quick overview of how to accomplish this with AET for your comparison.

Creating the BAdI Implementation

CRM_BWA_MFLOW implementation

Follow these steps to implement the BAdI:

1. Go to Transaction SE18, and open up the BAdI definition for `CRM_BWA_MFLOW`.
2. Choose Implementation • Create from the menu; you'll then see a popup window as shown in Figure 9.7.
3. Fill in the Implementation Name field; for this example, enter "ZCRM_TPP_BOOK_MFLOW".
4. Click the Checkmark button. On the next screen, you'll need to enter a description for your BAdI implementation. For this example, enter "Activity Data Source Enhancement" as shown in Figure 9.8.

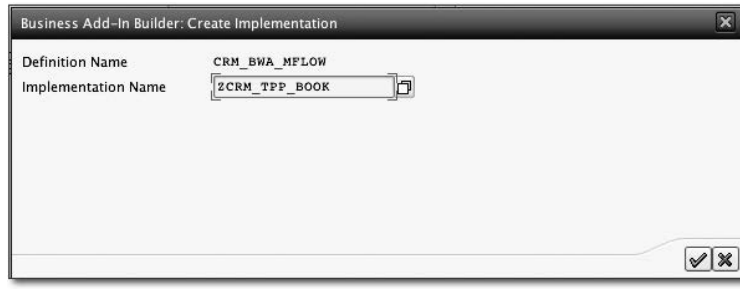


Figure 9.7 Create a New BAdI Implementation for CRM_BWA_MFLOW

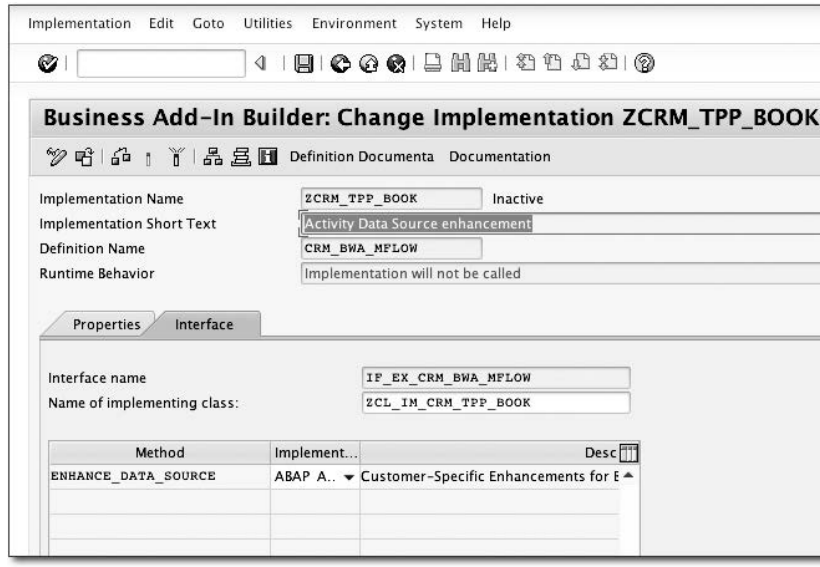


Figure 9.8 Implementation Definition ZCRM_TPP_BOOK

5. Click SAVE. You can now code the method `ENHANCE_DATA_SOURCE` of your BAdI implementation.
6. On the next tab, double-click on the method. The resulting screen allows you to add code to populate or change values that will be sent to the SAP NetWeaver BW system as shown in Figure 9.9.

Method code ENHANCE_DATA_SOURCE

The code for the `ENHANCE_DATA_SOURCE` method follows a general pattern, which is very straightforward. You first must check to see if the data source name matches the data source you want to enhance. This is provided by the `I_DATASOURCE` parameter.

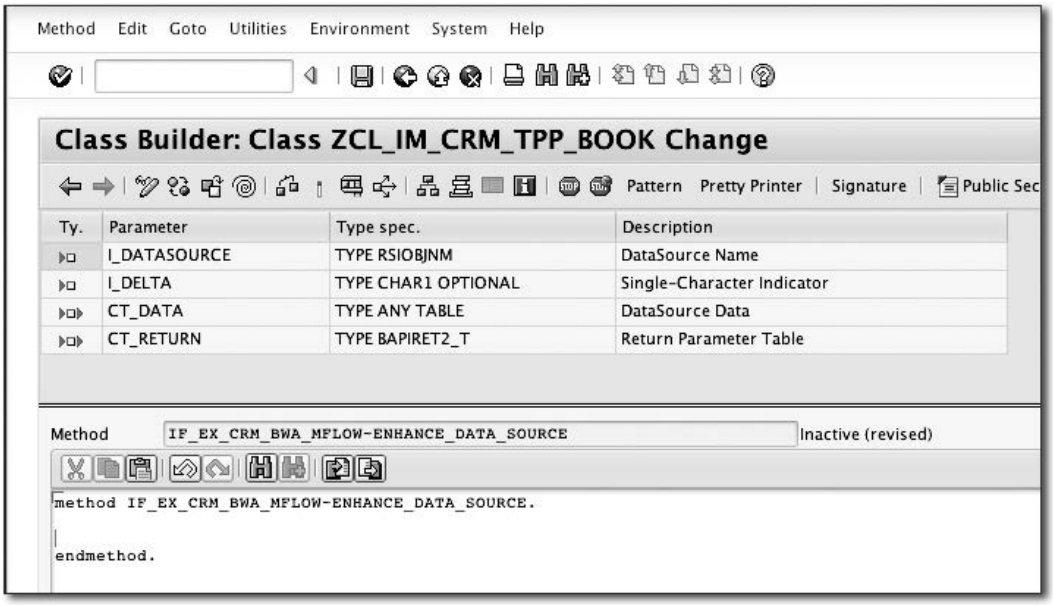


Figure 9.9 Signature of the ENHACE_DATA_SOURCE Method

Next, transfer the data to a local typed copy so you can change the fields. To do this, take the `CT_DATA` structure, and move the internal table to a local internal table. Finally, loop on the internal table. For each entry, use the GUID provided to find any other information on the business transaction, and then update the corresponding field.

In this example, `ZZHASHTAG` is stored on the `CRMD_CUSTOMER_H` table. You'll use the `CRM_CUSTOMER_READ_OW` function module to find the value of the `ZZHASHTAG` for each activity transaction extracted.

As the last step, you'll transfer your local copy of the data extract back to the `CT_DATA` parameter. Listing 9.1 shows the completed code to populate the `ZZHASHTAG` field through the `ENHANCE_DATA_SOURCE` method.

```
method IF_EX_CRM_BWA_MFLOW~ENHANCE_DATA_SOURCE.
  data:
    lt_data      type table of crmt_bw_ds_activity,
    wa_data      like line of lt_data,
    ls_customer_h_wrk type CRMT_CUSTOMER_H_WRK.
```

```
  field-symbols:
      <ls_data> like line of lt_data.

  case i_datasource.
* activities data source
    when '0CRM_SALES_ACT_1'.

* copy data from extract structure to internal table.
      lt_data[] = ct_data[].
      refresh ct_data.

      loop at lt_data assigning <ls_data>.

        clear: ls_customer_h_wrk.
        CALL FUNCTION 'CRM_CUSTOMER_H_READ_OW'
         EXPORTING
           IV_GUID                = <ls_data>-guid
         IMPORTING
           ES_CUSTOMER_H_WRK      = ls_customer_h_wrk
         EXCEPTIONS
           HEADER_NOT_FOUND       = 1
           OTHERS                 = 2.

        <ls_data>-ZZHASHTAG = ls_customer_h_wrk-zzhashtag.
      endloop.
* send the changed data back
      ct_data = lt_data[].
    when others.
  endcase.
endmethod.
```

Listing 9.1 Example Code to Populate ZZHASHTAG via ENHANCE_DATA_SOURCE

Testing Your Work

Transaction RSA3: extractor checker

To test your work, use Transaction RSA3. This is the extractor checker transaction that allows you to see if your BAdI is properly populating the new data source structure. As you can see from Figure 9.10, you can specify the name of your data source and pass selection parameters in to execute a test run of the extractor logic. For this example, enter "0CRM_SALES_ACT_1" as the data source name.

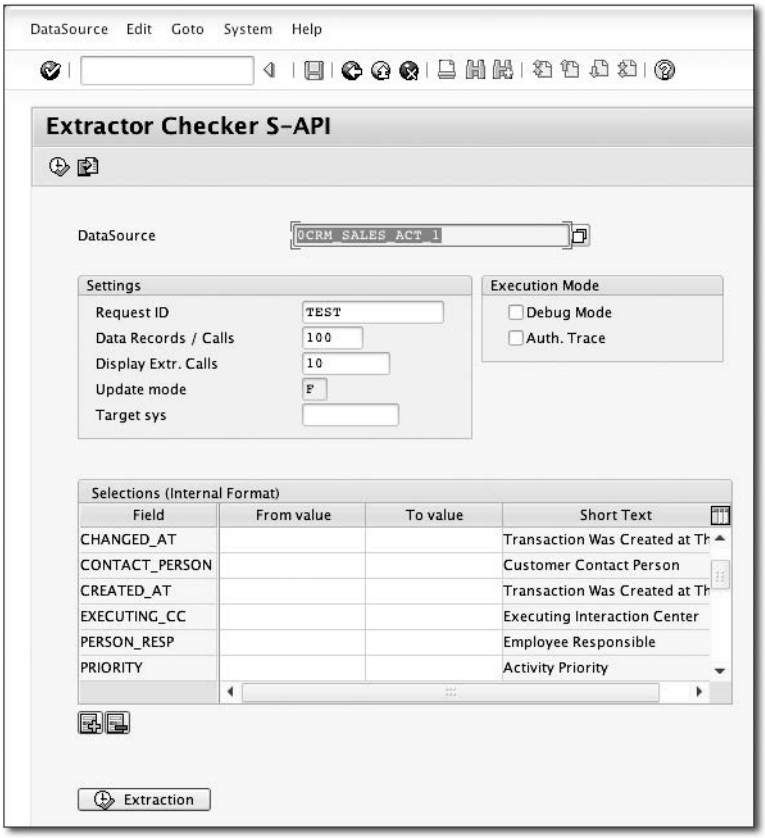


Figure 9.10 Extractor Checker

Now follow these steps:

1. Scroll down under SELECTIONS (INTERNAL FORMAT), and enter "ZSCR" as the FROM VALUE in the row with the key field of "PROCESS_TYPE".
2. Click the EXTRACTION button to run the extractor. After the extraction, another screen appears that allows you to display your results as shown in the Figure 9.11.
3. Click the ALV GRID button to get an ALV grid display of your results. This grid, as shown in Figure 9.12, allows you to verify that the extractor is now executing your BAdI logic to fill in the additional attribute to the extract structure.

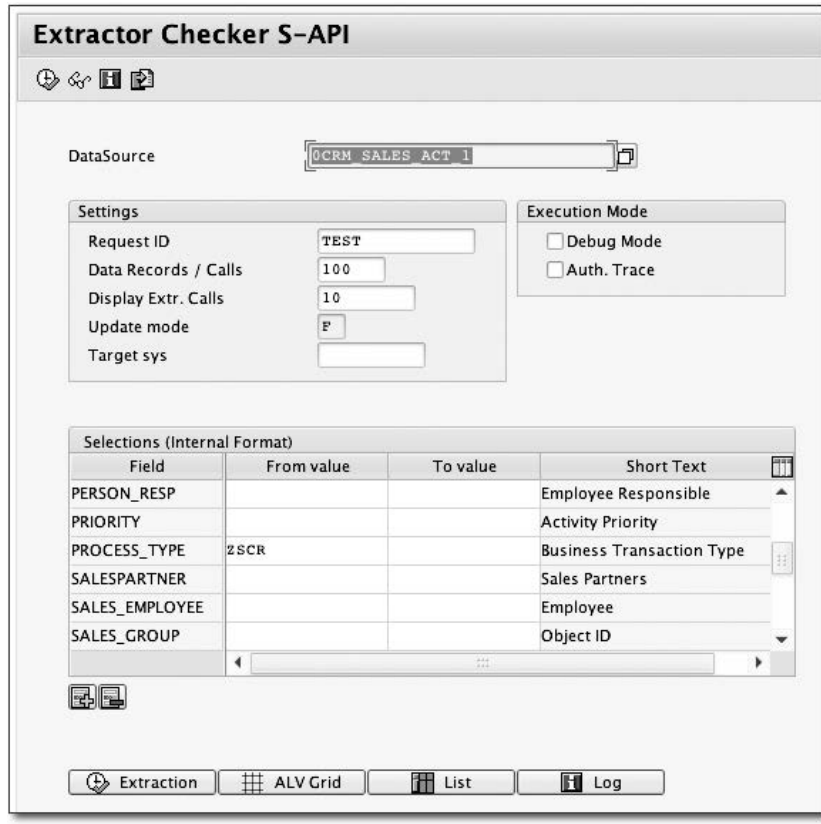


Figure 9.11 Transaction RSA3 Screen after the Extraction Completes

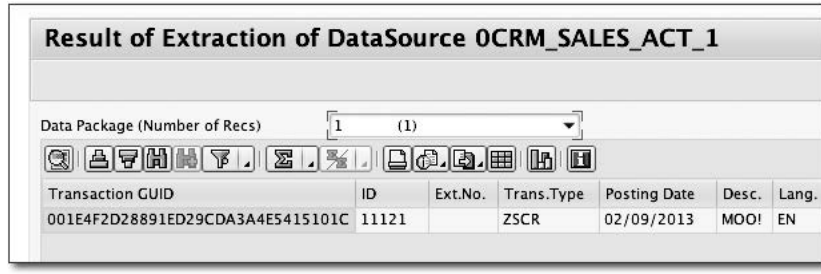


Figure 9.12 ALV Grid Display of Extracted Records

You can use the Detail display button to open a popup as shown in Figure 9.13 that will show all of the columns of the extract structure in a listing. You can see that your new Hash Tag field is indeed being populated by your extract logic.

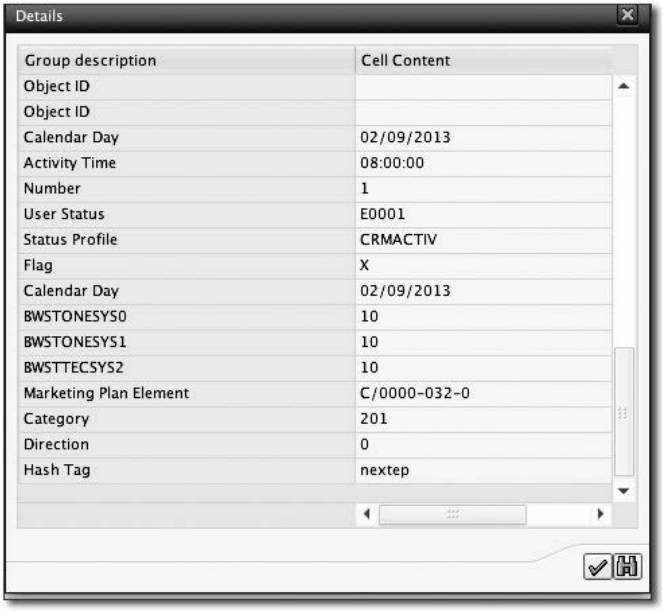


Figure 9.13 Detail Display of the Extracted Record

Comparison against AET

For the purpose of this example, we walked through the steps of adding your custom field manually. Understanding this manual technique allows you to add fields to the extractor data source that may not be available via the AET. This could include values from related transactions instead, such as follow-up activities or tasks. However, if you had created your field using the AET, then none of the coding would be required that we just described. Instead, SAP has automatically included exits in the standard extractors to bring in the AET fields, provided that you checked the BW Reporting checkbox when you created the field in the AET (see Figure 9.14).

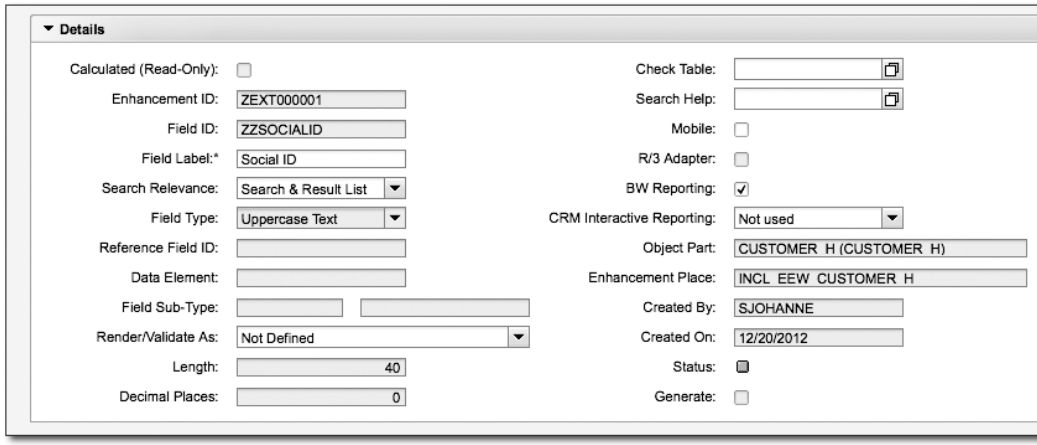


Figure 9.14 AET Field Definition for BW Relevant

Final Steps

After you've validated that the data source is properly enhanced, you'll need to work with your BW configurator/developer to enhance the SAP NetWeaver BW data structures to include your new field. This work is beyond the scope of this book; if you're interested in these steps, review the book *Data Modeling in SAP NetWeaver BW* by Frank K. Wolf and Stefan Yamada (SAP PRESS, 2011).

Standard extractor function module

If you look at the standard extractor function module `CRM_SALES_ACT_1_MAP`, there is a call to the method `MAP_EXT_FIELDS_FROM_BDOC` of the class `CL_CRM_1O_EXTENSIBILITY_TOOLS`. This method takes a reference GUID, reference GUID type, objects, and `bdoc` structure. It then returns the changed extract structure. In the `CRM_SALES_ACT_1_MAP` function module, it looks for extensions in three key areas: `ORDERADM_H`, `CUSTOMER_H`, and `ACTIVITY_H`.

Method steps

The method first finds the includes that are normally used to extend the object. These are predefined as part of the AET metadata model. Next, the tool determines if any of these includes have been implemented in the system. If the include has been implemented, which means that it contains an actual new field and not just the default dummy field, the tool then calls a mapper that moves the data from the passed in `bdoc` structure to the extract structure.

Tips & Tricks

You need to make sure that the fields in the extract structure are named exactly identical to the fields in the extension of the program.

The BDoc that is used is the source data provided by the SAP CRM Middleware being sent to the SAP NetWeaver BW system. SAP CRM uses an outbound BDoc when transactions are saved to send data to the SAP NetWeaver BW system.

9.2 Interactive Reporting Enhancements

In most SAP CRM implementations, there is always a requirement to report on transaction data based on custom attributes added to your system. To avoid building new custom reports or tools, you can instead extend existing reporting tools such as Interactive Reporting. Interactive Reporting allows users to perform ad hoc analytical reporting on transactional data within SAP CRM without requiring a separate SAP NetWeaver BW system. As of SAP CRM 7.0 EHP1, it's possible to add custom fields from the business transaction to the interactive reports using the AET. There is, however, a limitation on the interactive reports because you can't manually extend these reports and must use the AET to extend the report data sources.

Interactive Reporting extension method

The method to extend the Interactive Reporting data sources is through setting the INTERACTIVE REPORTING flag attribute for the custom field you created. As you recall, you can access the details of the custom field you created using the AET by first bringing up the CREATE INTERACTION LOG screen in the Web Client. Click on the menu option SHOW CONFIGURABLE AREAS, and then click on the INTERACTION LOG DETAILS section. This brings up the screen configuration popup window. Select DISPLAY ENHANCEMENTS, and you'll be prompted to choose the object type, which is INTERACTION LOG. You'll now see a list of the fields you've added. To flag the attribute, go into the details of the field that you added, and choose one of three options as shown in Figure 9.15.

Index

$TMP package, 400

A

ABAP class, 400
ABAP Debugger, 181
ABAP developer, 50
ABAP-based system, 396
Abstraction, 95
Access sequence, 339
  create, 332
Account and Contact Management, 28
Action, 246, 249
  define and schedule, 285
  define processing attributes, 269
  definition, 250
  list of, 253
  merging, 255
  new, 285
  nonoutput tasks, 278
  partner determination, 270
  rules, 254
  scheduling, 272
  select processing type, 270
  send to specified partner function, 254
Action profile, 247
  actions, 249
  attributes, 248
  create new, 267
  existing, 247
  for business transaction types, 247
Activities, 29
Activity, 159
  data, 351
  transaction, 226, 381
Activity Management, 29
ACTIVITY_H BADI, 301
Ad hoc analytical reporting, 393
Add new field, 106
Address data, 56, 64
  function module, 65
AET, 47, 48, 118, 393, 426
  add custom field, 391
  create new table, 128
  nonexpert mode screen, 121
Analytics and reporting, 380
API function module, 102, 312
Append, 126
Append structure, 352
  assign to tranportable package, 383
  button, 365, 382
Appending fields, 113
Application
  CRM_ORDER, 247
Application Enhancement Tool → see AET
Application programming interface (API), 38
ASCII code page, 221
ASUG, 422
Attribute, 96, 342, 365
  add additional, 141
  code logic, 266
  common, 94, 138
  create new, 139
  custom, 267
  logical grouping, 144
  organize by similar sets, 94, 138
  values, 101
Attribute set, 72, 94, 141, 147
  assign to business partner, 141
  assign to product category, 147
  header, 97
  receive metadata, 103
  update values, 103

B

BAdI, 349
  copy, 357
  create filter value, 280
  create implementation, 357
  CRM_BWA_MFLOW, 381, 385

BAdI (Cont.)
CRM_MKT_EXP_CAMP_DAT, 365
CRM_MKTLIST_BADI, 350
CUSTOMER_H, 313
EXEC_METHODCALL_PPF, 279
extend values to, 413
implement manually, 385
implementation, 303, 338
make more effective, 410
ORDER_SAVE, 307, 308
test, 373
test implementation, 361
BAPI, 61, 204
Batch job
schedule, 272
Batch processing, 262
BDoc, 202, 393
BRF+, 176
BTE framework, 155
business logic, 174
code, 171
code field updates, 179
Customizing, 177
customizing data, 157
objects, 162
specify business object categories, 179
BUS2001, 81
Business activity, 226
header data, 226
Business Add-In → see BAdI
Business logic, 155, 245
custom, 185
define, 167
Business Object Builder, 81, 262
Business object categories, 178
Business Object Layer (BOL), 93
Business Object Repository (BOR), 248
Business object type, 81
assign segments to, 83
Business partner, 28, 55
attachment, 67
attributes, 99
business process, 78
common characteristics, 94
create new, 64
create with function module call, 236
Business partner (Cont.)
create/update, 364
data, 95
get address data, 66
get document attachments, 69
GUID, 101
key table, 57
maintain, 362
read data, 61
retrieve contact person, 67
role, 59
search, 144
value set, 102
Business process type, 80
Business transaction, 77, 158
create required field, 295
custom attributes, 380
data, 284
defaulting values, 313
events, 315
leads, 82
lifecycle, 159
partner determination procedure, 334
raise error messages, 304
require additional fields, 298
save with errors, 307
see details, 364
store custom attribute, 352
Business transaction event framework → see BTE framework
Business workflow, 246
Business-to-business (B2B), 34
Business-to-consumer (B2C), 34

C

Call report, 296
Callbacks, 170, 174
Campaign, 366
execute/export information, 365
text messages, 376
third-party system, 376
type, 368
Campaign execution, 374
batch job, 375
Campaign Management, 32

Category data, 72
Central administration table, 78, 79
Characteristic, 95
  attribute sets, 95
  create, 96
CHECK_BEFORE_SAVE, 310
Checks, 158
CI include, 59
CL_CRM_DOCUMENTS API, 67
Class, 95
  CL_ACTION_EXECUTE, 282
Cluster tables, 185
Code change, 358
Code execution, 246
Code modification, 415
COMM_ATTRSET, 144
Commit work, 317
Communication media, 366
Communication medium
  set up, 366
Communication method, 366
Complaint and Return Management, 31
Component usage, 149
Condition configuration concept, 259
Condition definition, 262
Condition type
  schedule condition, 261
  start condition, 261
Context class, 249
Context nodes, 121
Contract Management, 31
Conversion program, 223
  troubleshoot, 223
Convert data, 223
Core modification, 414
Create new extensions, 106
Create UI component, 128
CREATE_ACTIVITIES method
  modify, 359
CRM_MKTTGGRP_EXPORT_BATCH, 375
Custom ABAP report, 397
Custom business transaction fields, 262
Custom report transaction, 398
Custom rule, 342
Customer attribute, 59
Customer base, 94, 138
Customer data, 26
Customer event customizing tables, 169
Customer experience management, 27
Customer namespace, 113
Customer relationship management (CRM), 25
CUSTOMER_H segment, 86
Customer-defined attributes, 144
Customizing data, 157
Customizing tables, 169

D

Data
  change fields, 387
  extractor, 394
  model for sales, 54
  object, 55
  segment, 79
  transfer to local copy, 387
Data source, 379
  0CRM_SALES_ACT_1, 381
  definition, 383
  enhancement, 385
  extend, 393
  manually enhance, 380
  review installed, 394
  test, 396
Date rule, 317
  ABAP logic, 320
  function module logic, 321
  underlying XML, 319
  use in transaction, 326
Debug mode, 396
Default values, 315
Defaulting a value, 198, 313
Define, 401
  technical names, 121
  the list of possible values, 142
Determination Technology option, 254
Determine timing, 301
Dialog, 253
Display in Toolbox option, 254
Distribution method, 366
DO_NOT_SAVE exception, 312

Duration, 321
Dynamic variant, 375

E

Easy Enhancement Workbench → see EEWB
Easy extension of new fields, 118
E-commerce, 34
E-commerce → see SAP CRM Web Channel Experience Management
EDI partner number, 212
EEWB, 47, 105, 394
  project, 107
Enhancement
  metadata structure, 126
  set, 149
Error message, 296
Event, 160
  publish to BTE framework, 171
Event handler, 157, 174, 190
  basic code structure, 194
  code structure, 194
  define, 166
  find list, 180
  import parameters, 193
  optional parameters, 175
  required import parameters, 175
  standard, 176
Event handler function module, 170, 175
  create new, 180
  isolate, 187
Event publishing process, 173
Event trace, 181
  review, 182
Execute custom logic, 290
Execution options, 269
Execution time, 164, 197
Expansion button, 112
Expert mode, 112, 123
Export, 409
  list, 370
Extend object, 392
Extension, 109
  type, 111
External Interface (XIF) adapter → see XIF adapter
External list, 361
External List Management, 350
  custom attribute, 351
  data structure, 351
  mapping format, 350
  target data structure, 355
External URL, 397
Extract data, 396
Extractor checker, 388
Extractor function module, 396
Extractor program
  enhance, 383
  logic, 385

F

Field
  placement, 113
  required, 297
  symbol, 190, 194
File port definition, 207
Filter value, 280, 357
  corresponding implementation, 280
Filter-dependent, 350
Flat file, 226, 354
  map in LSMW, 214
Fragment data → see attribute sets
Function group, 187
  create, 188
  method, 410
Function module, 396
  /CRMBW/EXTEND_FROM_DS, 396
  API, 102
  assign to object function, 196
  BUPA_CREATE_FROM_DATA, 365
  create, 411
  CRM_ACTIVITY_H_SAVE_EC, 178
  CRM_CUSTOMER_H_READ_OW, 317
  CRM_EVENT_PUBLISH_OW, 171
  CRM_EVENT_SET_EXETIME_OW, 173
  CRM_MKTBP_CHANGE_BP, 103
  CRM_MKTBP_READ, 103
  CRM_MKTBP_READ_DATA, 102
  CRM_ORDER_MAINTAIN, 351

Function module (Cont.)
CRM_ORDER_READ, 87, 284
CRM_ORDERADM_H_READ_OW, 194
CRMXIF_ORDER_SAVE, 225
manage execution time, 197
modify, 192
register, 196
search, 103
view source code, 184
Function module builder, 204

G

General data, 56
Generated object, 117
Global buffer table, 171
Global variables, 410
GUID, 54, 228

H

Hashtag, 107, 295
Header, 54, 296
Header extension table, 79
Hide field flag, 383
Hierarchy, 72
R3PRODSTYP, 72

I

IDoc, 204
error list, 225
message type, 209
structure, 218
Implementation, 280
Implementation class
CL_DEF_IM_CRM_MKTLIST_BADI, 356
Implicit enhancement, 407
common uses, 409
export, 409
Import data, 204
Import parameter, 351
Inbound IDoc processing, 207
Inbound processing, 210
Include, 392
Incompleteness group, 299
Incompletion procedure, 295
multiple, 299
Individual action, 268
Interaction Center, 34
Interaction log business transaction type, 106
Interaction log object, 120
Interactive Reporting, 393
client, 396
create objects, 396
data source location, 395
data sources, 393
Intermediate values, 409
Internal table, 387
Internet Pricing Configurator, 41
IT Service Management (ITSM), 32
Item data, 79
ITIL, 32

J

Job
specify run time, 277
step, 275

L

Launch external application, 398
Launch transaction
create logical link, 402
Lead, 351
Lead Management, 30
Leads business transaction, 82
Legacy data, 201
Legacy System Migration Workbench → see LSMW
Link group, 402
List Management, 33
Load and extract data, 201
Local object, 400
Logical link, 402
Logical object, 54
Logical system
create, 239
identifier, 221
Long text data, 228

Loyalty Management, 33
LSMW
for inbound IDoc processing, 207
map flat file, 214

M

Maintenance view
CRMV_EVC_ALL, 167
CRMV_FUNC_ASSIGN, 166
Manual enhancement, 137
Manual extension, 394
Map source fields, 220
Map source structures, 218
Mapping format, 350, 353
category, 355
data file, 353
definition, 352
Mapping logic
test new, 361
Marketing, 32, 349
campaign, 32
plan/track activities, 366
work center, 353
Marketing attribute, 94
custom logic, 102
data assignment, 101
MARKETINGPRO, 95, 138, 353
Material hierarchy, 72
Material product, 70
Menu item, 402
Method, 245
check, 365
CREATE_BUSINESS_PARTNERS, 364
CREATE_BUSINESS_TRANSACTIONS, 351
ENHANCE_DATA_SOURCE, 386
EXPORT_CAMPAIGN_DATA, 369
maintain parameter, 360
signature, 360, 372
Method call, 278
action, 257
COMPLETE_DOCUMENT, 279
COPY_DOCUMENT, 279
create new, 279
standard, 279
Method call (Cont.)
TRIGGER_ALERT, 279
Microsoft Excel, 353
Mobility, 36
Modification operations, 415
Multiple clients, 44
Multiple ERPs, 37

N

Naming convention, 121, 306
Navigation bar profile, 402
add link, 404
New attribute set, 138
New field, 112
New transaction type, 84
Node
/CRMBW/ROOT, 395

O

Object, 156, 162, 264
attribute, 214
type, 120
Object function, 163
assign to object, 163
define, 166
Object/event structure, 162
Object-oriented code, 40
One order API, 86
One order framework, 77, 156
One order object layer API function modules, 314
Open channel, 365
campaign customization, 366
Opportunity Management, 30
Order Management, 31
Organization model
access rules, 340
Organizational data profile, 345, 346
Organizational model
custom rule, 342
Organizational unit attributes, 342
Outbound IDoc processing, 239
Outbound parameter, 241
Output execution, 246

P

Packet, 375
Parameter, 369
  EV_GUID_REF, 283
  I_DATASOURCE, 386
Partner data, 227
Partner definition, 209
Partner determination, 270
  access rules, 327
  access sequence, 332
  configuration, 328
  configuration and BAdI Implementation, 338
  options, 254
  procedure, 334
Partner function
  create new, 331
Partner profile, 241
Pass data to external applications, 398
Pay it forward, 420
People Centric User Interface, 42
Persistent class, 248
Physical file, 221
Planned callback buffer, 174
Post Processing Framework (PPF), 245
  customizing, 247
Potential customer lead, 350
Potential customer list, 350
Precise timing, 186
Primary table, 79
Printing options, 271
Priority, 317
Private method, 358
Process type, 84
Processing messages, 285
Processing Time option, 251
Processing Times Not Permitted option, 252
Processing type, 250
Product, 69, 220
  category, 147
  create new with function module, 77
  create with function module call, 233
  general data, 70
  hierarchy, 220
Product master
  API, 74
  attribute set, 144
PRODUCT_ID, 71
Profile
  assign to business transaction, 272
  create new, 268
Program
  RSPPFPROCESS, 272
Program calls, 409

Q

Qualification, 30

R

Rebate, 34
Receiver port, 240
Relationship category, 329
Relationship data, 56, 66
Report
  restrict, 274
Repository.xml file, 149
Required field, 294
RFC destination, 239
Rule
  attributes, 343
  use in organizational data profile, 345

S

Sales, 28
Sales area data, 60
Sales Force Automation, 28
Sandbox, 110
  system, 426
SAP Community Network, 416
SAP CRM
  common mistakes, 425
  development system, 425
  enhance the business logic, 293
  extract data, 396
  organizational model, 342
  report, 396
  single client recommendation, 425

SAP CRM (Cont.)
technical landscape, 38
SAP CRM ABAP code, 306
SAP CRM ABAP Java Virtual Machine Container, 41
SAP CRM ABAP Server, 39
SAP CRM Document Management system, 67
SAP CRM Java Server, 40
SAP CRM Middleware, 45, 176, 201, 211
set up for inbound processing, 207
site, 242
SAP CRM Web Client, 41
SAP GUI, 42, 396
SAP GUI for HTML, 397
set up, 397
SAP Mentor, 421
SAP NetWeaver ABAP Application Server, 64
SAP NetWeaver Application Server ABAP, 39, 397
SAP NetWeaver Application Server Java, 40
SAP NetWeaver Business Warehouse (BW), 379
data sources, 385
SAP R/3, 53
SAP Text and Retrieval Engine (TREX), 35
SAP user groups, 421
SAP Web Client → see Web Client
SAPscript ITF, 228
Segment, 54, 83, 226
signal end of processing, 174
Segment BAdI, 301, 313
implement logic, 302
Segment Builder, 33
Selection report, 269
Service, 31
Set tables, 80
Set type, 144, 147
assign to product category, 150
configure UI, 149
Signature, 359, 369
Smart form, 256, 271
Social call report, 295
Social CRM, 26, 27
Social media interaction, 295
Source site, 211
Source structure, 215
field, 216
map, 219
SQL statement, 92
Standard extractor, 391
Standard object subtype, 263
Standard profile, 247
Standard SAP-delivered code, 415
Step
create, 276
definition, 275
Structure
CRMT_MKLIST_ACT_EXT, 352
CRMT_MKTLIST_ACT_EXT, 351
CRMT_MKTLIST_LEA_EXT, 351
Subscription, 243
Subtype, 263
delegate, 264
use as run-time definition, 265
System sizing, 405
System upgrade, 413

T

Table
AUSP, 101
BUT000, 57
BUT050, 66
BUT051, 67
BUT100, 59
CABN, 96
COMC_SETTYPE, 73
CRMC_EVENT_STRUC, 162
CRMC_EVENTS, 161
CRMC_OBJ_FUNC, 163
CRMD_CUSTOMER_H, 387
CRMD_ORDER_INDEX, 92
display in report, 134
INDX, 185
INOB, 100
KLAH, 97
T000, 221
Table enhancements, 129
Table extension without code, 128

Tabulator, 221
Target group partners, 370
Third-party system, 37
Timing, 301
Trade Promotion Management, 33
Transaction
  AXTREG, 126, 138
  AXTSYS, 128
  calculate dates, 317
  CL03, 97
  CL24N, 101
  COMM_ATTR_SET, 145
  CRM_ORDER_MAINTAIN, 360
  CRMD_EVENT_TRACE, 181
  CRMM_UIU_PROD_CONFIG, 149
  CRMM_UIU_PROD_GEN, 148
  CRMV_EVENT, 157, 176, 180
  CRMXIF_C1, 243
  CT04, 96
  EEWB, 107
  LSMW, 205
  MW_MODE, 212
  RSA3, 388, 396
  RSA6, 381, 383, 394
  SE11, 352, 365, 370
  SE18, 280, 302, 385
  SE37, 103, 180, 204
  SE38, 87
  SE80, 188, 357, 397
  SICF, 397
  SM30, 239
  SM37, 274
  SM38, 278
  SMOEAC, 211, 242
  SMOGTOTAL, 211
  SPPCADM, 248
  SPPFCADM, 285
  SPRO, 294
  SWO1, 262
  WE20, 241
  WE21, 240
Transaction Launcher, 397
  class and entry, 402
  configure, 398
  Wizard, 399
Transactional data, 92
Transfer data, 381
Transfer status, 375
Transport request, 107, 126, 383
TREX, 41
Type organization, 141

U

User exit, 385

V

Variant, 272
  create, 274
View, 149
  configuration, 149
  V_TBDLS, 239
Virtual attribute, 385
VPN, 308

W

Warranty Management, 32
Web Channel, 34
Web Client, 93, 118, 307, 353, 397
  access marketing attributes, 95
  add new custom field, 129
  customizing, 404
  generate configuration, 148
  logical link, 398
  marketing attribute, 97
  marketing attributes, 138
  menu items, 402
Wizard, 112
Work center, 353, 402
Workflow Conditions option, 254

X

XIF adapter, 201
  extract data, 239
  function module, 203
  IDoc errors, 224
  load data, 225
  site, 242
XIF IDoc processing, 211

XIF module, 203
XIF outbound processing, 242
XIF remote-enabled function module, 202
XML code, 319

Z

Z-table, 370
Z-version, 357