Thomas Schneider

SAP® Business ByDesign Studio — Application Development

- Learn how to develop your own add-ons and custom-specific solutions to SAP Business ByDesign.
- Work with the scripting language to implement new business objects and design your own UIs.
- Discover how to sell and maintain your custom applications.
Contents at a Glance

1 Getting Started ................................................................. 37
2 Architecture Overview .................................................. 47
3 Business Object Modeling ............................................... 75
4 User Interface .............................................................. 131
5 Business Object Implementation ................................. 209
6 Business Configuration .................................................. 267
7 Business Object and User Interface Extensions .......... 285
8 Business Analytics .......................................................... 327
9 Service Integration .......................................................... 355
10 SAP Business ByDesign Platform Objects and Services .... 413
11 Lifecycle Management .................................................... 435
## Contents

**Foreword** ................................. 15  
**About This Book** ........................... 17  
**Introduction** ............................... 21  

### 1 Getting Started ....................... ...................... 37  
1.1 Introducing Site Management ................. 37  
1.2 Introducing SAP Business ByDesign Studio .................. 42  
1.3 The First Development Project ..................... 43  
1.4 Summary ........................................ 46  

### 2 Architecture Overview .................. ...................... 47  
2.1 Architecture Overview ........................... 47  
2.2 Repository ........................................ 54  
2.3 Business Object Engine ........................... 56  
2.4 User Interface Engine ............................ 60  
2.5 Search Engine .................................... 64  
2.6 Integration of Add-ons with SAP Business ByDesign ...... 66  
   2.6.1 Component Model ............................. 68  
   2.6.2 Public Model .................................. 71  
2.7 Summary ........................................ 73  

### 3 Business Object Modeling ............... ...................... 75  
3.1 Introduction to Business Objects .................. 76  
3.2 Defining Business Objects, Nodes, and Associations ...... 79  
   3.2.1 Defining Business Objects ..................... 79  
   3.2.2 Defining Business Object Nodes ................ 80  
   3.2.3 Defining Object Relationships ................ 81  
3.3 Defining Elements and Data Types ..................... 88  
   3.3.1 Core Data Type .............................. 89  
   3.3.2 Global Data Types ........................... 90  
   3.3.3 Important Data Types ........................ 92  
3.4 Adjusting Business Object Data for the User Interface and Other Consumers ..................... 98
Contents

3.5 Defining Actions ............................................................. 100
3.5.1 Defining the Status and Action Management ...... 101
3.5.2 Site Reservation: Status and Action Model ........ 103
3.5.3 Defining Actions that Do Not Belong to Status and Action Management ..................... 110
3.6 Defining Queries ............................................................. 111
3.7 Defining Messages .......................................................... 117
3.8 Example: Sales Order ...................................................... 118
3.9 Naming Conventions and Useful Guidelines .................... 122
3.9.1 Naming Conventions for Business Objects and Nodes ................................................................ 124
3.9.2 Naming Conventions for Business Object Node Elements ............................................................ 125
3.9.3 Naming Convention for Data Types ....................... 126
3.9.4 Naming Convention for Queries ......................... 127
3.9.5 Order of Elements in a Business Object Node ..... 127
3.9.6 Messages ........................................................... 127
3.10 Summary ........................................................................ 128

4 User Interface ...................................................................... 131
4.1 User Interface Fundamentals ........................................... 131
4.1.1 Floorplan Types .................................................. 131
4.1.2 Generation and Configuration of Screens ............ 133
4.1.3 UI Model View ................................................... 137
4.1.4 Data Model ........................................................ 140
4.1.5 Controller ........................................................... 141
4.2 Configuring a Form Pane ................................................. 143
4.2.1 Configuring Field Properties and Action Properties .......................................................... 143
4.2.2 Adding a User Interface Field or a Button to a Form .......................................................... 150
4.2.3 Adding and Removing Panes ......................... 153
4.2.4 Configuring the Identification Region .......... 154
4.3 Configuring a List Pane ................................................... 154
4.3.1 Configuring a Preview Pane ......................... 158
4.3.2 Defining Query and Default Set ..................... 159
4.3.3 Configuring Find Form and List Pane Variant ..... 164
4.4 Using Frontend Scripting ................................................ 165
   4.4.1 Implementing Dynamic Properties ...................... 166
   4.4.2 Filling Data Lists ............................................. 169
4.5 Configuring Object-Based Navigation ............................. 171
   4.5.1 Configuring a Create Scenario ......................... 174
   4.5.2 Configuring a Show Details Scenario .................. 175
   4.5.3 Configuring a Create with Reference Scenario .... 177
   4.5.4 OBN Troubleshooting ...................................... 181
4.6 Configuring Work Center and Work Center View .............. 182
4.7 Defining Authorizations ............................................ 185
   4.7.1 Authorization Fundamentals ............................. 185
   4.7.2 Defining an Authorization Check ...................... 188
   4.7.3 Defining an Instance Authorization Check .......... 190
   4.7.4 Defining Segregation of Duties ....................... 194
4.8 Configuring Embedded Components ............................... 195
4.9 Mobile User Interfaces .............................................. 201
4.10 Analyzing Issues in User Interfaces ............................ 202
4.11 Summary ..................................................................... 207

5 Business Object Implementation .................................... 209

   5.1 SAP Business ByDesign Scripting Language ............ 209
   5.1.1 Scripting Language Syntax Basics .................... 210
   5.1.2 Write Access to Business Objects .................... 219
   5.1.3 Read Access to Business Objects ..................... 226
   5.1.4 Messages ................................................... 232
   5.1.5 Functions .................................................. 233
   5.2 Implementing a Business Object—Fundamentals .......... 234
   5.2.1 Transaction ................................................ 234
   5.2.2 Locks ....................................................... 237
   5.3 Implementing a Business Object—Example ................ 239
   5.3.1 Event: AfterModify ..................................... 239
   5.3.2 Action: Check ............................................ 242
   5.3.3 Action: Copy ............................................. 244
   5.3.4 Status Action: Release ................................ 245
   5.3.5 Status Action: Start Sales Processing ............... 246
   5.3.6 Status Action: Confirm Sales Processing .......... 247
   5.3.7 Action: GetAvailableSites ............................ 249
### 7.5 Other User Interface Extensions ...................................... 320
#### 7.5.1 Renaming a Label or Hiding a UI Element ........ 320
#### 7.5.2 Adapting Work Centers ........................................ 321
#### 7.5.3 Adapting Work Center Views ............................. 322
### 7.6 Summary ........................................................................ 324

### 8 Business Analytics ......................................................... 327
#### 8.1 Business Analytics Fundamentals ............................ 327
#### 8.2 Defining Basic Data Sources ...................................... 332
#### 8.3 Defining Reports ...................................................... 336
##### 8.3.1 Defining a Report ........................................... 336
##### 8.3.2 Key Figures ................................................... 337
##### 8.3.3 Characteristics ................................................. 344
##### 8.3.4 Report Assignment ...................................... 345
#### 8.4 Adjusting Business Object Data for Analytics .......... 345
#### 8.5 Defining Joined and Combined Data Sources .......... 346
##### 8.5.1 Defining a Joined Data Source ....................... 347
##### 8.5.2 Defining a Combined Data Source (Union) ...... 350
#### 8.6 Adding Extension Fields to Data Sources and Reports ........................................ 351
#### 8.7 Summary .................................................................... 352

### 9 Service Integration .......................................................... 355
#### 9.1 Output Management .................................................. 358
##### 9.1.1 Output Management Fundamentals ............... 358
##### 9.1.2 Installing Adobe LiveCycle Designer .............. 360
##### 9.1.3 Creating a Print Form for a New Business Object ........................................ 361
##### 9.1.4 Adopting Existing Form Templates .................... 366
##### 9.1.5 Adding Extension Fields to Existing Form Templates ........................................ 368
##### 9.1.6 Extending an Existing Form Data Type .......... 369
#### 9.2 Business Task Management ....................................... 370
##### 9.2.1 Business Task Management Fundamentals ........ 370
##### 9.2.2 Implementing an Approval Scenario .................. 372
#### 9.3 Internal Communication ............................................. 378
##### 9.3.1 Internal Communication Fundamentals .......... 378
9.3.2 Defining the Interaction Structure and Mapping ...................................................... 382
9.3.3 Implementing the Process Integration Choreography .............................................. 388
9.3.4 Defining the Conflict and Error Handling and Monitoring ....................................... 394
9.3.5 Adding Extension Fields to an Existing Process Integration ........................................ 397
9.4 External Web Services .................................................................................................. 398
  9.4.1 Defining an External Web Service .............................................................................. 399
  9.4.2 Defining Authorizations for a Web Service ............................................................. 400
  9.4.3 Using Web Services .................................................................................................. 403
  9.4.4 Implementing Conflict and Error Handling ............................................................... 407
9.5 File Input ....................................................................................................................... 409
9.6 Summary ......................................................................................................................... 410

10 SAP Business ByDesign Platform Objects and Services .................................................. 413
  10.1 SAP Business ByDesign Platform Overview ............................................................. 413
  10.2 Business Partner, Organizational Unit, Party, and Identity ......................................... 417
      10.2.1 Business Partner .................................................................................................. 417
      10.2.2 Organizational Unit ............................................................................................. 419
      10.2.3 Party .................................................................................................................. 422
      10.2.4 Employee and Manager Assignment .................................................................... 422
      10.2.5 Identity ................................................................................................................ 424
      10.2.6 Summary .............................................................................................................. 424
  10.3 Reuse Services for Address, Text, and Attachment ....................................................... 425
      10.3.1 Managing Addresses ......................................................................................... 425
      10.3.2 Managing Texts ................................................................................................. 428
      10.3.3 Managing Attachments ...................................................................................... 430
  10.4 Cost Objects ............................................................................................................... 431
  10.5 Summary ....................................................................................................................... 433
11 Lifecycle Management .................................................. 435

11.1 Preparation Phase .................................................. 436
    11.1.1 Getting Detailed Information on SAP Business ByDesign and SAP Business ByDesign Studio .................................................. 436
    11.1.2 Deciding on the Solution Type ......................... 438
    11.1.3 Designing a Solution ....................................... 440
    11.1.4 Installing SAP Business ByDesign Studio .......... 441
    11.1.5 Creating Users and Test Data ......................... 442

11.2 Development Phase .............................................. 445
    11.2.1 Creating a Solution ....................................... 445
    11.2.2 Getting Support .......................................... 445

11.3 Deployment Phase ................................................. 447
    11.3.1 Testing the Solution (Test Deployment) ........... 449
    11.3.2 Quality Review ........................................... 450
    11.3.3 Listing the Solution in the SAP Store ............. 451
    11.3.4 Deploying Customer-Specific Solutions .......... 453

11.4 Operation and Support Phase ............................... 454
    11.4.1 Purchasing and Activating a Solution ............. 454
    11.4.2 Working with Customer Incidents .................. 455

11.5 Continuous Improvement Phase ............................ 456

11.6 Summary ............................................................. 458

Appendices ........................................................................ 461

A Information Sources ................................................ 461
B Glossary ..................................................................... 469

Index ............................................................................. 475
5.2 Implementing a Business Object—Fundamentals

In a multiuser IT system, many users can simultaneously read the contents of business objects. However, for changes to the dataset, you must ensure that only one user at a time can change a particular object’s content consistently. For this purpose, content is managed in a transaction. Transactions process data compliant with the ACID principles (atomicity, consistency, isolation, and durability) over multiple user interaction steps.

5.2.1 Transaction

A transaction consists of two phases: During the interaction phase, data is retrieved and entered, and actions are executed on the business object. In the save phase, the system performs last determinations and checks and writes the changed data to the database.

Interaction phase

Figure 5.1 shows the process flow of user activities that are triggered in the interaction phase. If the UI triggers a query, the data is read directly from the database and search engine. For a retrieve operation, the business object buffer is checked first, before data is retrieved from the database and search engine.
If data is changed via an action triggered by the UI, the business object framework acquires a lock for the business object instance, and the AfterModify and/or the action coding is processed. The AfterModify event is processed after an action if the coding of the action changed data of the business object. If data is changed on the UI and an action is triggered, the AfterModify event is called before and after the action coding.

![Sequence Diagram for a Transaction, Part 1: Interaction Phase](image)

Figure 5.1 Sequence Diagram for a Transaction, Part 1: Interaction Phase

Figure 5.2 shows the save phase of a transaction’s lifecycle. In the Before-Save event, business objects can still call other services that may change the state.

The CheckBeforeSave event performs all checks required to ensure data consistency from the business point of view. At this point in the save phase, to ensure a consistent view for all checks, no further changes are allowed. The service provider can either accept or reject the request. The result of CheckBeforeSave is that either all changes are released, which
allows them to be written to the database, or the save process is interrupted. If any check is unsuccessful, the entire save process is stopped and error handling is started. However, an error does not abort the transaction. The business object framework sets the transaction back into a state in which user interaction can be accepted once again.

We recommend using the option to stop the save sequence with care and only in cases of a severe inconsistency. Instead, a business object should be saved even in an inconsistent, draft, or “prima nota” status.

---

**Figure 5.2** Sequence Diagram for a Transaction, Part 2: Save Phase

**Events**

The business object framework provides the following events that can be implemented with script coding:

- **Determination event AfterModify**
  
  This event is executed whenever a consumer call or an action modifies a business object, in particular, if a business object node is created, updated, or deleted. In this event determination logic is imple-
mented, for example, the calculation of default values and consistency checks.

- **Action event**
  For each application-specific action an event is provided for the implementation of action-specific validations (action precondition) and result determination.

- **Determination event BeforeSave**
  This event is executed whenever a consumer calls the save operation of a business object. In this event determination logic can be implemented before the business object is saved on the database. In this event modifications to the own business object and to other business objects are possible.

- **Validation event CheckBeforeSave**
  This event is executed as part of the save sequence after the BeforeSave determination event and before the business objects are saved on the database. In this event validation logic can be processed and the save sequence can be stopped and messages can be raised. The CheckBeforeSave event is not released for business objects created with SAP Business ByDesign studio in version 3.0.

It is possible to enable script files for mass-processing. For mass-enabled script files the `this` operator represents a collection of business object node references. The decision about if a script file will be mass-enabled has to be made when the script file is created. To create a mass-enabled script file the mass-enabled indicator has to be set in the script file creation dialog. This indicator can only be set during the creation of the script file; later on this decision cannot be changed.

### 5.2.2 Locks

**Locks** are required to ensure that only one user at a time can change a particular objects’ content consistently.

To book a vacation, you must check that all necessary components are available: flights, hotels, bus or boat transfers, and so on. The “all or nothing” principle applies: If no flights are available, you will not need a hotel room and so on. Since you usually check the availability of the various components one after the other, you want to be certain that
another user doesn’t change any of the items in the sequence before the entire booking is completed.

**Database locks**
When we are talking about transactions and locks in this section, we refer to transactions and locks that are managed by the business object engine of SAP Business ByDesign. The database system also manages transactions and locks, but these are only used during a very short time frame when the business object engine saves the business object changes on the database in the save phase. Database locks are never used to lock objects through several user interaction steps.

**Application locks**
To lock business object instances through several user transaction steps, the business object engine of SAP Business ByDesign manages locks using a central lock server (also called enqueue server) that serves lock requests for all application servers.

**Rules for transaction and lock handling**

- Queries are always directly served by the database and search engine. Changes that are made on the business object buffer are not reflected in the result of a query. This rule applies not only for different transactions, but also within a transaction.

- Object-based navigation opens a new transaction. Data that has not been saved on the database is not visible in the new transaction. In the site reservation example, a sales order is created when the site reservation is released. If you open the sales order link without saving the site reservation first, the newly created sales order is not found; the sales order UI will not terminate but will be empty. You can avoid this behavior, which users might recognize as an error: Using frontend scripting in the UI allows you to dynamically deactivate the link if the site reservation is not saved.

- The lock behavior is determined by the UI:
  - The default lock behavior for UIs is the optimistic locking. Objects are locked as soon as the first round-trip with changed data is processed. Opening an object in edit mode does not lock an object. If you open a site reservation by clicking **Edit** on the object work list, it is not yet locked. If you or a second user opens the same reservation and changes data, the lock will be granted to the second transaction. If you as the first user, having opened the site
reservation without changing it, try to change the object now, the lock handler raises an error.

- You can also set the lock behavior in the UI (property LockBehavior of a data list) to “exclusive.” In this case, the lock will be set as soon as the object is read the first time.
- If the lock behavior in the UI is set to “read-only,” no change locks are set and therefore no change operations can be performed.
- Locks are set on the object level. If a site reservation is locked, you cannot add a new instance of the subnode Additional Services from another transaction. There are some exceptions to this rule for SAP-delivered business objects.

5.3 **Implementing a Business Object—Example**

This section explains how to implement the event coding for a business object for the site management example.

5.3.1 **Event: AfterModify**

Let’s have a look at the AfterModify event of the Site Reservation business object:

```java
import ABSL;
// initialization sequence
if (this.ReleaseStatusCode.IsInitial()) {
    this.ReleaseStatusCode = "1";
    this.SalesProcessingStatusCode = "1";
    if (this.OriginSiteReservationID.IsInitial()) {
        // (1) normal creation (no copy)
        this.ArrivalDate = Context.GetCurrentUserDate();
        this.DepartureDate = this.ArrivalDate.AddDuration(Duration.ParseFromString("P1D"));
    } else {
        // (2) Copy
        this.Copy();
    }
}
```
// (3) call check action
this.Check();
// (4) update sales order
if ( this.ConsistencyStatusCode == "3" &&
    this.SalesProcessingStatusCode == "2" ||
    this.SalesProcessingStatusCode == "5" )
{
    this. MaintainSalesOrder();
}
// (5) set name
this.Name = "Reservation: ".
    Concatenate(this.ArrivalDate.ToString());
if ( this.ToAccount.IsSet() )
{
    this.Name = this.Name.Concatenate(this.ToAccount.
        CurrentCommon.BusinessPartnerFormattedName);
}
// (6) set UI status
switch (this.ReleaseStatusCode) {
    case "1" {
        switch (this.ConsistencyStatusCode) {
            case "2" {this.SiteResStatusCode.content = "1"; }
            case "3" {this.SiteResStatusCode.content = "2"; }
            default {}
        }
    }
    case "3" {
        < ... >
    }
    case "5" {this.SiteResStatusCode.content = "9"; }
    default {}
}

Listing 5.8 Implementation of the AfterModify Event (Excerpt)

The coding starts with the import of the namespaces of the SAP Business
ByDesign Scripting Language tools library (import ABSL).

The next coding part is the initialization sequence. It starts with the
statement if (this.ReleaseStatusCode.IsInitial()) and continues
with statements for setting the default values for the status variables.

Copy sequence In Section 4.5.3, "Configuring a Create With Reference Scenario," we
describe the implementation the UI part of the copy use case. For a
copy, the element OriginSiteReservationID is filled. The statement if (
this.OriginSiteReservationID.IsInitial() checks if OriginSiteReservationID is filled. If this is the case, the coding continues with the else statement, followed by the execution of the Copy action in the Site Reservation business object ②.

If OriginSiteReservationID is not filled, the coding continues with the regular initialization sequence ① and sets default values for arrival and departure date.

After the initialization sequence, the coding calls the Check action ③. It is assumed that every business object has a basic set of checks. During modify, all “local” checks are performed on the nodes that are changed or change as a side effect. However, checks that are performance critical are usually not performed in this event. These checks are usually performed in a separate Check action, for example, before an object is released. In our example, we do not distinguish between performance-critical and non-critical checks. The Check action is executed in every AfterModify event.

If the site reservation is consistent and the sales process is in process or interrupted, the sales order is updated ④.

The next coding block ⑤ sets the name of the site reservation. The last part of the coding ⑥ determines the SiteReservationStatus (also introduced as UI status in Section 3.5.2; see also Table 3.1).

### Before Image

The implementation of the AfterModify event as presented in Listing 5.8 has two disadvantages:

- “Expensive” activities, such as the Check action and the update of the sales order (MaintainSalesOrder action) are executed even if the relevant input data did not change. This may lead to a performance impact.
- The business object itself cannot reject input data, such as a change of the arrival and departure date after the confirmation of the reservation.

To overcome these disadvantages, you may implement a before image of the site reservation. In the current version of SAP Business ByDesign studio, this is not a supported feature. Therefore, you have to implement it using the following workaround:
5.3.2 Action: Check

The following listing shows the implementation of the Check action.

```javascript
import ABSL;
import AP.FO.BusinessPartner.Global;
var qySiteRes = SiteReservation.QueryByElements;
var selParSiteRes = qySiteRes.CreateSelectionParams();
var found;
var customer;
// (1) set default: consistency status = Consistent (3)
this.ConsistencyStatusCode = "3";
// (2) check date
if ( this.ArrivalDate.IsInitial() ||
    this.ArrivalDate.IsInitial() ||
    this.ArrivalDate.GreaterEquals(this.DepartureDate) ) {
    this.ConsistencyStatusCode = "2";
    raise NoValidPeriod.Create("E");
}
// (3) check site
if ( !this.ToSite.IsSet() ) {
    this.ToSite.Reset();
    this.ConsistencyStatusCode = "2";
    raise NoValidSite.Create("E");
}
// (4) map & check account
if ( !this.MaintenanceIndicator ) {
```
found = false;
if (this.AccountID != "") {
    customer = Customer.Retrieve(this.AccountID);
    if (customer.IsSet()) {
        this.ToAccount = customer; found = true;
    } else {
        this.ToAccount.Reset(); found = false;
    }
} else {
    this.ToAccount.Reset();
}
if (this.AccountID == "" || !found) {
    this.ConsistencyStatusCode = "2";
    raise NoValidAccount.Create("E");
    this.AccountID = "";
}
// (5) check site availability
if (this.ConsistencyStatusCode == "3") {
    selParSiteRes.Add(qySiteRes.ToSite, this.ToSite);
    selParSiteRes.Add(qySiteRes.ReleaseStatusCode, "I", "EQ", "3");
    selParSiteRes.Add(qySiteRes.ArrivalDate, "E", "GE", this.DepartureDate);
    selParSiteRes.Add(qySiteRes.DepartureDate, "E", "LE", this.ArrivalDate);
    selParSiteRes.Add(qySiteRes.ID, "E", "EQ", this.ID);
    if (qySiteRes.Execute(selParSiteRes).Count() > 0) {
        this.ConsistencyStatusCode = "2";
        raise SiteAlreadyReserved.Create("E");
    }
}

Listing 5.9 Implementation of the Check Action (Excerpt)

The coding starts with imports of the namespaces of the SAP Business
ByDesign Scripting Language tools library (import ABSL) and of the SAP
business partner and identity management. The next lines are defini-
tions of the required variables.
In section 1, the consistency status is set to "Consistent (3)." If inconsistent behavior of the business object instance is detected by the subsequent coding, the value will change to "Inconsistent (2)."

Check of arrival and departure data
The next part of the coding 2 checks if arrival and departure data are entered correctly.

Site ID check
The following coding block 3 is the ID check for the site. The business object Site exposes a human-readable ID as an alternative key. This means the association to the site can be set directly by the consumer, by exposing the element SAP_ToSite as an input field on the UI. The validation of the input is done by the statement if (this.ToSite.IsSet()). This statement checks if a site with the ID that has been specified in the input field really exists. If not, the association ToSite is cleared and a message is raised.

Account ID check
The next coding block 4 is the ID mapping and check for the account. Please look at the differences between this coding block and the previous check for the site: The human-readable ID of the Customer business object (InternalID) is not exposed as a stable alternative key, so it cannot be used to define the association directly. Instead, a separate element (AccountID) is used as an input field on the UI that contains the human-readable ID of the account. In the coding the input needs to be verified. For this purpose, a Retrieve is performed. If a result is found, the association to the respective instance of the customer business object is set. If no matching customer is found, the element AccountID is cleared and a message is raised.

Site Availability Check
The last coding block 5 checks if a selected site is actually free. The coding is very similar to the coding used in the action GetAvailableSites. We will explain this coding in the section on this action below (see Section 5.3.5, "Status Action: Start Sales Processing").

5.3.3 Action: Copy
The copy action performs a Retrieve operation to read the origin site reservation and copies the account ID and arrival date and time to the new site reservation:

```javascript
var originSiteRes = 
    SiteReservation.Retrieve(this.OriginSiteReservationID);
```
if (originSiteRes.IsSet()) {
    this.AccountID = originSiteRes.AccountID;
    this.ArrivalDate = originSiteRes.ArrivalDate;
    this.DepartureDate = originSiteRes.DepartureDate;
}
else {
    // code block: error handling
    <...>
}

Listing 5.10 Implementation of the Copy Action (Excerpt)

5.3.4 Status Action: Release

The implementation of an action consist of two parts: The first part checks the preconditions of the action. If the preconditions are not met, the second part, which contains the execution of the action, is skipped.

The implementation of the Release action of the Site Reservation business object is very simple. It looks like:

```javascript
var preconditionOkay = false;
// (1) check precondition
if ( this.ConsistencyStatusCode == "3" &&
    this.ReleaseStatusCode == "1" ){
    preconditionOkay = true;
}
if ( preconditionOkay ) {
    // (2) execute action
    this.ReleaseStatusCode = "3"; // Released
}
else {
    // (3) raise messages
    if ( this.ConsistencyStatusCode != "3" ) {
        this.Check();
    }
    if ( this.ReleaseStatusCode != "1" ) {
        raise ReservationAlreadyReleased.Create("E");
    }
}
```

Listing 5.11 Implementation of the Release Action
The precondition check ① evaluates the values of the consistency and the release status as defined in the status and action model (Chapter 3, Figure 3.5). In the second (execution) part ②, the release status is set to “Released (3).” If the precondition is not fulfilled, appropriate error messages have to be raised. If the site reservation is not consistent, the easiest solution is to execute the Check action, which will raise the appropriate error message. If the release status is not “Released,” an appropriate message has to be defined.

5.3.5 Status Action: Start Sales Processing

The following listing shows the coding of the StartSalesProcessing action:

```javascript
var preconditionOkay = false;
// (1) check precondition
if ( this.SalesProcessingStatusCode == "1" &&
    this.ReleaseStatusCode == "3" ){
    preconditionOkay = true;
}
if ( preconditionOkay ) {
    // (2) execute action
    if ( this.MaintenanceIndicator == true ) {
        this.SalesProcessingStatusCode = "4"; // Not relevant
    }
    else {
        this.MaintainSalesOrder();
        if ( !this.ToSalesOrder.IsSet() ) {
            this.SalesProcessingStatusCode = "5"; //Interrupted
        }
        else {
            this.ToSalesOrder.Release();
            if ( this.ToSalesOrder.Status.ConsistencyStatusCode
                 != "3" ||
                 != "3" ) {
                this.SalesProcessingStatusCode = "5";
                        //Interrupted
            }
        }
    }
}
```
Implementing a Business Object—Example

5.3

this.SalesProcessingStatusCode = "2";
// In process
}
}
}
else {
    // (3) raise messages
    < ...>
}

Listing 5.12 Implementation of StartSalesProcessing Action

The action evaluates the preconditions as defined in the status and action model in Chapter 3, Figure 3.5.

If the preconditions are fulfilled, the action implementation checks if the reservation is made for site maintenance. If yes, it sets the sales processing status to "Not Relevant." No sales order is created in this case. If no, a sales order is created or updated by calling the MaintainSalesOrder action and released by calling the Release action on the Sales Order. If the creation and release is successful, the sales processing status is set to "In Process." If an error occurs, the status goes to "Interrupted."

If the preconditions are not fulfilled, error messages must be provided.

5.3.6 Status Action: Confirm Sales Processing

The following listing shows the ConfirmSalesProcessing action:

var preconditionOkay = false;
// (1) check precondition
if ( ( this.SalesProcessingStatusCode == "2" ||
      this.SalesProcessingStatusCode == "5" ) &&
      this.ReleaseStatusCode == "3"){
    preconditionOkay = true;
}
if ( preconditionOkay ) {
    // (2) execute action
    if ( !this.ToSalesOrder.IsSet() ) {
        this.SalesProcessingStatusCode = "5"; // Interrupted
    }
else {
    this.MaintainSalesOrder();
    this.ToSalesOrder.
        FinishFulfilmentProcessingOfAllItems();
    if ( this.ToSalesOrder.Status.ConsistencyStatusCode != "3" ||
        this.ToSalesOrder.Status.
            ItemListFulfilmentProcessingStatusCode != "3" ) {
        this.SalesProcessingStatusCode = "5":
            // Interrupted
    }
    else {
        this.SalesProcessingStatusCode = "3": // Finished
    }
}
if ( this.SalesProcessingStatusCode == "5" ) {
    raise SalesOrderCreationError.Create("E");
}
else {
    // (3) raise messages
    < ...>
}

Listing 5.13 Implementation of ConfirmSalesProcessing Action

The action implementation updates the sales order by calling the action
MaintainSalesOrder and calls the action FinishFulfilmentProcessing-
OfAllItems on the Sales Order. This action sets the fulfillment process-
ing status of all items in the sales order to “Finished.” If the processing
was successful, the sales processing status of the site reservation is set to
“Finished,” too. If an error occurs, the status is set to “Interrupted.”

In Section 3.5, “Defining Actions,” we have already differentiated
between requiring preconditions, enabling preconditions, and inhibit-
ing preconditions. The different preconditions can be translated into
coding using the following pattern:

var preconditionOkay = false;
// check precondition
if ( < requiring precondition 1 > &&
    < requiring precondition 2 > &&
    ...
    < requiring precondition n > &&
Listing 5.14 Implementation Pattern for Prerequisites

5.3.7 Action: GetAvailableSites

The actions GetAvailableSites, Pick, and MaintainSalesOrder are not related to the status and action model.

The action GetAvailableSites fills the node Available Site based on the arrival and departure date specified in the site reservation:

```javascript
import ABSL;
var allSites;
var reservations;
var newSite;
var qySiteRes = SiteReservation.QueryByElements;
var selParSiteRes = qySiteRes.CreateSelectionParams();
var found;

// (1) check date
if ( this.ArrivalDate.IsInitial() ||
    this.DepartureDate.IsInitial() ||
    this.ArrivalDate.GreaterEquals(this.DepartureDate) ) {
    raise NoValidPeriod.Create("E");
} else {

    // (2) delete AvailableSites nodes
    foreach ( site in this.AvailableSites ) {
        site.Delete();
    }

    // (3) get all sites
    allSites = Site.QueryByElements.Execute();

    // (4) get all reservations in booking period
    selParSiteRes.Add(qySiteRes.ReleaseStatusCode, "I", "EQ", "3");
}
```
Listing 5.15 Implementation of GetAvailableSites Action (Excerpt)

After checking the validity of the date period, the next coding block deletes the instances of the Available Site node.

The availability check follows. The first step of the availability check reads all sites into the variable allSites. The next step checks for released site reservations in the time period between the arrival and the departure date. If no reservation is found, the site is free and is added to the node Available Site.

### 5.3.8 Action: Pick

The Pick action, which is defined on the Available Site node, ensures the transfer of the ID of a selected site from the list of available sites to the root node of the site reservation. It consists of exactly one line:

```plaintext
this.ToParent.ToSite = this.ToAvailableSite;
```
This statement makes use of the `ToParent` association, which is implicitly defined on every subordinate node. It transfers the site that is selected from the Available Site node to the association `ToSite` on the root level of the Site Reservation.

### 5.3.9 Action: MaintainSalesOrder

The action `MaintainSalesOrder` is used to create or modify a sales order based on the data of the site reservation. The code consists of two branches: The first branch creates a sales order. The second branch is executed if a sales order is already associated; it ensures the update of the sales order.

```plaintext
import ABSL;
import AP.CRM.Global;
var elSO : elementsof SalesOrder;
var newSO;
var newSOItem;
var elSOItem : elementsof SalesOrder.Item;
var quantity;
// (1) check if consistent
if ( this.ConsistencyStatusCode == "3" &&
    ( this.SalesProcessingStatusCode == "1" ||
      this.SalesProcessingStatusCode == "2" ||
      this.SalesProcessingStatusCode == "5" ) &&
    !this.MaintenanceIndicator ) {
    // (2) check service product
    if ( this.ToSite.ToSiteCategory.IsSet() ) {
        if ( this.ToSite.ToSiteCategory.
            ToDefaultServiceProduct.IsSet() ){
            // (3) check if SO is already created
            if ( !this.ToSalesOrder.IsSet() ) {
                // (4) create sales order
                elSO.Name.content = this.Name;
                newSO = SalesOrder.Create(elSO);
                if ( newSO.IsSet() ) {
                    this.ToSalesOrder = newSO;
                    raise SalesOrderCreated.Create("S",
                        this.ToSalesOrder.ID);
                    // (5) party node is created automatically
                    // – maintain buyer party (account)
```
newSO.BuyerParty.PartyKey.PartyID.content = this.AccountID;

// (6) create item node, enter item product
e1SOItem.Description.content = "MAIN";
newSOItem = this.ToSalesOrder.Item.Create(e1SOItem);
newSOItem.ItemProduct.ProductKey.ProductID.content =
   this.ToSite.ToSiteCategory.
   ToDefaultServiceProduct.InternalID.content;

// (7) enter quantity
if (newSOItem.
    FirstRequestedItemScheduleLine.IsSet()) {
    quantity = Date.Delta(this.DepartureDate,
    this.ArrivalDate).ConvertToDays();
    newSOItem.FirstRequestedItemScheduleLine.
    Quantity.content = quantity;
}

// (8) enter service duration
if (newSOItem.ItemServiceTerms.IsSet()){
    newSOItem.ItemServiceTerms.
    ServicePlannedDuration = Duration.
    ParseFromString((quantity * 24).ToString());
}

}
}

else {
// (9) update existing sales order
<...>
}

// (10) check the sales order's consistency
if (this.ToSalesOrder.Status.ConsistencyStatusCode
    != "3") {
   raise SalesOrderCreationError.Create("E");
}
}

else {
// (11) raise error msg: no service product / resource
}
}

else {
// (12) raise error msg: site reservation not consistent
}

Listing 5.16 Implementation of the MaintainSalesOrder Action (Excerpt)
The initial if clause 1 checks the consistency and the sales processing status of the site reservation. If the object is not consistent or the sales processing is already completed, the remaining coding is not processed. It also checks 2 that the category of the selected site is set and has a service product and a resource assigned, which is required to fill the sales order, and that the site reservation is not created for maintenance purposes.

### Initial checks

The statement if (newSalesOrder.IsSet()) 3 opens the two branches (code blocks 4 and 9) for creation and update of the sales order. The creation of the sales order consists of five steps:

1. **Creation of the sales order** 4
   Before creating the sales order, we fill the structure eISO to hand over the name of the sales order in the Create operation.

2. **Enter account** 5
   On creation of the sales order, the service provider class of the Sales Order automatically creates subnodes, for example, the Party node. The account, which is called Buyer in the Sales Order, is entered into the Party node using the association BuyerParty (see also Section 3.8, “Example: Sales Order”).

3. **Create the Item node and enter service product** 6
   On creation of the Item node, the subnode Item Product is automatically created. The service product that is assigned to the site category of the selected site is entered into this node.

4. **Enter quantity** 7
   On creation of the Item node, the subnode Item Schedule Line is created. The association FirstRequestedItemScheduleLine is used to fill the quantity, which is calculated from the booking period.

---

**Note**

Please note that for a statement with chained associations (<...>.ToSite.ToSiteCategory.ToDefaultServiceProduct.<...>) you have to check the existence of the association at runtime on each level. If one of the associations is not set, the program will terminate. The IsSet method checks the existence of only the last association in the chain.

---

**Five steps of Sales Order creation**
5. **Enter service duration**

On creation of the Item node, the subnode Item Service Terms is automatically created. The service duration is entered into this node.

**Note**

You can see from this example that the business object service provider automatically creates subnode instances on creation of a super-ordinate node instance, in our example, the Party, the Item Product and the Item Schedule Line node. In this case you must not create these node instances with a Create operation—this would lead to a corrupt object. On the other hand, you need to check if the expected instance is really created using the IsSet operation on the association. In case of an error, the service provider may not create the subnode instance automatically. Accessing a nonexistent node instance from your coding will lead to a program termination.

**Sales Order update**

The update branch, which is not shown in the listing, consists of three steps:

1. **Synchronization of the account (buyer)**
   If the account changes in the site reservation, the buyer is changed in the sales order.

2. **Synchronization of the service product**
   If the site category and the related service product change in the site reservation, the service product is changed in the sales order.

3. **Synchronization of the quantity**
   If the booking period changes in the site reservation, the quantity is changed in the sales order.

Finally the action checks the consistency of the sales order. If the sales order is not consistent, the sales processing status is set to error status 5 (Interrupted) and messages are raised.

**Master Data Maintenance**

The creation and processing of a sales order requires only very few input data (account, product, and quantity). However, some master data are a prerequisite for the successful creation and processing of a sales order.

For the account, maintain the following data:
5.3.10 Event: BeforeSave

The BeforeSave event is processed as part of the save sequence that is triggered whenever a consumer of a business object calls the save operation. It allows modifications to business objects before the data are persisted on the database.

```plaintext
import ABSL;
import AP.PC.IdentityManagement.Global;
var identity;
var datetime;
// (1) delete AvailableSites nodes
foreach ( site in this.AvailableSites ) {
    site.Delete();
}
identity = Identity.Retrieve(Context.GetCurrentIdentityUUID());
datetime = Context.GetCurrentSystemDateTime().ConvertToGlobalDateTime();
```

For the service product, maintain the following data:

- Communication: Address (minimum required: country, zip code)
- Relationship: Employee Responsible
- Sales Data: Sales Organization with distribution channel, incoterms (e.g., Free on board), payment terms (e.g., Payable immediately due net), currency
- Financial Data: Company and Account Determination Group (Domestic, third party)

The booking period must be open (see work center General Ledger, function OPEN AND CLOSE BOOKING PERIODS).

The action FinishFulfilmentProcessingOfAllItems confirms the service delivery and starts the invoice processing. The sales order appears in the INVOICE REQUESTS view of the Customer Invoicing work center. A background job will automatically create an invoice. Alternatively, you can trigger the invoice in the INVOICE REQUESTS view manually.
if ( this.CreationDateTime.IsInitial() ) {
// (2) Creation date/time, user
this.CreationDateTime = datetime;
if ( identity.IsSet() ) {
    this.CreationIdentity = identity;
}
} else {
// (3) Last changed date, time, user
this.LastChangedDateTime = datetime;
if ( identity.IsSet() ) {
    this.ToLastChangedIdentity = identity;
}
}
if ( !identity.IsSet() ) {
    // error handling
}

Listing 5.17 Implementation of the BeforeSave Event (Excerpt)

The first code block 1 deletes the instances of the transient Available Site node.

The second and third coding blocks set the information about the creation date/time and user 2 or about the last changed date/time and user 3.

5.4 Recording and Evaluating a Trace

SAP Business ByDesign Scripting Language is supported by a trace that can help to analyze unexpected behavior in three situations:

- **Developer trace**
  You can start the trace as developer directly in SAP Business ByDesign studio for your development user.

- **Trace for a test user**
  You can start the trace as developer in SAP Business ByDesign studio for a test user (business user) who is logged on to the SAP Business ByDesign UI.

- **Incident trace**
  In a production system, a user can start a trace and attach it to an incident and send the incident to SAP. SAP will forward the trace file to
the add-on developer, who can analyze the trace file in SAP Business ByDesign studio.

The trace will help you to answer the following questions:

- **Has a particular script been executed at all, and have the scripts been executed in the expected order?**
  The trace automatically logs the execution of each script. This may be helpful, especially in case of nested action calls and when scripts on different nodes are involved in a round-trip.

- **What was the instance a script was executed on?**
  The trace logs the identifier (UUID) or—if available—the alternative key of the node that is represented by the *this* pointer in a script.

- **Which messages have been raised during execution?**
  Every message that is raised by a business object operation that is called during processing is logged. This is especially important when working with SAP business objects.

- **What was the result of a retrieve operation?**
  The trace logs the UUID or—if available—the alternative key of the result of a retrieve operation.

- **Which associations has the script accessed?**
  For every association that is accessed, the trace logs the identifier (UUID) or—if available—the alternative key of the accessed node.

- **What was the result of a query?**
  The number of records that were returned by a query execution is logged in the trace. This information gives you a hint that something might have gone wrong if you expect a result from the query and nothing was returned or if you expected exactly one result record, but many have been delivered.

All this information is automatically logged if the trace is started. But you can also add specific information, for example, the information that a certain branch in your coding is executed, or you can report the value of a variable. For this purpose, you can use the Trace class of the ABSL library:

```plaintext
import ABSL;
Trace.Error(“Creation failed”);
```
Trace.Error("Creation failed:", this.ID);
Trace.Info("Initialization of status elements");
Trace.Info("Amount =", this.Amount.Content.ToString());

The Trace class has two methods: the Error method reports an error to the trace; the Info method indicates information. In brackets, you can provide one or two arguments of type String.

### Data Volume of a Trace

Tracing can record a potentially large amount of information; therefore, we recommend activating traces for limited periods of time. Otherwise, the amount of data recorded by the trace may become unmanageable. Traces are automatically deactivated if they are left running.

#### Tutorial: Creating and analyzing a trace

To create a trace for your development user and to analyze the trace, proceed as follows:

1. Enable the Trace Explorer.
   - In the SAP Business ByDesign studio menu, select View • Toolbars • Trace Explorer. The Trace Explorer toolbar appears.
2. Activate the trace.
   - In the Trace Explorer toolbar, click Activate Trace. Execute the UI that you want to test.
3. Deactivate the trace.
   - In the Trace Explorer toolbar, click Deactivate Trace.
4. Open the Trace Explorer window.
   - In the Trace Explorer toolbar, click Show Trace Explorer. The Trace Explorer appears in the side window (see Figure 5.3).
5. Open a trace.
   - In the Trace Explorer window, navigate to the Local Traces tab, and click Refresh Traces. The list of traces that are available in the tenant is displayed, identified by the time the trace was started and the ID of the user executing the UI (see Figure 5.3). Select a trace and click Get Trace Details.
6. Analyze the sequence of events.
   - In the lower window of the Trace Explorer, labeled as Sequence Explorer in Figure 5.3, the sequence of the traversed events is displayed in a hierarchical tree. Figure 5.3 shows the sequence of the
Release action. In the first step of the sequence, the AfterModify event of the root node of the SiteReservation business object is triggered, because data has been modified in the UI. The second step is the StartSalesProcessing action that calls the MaintainSalesOrder action. Finally the AfterModify event is called again. A red alert icon indicates that an error occurred in the action.

7. Analyze the details of an event.

Click an event. The trace records and messages that have been logged during the execution of the event are displayed in the SEQUENCE DETAILS window. This window has two tabs, the TRACE DETAILS tab that shows the trace records and the MESSAGES tab that shows the messages. Figure 5.3 shows the trace details of the Check action (see Listing 5.9 in Section 5.3.2). The trace for this event has four entries. The first entry indicates the value of the this pointer (NUMBER column: 1). This entry is available for each event. The additional three entries show the result of the retrieve operation on the Customer business object (value 2 in the NUMBER column), the result of the retrieve by association to the Site business object (3), and the result of the query to the Site Reservation business object (4).

Figure 5.3 Trace Toolbar and Trace Explorer
Enable tracing for a business user

Tracing can be enabled for another business user. This can be necessary when a sequence should be traced for a user who has no access to SAP Business ByDesign studio but is used for testing the SAP Business ByDesign UI in the final work center. To enable the trace for another user, in the SAP Business ByDesign studio menu select \texttt{TOOLS \cdot OPTIONS \cdot SAP \cdot GENERAL}. Navigate to the \texttt{TRACING} tab. Enable the tracing for a business user by entering the business user's name.

5.5 Implementing a Number Range

Universally unique identifier

With many business objects, it is necessary to be able to directly access individual objects. You do this with a unique identifier. The business object engine automatically assigns a universally unique identifier (UUID) to every business object node. This UUID is used internally, for example, if you access a business object traversing an association or from a result set of a query. The computer provides the UUID directly based on data such as the serial number of the processor and a submicrosecond time stamp. The disadvantage of the UUID provisioning process is the hardly human-readable UUID; a UUID cannot be used for communication between people.

In many cases it is necessary to have a human-readable identifier for a business object. In some cases this identifier is expected to be entered manually by the users; in our example, this is the case for the Site business object. In other cases you can take over a human-readable ID from another business object. In a third case, you expect the system to provide a serial number as a human-readable ID.

Number range

Number ranges assign a serial number that forms a unique human-readable key, either alone or with supplementary information (such as an organization or a fiscal year). A number range assures that previously assigned numbers are not reissued. A number range contains a \texttt{number range interval} with a set of permitted characters. The number range interval is made up of numerical or alphanumeric characters and is limited by the \texttt{FROM-NUMBER} and \texttt{TO-NUMBER} fields.
A reuse service for number ranges is not supported in version FP3.0 of SAP Business ByDesign studio. The following workaround may be used. The implementation of a number range service consists of the following steps:

1. Define a new Number business object.
2. Implement the `GetNext` action and `BeforeSave` event of the Number object.
3. Implement the call to the Number object in the consumer business object.

For the definition of the Number business object, you may use the following coding:

```java
import AP.Common.GDT as apCommonGDT;

businessobject Number {
  [AlternativeKey] element RangeIntervalName : LANGUAGEINDEPENDENT_EXTENDED_Name;
  element NextNumberValue  : NumberValue;
  element NextID    : ID;
  element Changed   : Indicator;
  action GetNext;
}
```

**Listing 5.18** Definition of the Number Business Object

The business object definition consists of:

- **`RangeIntervalName` element**
  This element is a key of the business object. The Number business object supports multiple number range intervals. Usually, you will define one interval per business object, but it is also possible to share the same interval between different objects or to use multiple intervals for one business object, for example, by type, organization, or fiscal year. In the latter case, the key is the concatenation of the type, organization, or fiscal year and the number retrieved from the number range interval.

- **`NextNumberValue` element**
  This element represents the next free number in the interval.
NextID element
This element represents the next free number in the interval converted to an element of type ID.

Changed element
This is the indicator for internal processing (see explanation below).

GetNext action
This action increases the number range counter.

You can extend the Number business object, for example, by elements that define the first value in a number range, that define a prefix, and so on.

GetNext action
The following listing shows the implementation of the GetNext action of the Number business object:

```csharp
// Action: GetNext
import ABSL;
this.NextNumberValue = this.NextNumberValue + 1;
this.NextID = this.NextNumberValue.ToString();
```

Listing 5.19 Implementation of the GetNext Action of the Number Business Object

BeforeSave event
The following listing shows the implementation of the BeforeSave event of the Number business object:

```csharp
// Event: BeforeSave
this.Changed = false;
```

Listing 5.20 Implementation of the BeforeSave Event of the Number Business Object

Coding in the consumer business object
The following listings show the implementation of the BeforeSave event of the consumer business object, in our example the Site Reservation:

```csharp
var qyNumber = Number.QueryByElements;
var selParNumber = qyNumber.CreateSelectionParams();
var number;
var found = false;
// (1) check is ID is initial
if (this.ID.IsInitial()) {
  selParNumber.Add(qyNumber.RangeIntervalName, "I", "EQ", "SITERESERVATION");
  number = qyNumber.Execute(selParNumber).GetFirst();
```
if ( !number.IsSet() ) {
    // (2) number range interval does not exist -> create
    number = Number.Create();
    number.RangeIntervalName = "SITERESERVATION";
}
if ( number.IsSet() ) {
    // (3) number range interval exists -> check if locked
    number.Changed = true;
    this.Flush();
    if ( number.Changed ) {
        // (4) successful -> get new number
        number.GetNext();
        this.ID = number.NextID;
    } else {
        // (5) not successful -> raise message
        raise NumberRangeLocked.Create("E");
    }
} else {
    // (6) error message: No number range interval found
}

Listing 5.21 Implementation of the BeforeSave Event of the Consumer Business Object
(Site Reservation)

If the ID of the site reservation is initial (code block 1), a query is executed to retrieve the instance of the Number business object that represents the number range interval that belongs to the Site Reservation business object. If no interval is found 2, it is assumed that the number range interval is not yet created, and the program tries to create a new one.

If an interval is found, the program checks if the number range is locked 3. If so, it sets the Changed element to "true," triggers a flush to the Number business object, and checks if the Number business object has accepted the change. If this is the case, we can be sure that the Number business object is exclusively locked, and we can let the program get the next number and assign it to the ID of the Site Reservation business object 4. If the update of the Changed element is not successful, we can assume that the number range interval is locked by another session 5.
It is possible to implement the get number sequence either in the BeforeSave event or in the AfterModify event.

If you implement the get number sequence at the end of the BeforeSave event, it is very unlikely that a locking conflict occurs, because the time period between the lock acquisition in the BeforeSave event and the release of the lock after the physical save on the database is very short. We call this *late locking*. If you implement the get number sequence in the AfterModify event, the number range interval will be locked for other user sessions until the document is saved.

To complete the implementation of the number range, do not forget to set the ID field to read-only in the UI Designer to prevent a user from filling the ID manually.

### 5.6 Summary

With SAP Business ByDesign Scripting Language, you can implement business logic in business objects. It therefore refers to the global data type system and to the business object model of SAP Business ByDesign.

The “atoms” of the language are literals, local variables, and static or instance references to business objects. Path expressions explore the hierarchy of a business object model (including associations between business objects) or a structured data type. Based on these “atoms,” assignment operations, arithmetic, logical, and conditional operations can be defined. Control and loop statements are supported as well.

The generated service provider class provides common features of a business object, such as the management of the database tables (object-relationship mapping), locking and transaction handling, and main memory buffering. The following events can be used for implementing application-specific business logic:

- AfterModify event
- BeforeSave event
- Application-specific actions
SAP Business ByDesign scripting language provides tracing capabilities for error analysis. The tracing can be used in the development environment as well as in the production environment of a customer system.
Index

500 Internal Server Error  408

A

Access rules  414
Account ID check  244
ACID principle  234, 469
Action  101, 129
  Check  110
  Confirm Sales Processing  107
  Copy  110
  GetAvailableSites  110
  Instance-based  221
  MaintainSalesOrder  106, 110
  Mass-enabled  221
  Nested  222
  Parameter  222
  Pick in the Available Site  110
  Start Sales Processing  106
  Submit for Approval (Cancel)  107
Action evaluation  393
Activity diagram  105
Activity list  267
Ad hoc output  358
Adaptation content  269
Add-on solution  48, 66, 274, 283, 288,
  327, 331, 352, 368, 431, 438, 458, 469
  Adaptation mode  290
Address data  426
Address Snapshot business object  426
Administrator toolbar  466
Administrator user  442
Adobe LiveCycle Designer  48, 360, 465
  Integrated version  367, 368
Adobe Reader  50
Adopting existing form templates  366
Analytical application  327
Analytical data model  327
Analytics  22
Analytics enabling node  346
Analytics enabling object  346
Analytics engine  51
Application and User Management work
  center  360
Application error  408
Application-specific query  75, 115, 129
Approval scenario implementation  373
Approval task  372
Approval Task Wizard  373
Arithmetic operations  338
Assigned object, maintaining  190
Assignment  213
Association  77, 85, 469
  Composition  77
  Features  231
  ToParent  251
  ToSite  251
Asynchronous message-based communication  69, 378, 379, 394
ATP check  70
Attachment management  430
Authorization
  Read and write access  187
  Role-based  185
Authorization check  188, 393, 400
  Instance  188, 190, 191
  Instance implementation  191
  Start  188
Authorization concept  208
Authorization policy  186, 188
Authorization profile  324
Avoiding a division by zero error  339

B

B2B (business to business)  54
B2B service  355
Background process  78
Background Processing  52, 53
Basic find  113
BC option  360
BC set  270, 278, 283, 360
Block  65
Boolean and conditional operations  339
<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser</td>
<td>469</td>
</tr>
<tr>
<td>Buffering</td>
<td>224</td>
</tr>
<tr>
<td>Business Adaptation Catalog</td>
<td>269</td>
</tr>
<tr>
<td>Business administration object (BAO)</td>
<td>78</td>
</tr>
<tr>
<td>Business analytics</td>
<td>327, 464</td>
</tr>
<tr>
<td>Business analytics tutorial</td>
<td></td>
</tr>
<tr>
<td>Creating a restricted or calculated key figure</td>
<td>337</td>
</tr>
<tr>
<td>Defining a data source</td>
<td>332</td>
</tr>
<tr>
<td>Defining a joined data source</td>
<td>348</td>
</tr>
<tr>
<td>Defining a report</td>
<td>336</td>
</tr>
<tr>
<td>Business Analytics work center</td>
<td>327, 336</td>
</tr>
<tr>
<td>Business application</td>
<td>50</td>
</tr>
<tr>
<td>Business area</td>
<td>269</td>
</tr>
<tr>
<td>Business character</td>
<td>417, 419</td>
</tr>
<tr>
<td>Company</td>
<td>419</td>
</tr>
<tr>
<td>Functional unit</td>
<td>420</td>
</tr>
<tr>
<td>Permanent establishment</td>
<td>419</td>
</tr>
<tr>
<td>Profit center</td>
<td>420</td>
</tr>
<tr>
<td>Program</td>
<td>420</td>
</tr>
<tr>
<td>Reporting line unit</td>
<td>420</td>
</tr>
<tr>
<td>Segment</td>
<td>419</td>
</tr>
<tr>
<td>Business concept</td>
<td>76</td>
</tr>
<tr>
<td>Business configuration</td>
<td>28, 51, 267, 272, 382, 463, 469</td>
</tr>
<tr>
<td>Use cases</td>
<td>271</td>
</tr>
<tr>
<td>Business configuration architecture</td>
<td>269</td>
</tr>
<tr>
<td>Business configuration content</td>
<td>270</td>
</tr>
<tr>
<td>Business configuration object (BCO)</td>
<td>51, 78, 270, 283</td>
</tr>
<tr>
<td>Business configuration tutorial</td>
<td></td>
</tr>
<tr>
<td>Defining a BC set and BCO</td>
<td>278</td>
</tr>
<tr>
<td>Defining a BC set on an existing BCO</td>
<td>280</td>
</tr>
<tr>
<td>Defining a business option</td>
<td>276</td>
</tr>
<tr>
<td>Defining a business topic</td>
<td>277</td>
</tr>
<tr>
<td>Defining a scoping rule</td>
<td>282</td>
</tr>
<tr>
<td>Defining an implementation project template</td>
<td>282</td>
</tr>
<tr>
<td>Business Configuration Wizard</td>
<td>276</td>
</tr>
<tr>
<td>Business Configuration work center</td>
<td>267, 283</td>
</tr>
<tr>
<td>Business logic extension</td>
<td>297, 300</td>
</tr>
<tr>
<td>Business object</td>
<td>49, 50, 56, 73, 76, 168, 191, 285, 413, 433, 469</td>
</tr>
<tr>
<td>Access-controlled</td>
<td>193</td>
</tr>
</tbody>
</table>
Business topic 269, 274
Business transaction document 77, 86, 122
Business user 400, 443
businessobject statement 87
Button 469

C
Calculated element 98
Calculation 75
  Price 105
Cardinality 81, 100
CCTS representation type 147
Change project 268
Characteristic 328, 334
  Selecting 331
Characteristic property
    Display setting 344
    Hierarchy setting 344
    Label 344
    Value selection 344
Check action 241
Check, initial 253
Child node 77
Clearing House business object 418
Client compatibility 407
Client monitor (HTTP trace) 202
Cloud 21, 33, 49
Code completion 212
Code value 229
Co-innovation 30
Collaboration 29
Collection 216
Collection, method 216
Comment 210
Communication pattern 70, 71
Compiler 49, 54
Compound Service → Service
Configuration content 270
Conflict and error handling 394
  Implementation 407
Conflict and error monitoring 395
Conflict and error resolution strategies 394
Constraint 270
Consumer-independent software implementation 76
Consuming object 380
Content classification 439
Control center 131
Controller implementation 306
Controls 62
Copy action 244
Copy sequence 240
Core data type 89
  Code 92
  Indicator 93
Core Service → Service
Cost object 431
Create a solution 43
Create operation 219, 403
Create Screens dialog box 44
Create Screens Wizard 177
Create script files 44
Create Solution dialog box 43
Create user interfaces 44
Creating a solution 445
CRUD service 58, 59
Crystal Reports 48, 50
Currency conversion 343
Customer 418
Customer invoice 80
Customer invoice processing 382
Customer-specific reporting 327
Customer-specific solution 30, 438, 448, 455, 458

D
Data binding 140
Data model browser 202, 206
Data source 328, 470
  Data structure 328
  Display setting 328
  Extension field 351
  Field-to-field mapping 328
  Inner join 347
  Join 347
  Join condition 347, 350
  Left outer join 347
  Partner Report 345
Index

Data source (Cont.)
- Report 328
- Report assignment 345
- Union 350

Data source operation
- Drill-down 331
- Pivoting 331
- Rollup 331
- Slice and dice 331

Data Source Wizard 332, 350

Data structure 358

Decoupling 26

Define a status schema 102
Define the action 102
Define the precondition 102
Define the status variable 102
Delete operation 221
Deleting a solution 445

Deployment 270

Deployment unit 53, 68, 74, 378, 382, 414, 470

Deprecation 72
- Period 73

Design document 440
Design time tool 47
Detailed design 85
Determinations 59
Developer user 442
Development guideline 467
Development object 54, 55, 285
Device-specific software 201
Document address 425
Document window 42
Domain-specific language 209
Dot notation 445

E

Easy Form Editor 360, 367, 368
Element statement 87
Element, transient 99
Embedded analytics 29
Embedded component 195, 199, 208
- Reuse UI 427
- Test 310

Embedded component (Cont.)
- Use case 195

Employee business object 418, 419
Enterprise service 470
Entity 470
Error handling 357, 407
Error list 43

Event 236
- Action event 237
- BeforeSave 255, 262
- Determination event AfterModify 236
- Determination event BeforeSave 237
- Validation event CheckBeforeSave 237

Event handler 141, 169

Exactly once request execution 404

Exception aggregation 341

Execute operation 227

Extending a form message type 368

Extending the Customer Invoice Request 384

Extending the Site Reservation 383

Extensibility 26, 285, 324, 464

Extensibility Explorer 293
- Change transaction 293
- Stable anchor 293

Extensibility tutorial
- Adding a common task to a work center view 323
- Adding a mashup to a UI 319
- Adding an embedded component to a UI 302
- Adding an extension field to a business object 290
- Adding an extension field to a UI 293
- Adding an extension field to enterprise search 297
- Changing the work center order 322
- Defining a data mashup 317
- Defining a web service 316
- Extending a result list 296
- Extending a selection parameter structure 297
- Hiding a button 321
- Implementing a business object extension 298
Extensibility tutorial (Cont.)

Implementing the controller for a supplemental business object 307
Renaming a label or hiding a UI field 320
Extensible Markup Language (XML) 473

Extension
Access 298
Data type 292
Eventing 298
Features 298
Scripting language 298
Sequence of events 300
Extension field 285, 292, 324
Label change 296
With business partners 293
Extension field calculation 286, 297
External namespace 445
External web service 355, 399, 412

F
Fact sheet (FS) 132
Field extensibility 285, 413
Field-to-field mapping 359
File input 412, 466
File Input Run 409
Filtering 51
Find form 164
Fine-tuning 273, 283
Flat database table 329
Flexibility 26
Flexibility change log 290
Floorplan 470
Form data type 358
Form rendering 360
Form template 359
Form template group 359
Form Template Maintenance 358, 366
Form template variant 359
Configuration 361
Extension field 368
Form Template Wizard 361
Form-based document 358
Forms 465
Foundation 68, 470

Frontend print scenario 361
Frontend scripting 165, 168, 207
Trace 202, 206
Function
Instance-based 233
Static 233
Functional unit, role 420

G
GetNext action 262
Global data type 88, 90, 470
Aggregated 90, 126
Basic 126
Global Data Type Catalog 462
Go-live activity list 273
Graphical user interface (GUI) 473
Guided activity floorplan (GAF) 132

H
Help desk cost 435
House Bank business object 418
HTTP error and fault message 407
HTTP notation 445
Human-readable identifier 83, 244, 260

I
Idempotency 404
Identity 424
If-else statement 214
Implementation Manager toolbar 453
Implementation project 267
Implementation project template 282, 284
Import 210
Import statement 87
Inbound processing 53, 69, 381
Error 410
Inbound service 391
Interface 410
Incident 446
Content 455
Handling by key users 446
Report 454
Index

Incident (Cont.)  
  Report from Business Center  447
Incident management  34
Import  143, 306, 471
Input parameter  100
Input processing  356
Insight-to-action  29
Installation  466
Instance identification  391
Integration  66
Integration concept  431
Integration scenario  38
Internal communication  356, 385, 411
Internal namespace  445
ISO 11179  123
IsSet check  232
IsSet operation  220

L

Job business object  422
Join  58, 111
  Logic  58

K

Key figure  328, 334
  Basic  329
  Calculated  329, 337, 338
  Calculation editor  338
  Counter  337
  Restricted  329, 337, 343
  Selecting  329
Key Figure Wizard  337, 343

L

Lambda expression  217
LAN  471
Leading business object  140
Learning Center  461
Lifecycle management  34, 56, 435, 438, 467
  Continuous improvement phase  435, 456
  Deployment phase  435, 447
Lifecycle management (Cont.)
  De-provisioning phase  436
  Development phase  435, 445
  Operation and deployment phase  454
  Operation and support phase  435
  Preparation phase  435, 436
Lifecycle management tutorial
  Analyzing an incident trace  456
  Creating new users  443
  Deploying a customer-specific solution  453
  Search for known problems  446
Literal  211
Lock  234, 237, 471
  Application lock  238
  Database lock  238
  Late locking  264
Lock handling  58
Locking  79
Logical and conditional operation  213
Logical unit of work (LUW)  471
Loop statement  215
Loose coupling  195

M

Main memory buffer  58
Main node semantic  124
Mashup  50, 310, 325, 471
  Architecture  310
  Category  311
  Configuration and integration  315
  Development  315
  Integrated authoring tool  314
  Type  310
  Write-back  311
Mashups  31, 464
Mass data run object (MDRO)  53, 78
Master data address  425
Master data object (MDO)  78
Master data, maintenance  254
Mathematical operator  213
Message  75
Message-based communication  68, 378, 382, 414
Method tool tip  222
Microsoft Silverlight 50, 60
Mobile strategy 201
Mobile UI 208
Mobility 21, 29
Mockup 41
Model View Controller (MVC) 74
  Design pattern 61
Multiple selection parameter 228
Multiplicity 80, 128

N

Naming Conventions and Useful Guidelines 421
Naming rules 123
  Capitalization 124
  Object class 126
  Property 126
  Representation 126
Node 75
  Action 100
  Additional Service 87
  Administrative 127
  Aggregation 105
  Category 127
  Classification 127
  Delegation 105
  Extendable 291
  ID 127
  Invoice Reference 87
  Key 127
  Lockable 80
  Processing 127
  Property 127
  Reference 127
  Root 77
  Status 127
  Subnode 82
  Transient 75, 99
Notification 70
  Notification communication pattern 393
  Notification task 372
Number range 260
  Interval 260

O

Object instance floorplan (OIF) 132
Object relation mapping 57
Object value selector (OVS) 133
Object work list (OWL) 53, 132
  floorplan 296
Object-based navigation 133, 171, 207, 238
  Configuration 177, 180
  Implementation 172
  Troubleshooting 181
  Use Case 171
Offering object 380
On-demand service 21
On-demand strategy 23
Operation 142, 143
Organizational function 420
Organizational structure 419
Organizational unit 419
  Hierarchy 421
Organizational unit template 421
Outbound processing 53, 69, 357, 389, 390
Outbound service 371
Outport 143, 472
  Parameter 303
Output management 358, 410
Output management architecture 358

P

Pace of innovation 22
Pane 153
  Adding and removing 153
  Advanced List Pane (ALP) 154
  Features 155
  Newspaper format 153
Party 422
  Role 422
Patch 450, 454
Path 124
Path expression 212
Pattern 472
Pick action 250
Platform tutorial
  Including the Address Snapshot  427
  Including the Attachment Folder  430
  Including the Text Collection  428
Port type package  133, 177, 312
Position business object  422
Precondition  101
  Enabling  103, 248
  Inhibiting  103, 248
  Required  104
  Requiring  248
Primary service provider  49
Primitive data type  89
Process (or document) flow  38
Process automation  27
Process Communication Error  397
Process integration  53, 382, 388, 465
  Definition  392
  Persistence  379
Process scenario, extensible  465
Process variant  271
Process-integrated output  358
Programming language  209
Properties window  43
Prototype  40
  Advanced Prototype  41
  UI prototype  40
Public model  56, 71, 413, 466

Q
Qualifier  124
Quality review  450
  Request  451
Quality Review Questionnaire  450
Query  111, 227, 238
  Default set  159
  Extension  296
Query operation  59, 405
Query/response pattern  70
Quick Activity Floorplan (QAF)  45, 63, 132

R
Real-time analytics  327
Relationship  39, 128
Release action  245
Relevance condition  357, 389
Report Wizard  336, 346
Repository  48, 54, 74
  View  42, 43
Request business object  381
Request/confirmation pattern  70, 393
Reserved word  210
Reset method  232
Responsible manager  423
REST (Representational State Transfer)  315
Result list  111
Retrieve method  226
Retrieve operation  244
Reuse service  425, 433
Rich Internet application (RIA)  50
RootQuantity  342
RSS (Really Simple Syndication)  315
Rules for transaction and lock handling  238
Runtime tool  47

S
Sales order  80, 383
  Creation  253
  Item  118
  Party  120
  Party role  120
  Schedule line  119, 120
  UI  305
  Update  254
Sandbox solution  439
SAP BI OnDemand  24
Architecture  286
Environment  437
Platform  433
Studio frontend  47, 54, 55
SAP Business ByDesign (Cont.)
Service 56, 470, 472
   on-demand 21
Version N 72
Service adaptation 50, 74
Version N+1 72
Service Consumer 473
SAP Business ByDesign Business Center
Service integration 355
437, 458, 461
Service integration tutorial
Forum 437
Adding an extension field to a form
SAP Business ByDesign Business Center
Adding extension fields to an existing
437, 458, 461
process integration 397
Service Consumer 473
SAP Business ByDesign for Subsidiaries
Adding an existing form template 368
23
Configuring a form template variant
SAP Business ByDesign Library 438, 461
with Adobe LiveCycle Designer 363
SAP Business ByDesign Forum 437
Defining a web service authorization
SAP Business ByDesign Repository 55
profile 402
SAP Business ByDesign Scripting Language 209, 223, 256, 346
Defining an approval scenario 373
Tools library 240, 243
Defining an external web service 399
SAP Business ByDesign studio Library 438, 461
Defining the interaction structure and
SAP Business ByDesign studio Wiki 437
mapping for an internal communication 385
SAP BusinessObjects 470
Finalizing the approval scenario 375
SAP Carbon Impact OnDemand 24
Implementing the process integration
SAP Career OnDemand 24
conditions 390
SAP Cloud Infrastructure 34
Installing Adobe LiveCycle Designer 360
SAP Global Data Type Catalog 462
Service Integration Wizard 385
SAP namespace 81
Service interface 356, 358, 410
SAP Sales OnDemand 23
Service operation 357
SAP Sales Order 76, 118
Service orientation 26
SAP solution adviser 267
Service Provider 473
SAP Sourcing OnDemand 24
Service provider class 49, 56, 57
SAP Store 32, 448, 451, 472
Signature 473
Access 451
Silverlight 463
Features 452
Simplicity 26
SAP Streamwork 24
Site 81
SAP Support 447
Site availability check 244
Application sharing 447
Site category 81
SAP Travel OnDemand 23
Site ID check 244
Scalable solution 30
Site Management 37
Scoping 267
Site reservation 81, 105, 382
Scoping question 275
Cancellation 372
Scoping rules 272, 281
Work center view 182
Scoping status 272
Site Reservation business object 103,
Scoping workshop 267
108, 110, 114, 168
Screen Creation Wizard 133
Site Reservation Object Work List (OWL) 45
Scripting Language, syntax documentation 462, 463
Snippet 59
Search engine 65
SearchText element 113
Selection parameter 111, 227
Self-enablement system 437
Index

SOAP (Simple Object Access Protocol) 315
Software layer 288
Solution 473
Solution design 441
Solution Documentation 441, 450
Solution Explorer 42
Tool window 43
Solution implementation, troubleshoot 454
Solution profile 51, 269, 270
Solution purchasing and activation 454
Solution search 446
Special-purpose web service 356
Stable anchor 72
Standard function library 233
Standards and criteria 450
Documentation 450
Functional correctness 450
Performance 450
Security 450
Usability 450
Start condition 357, 390
Statement 210
Status and action management 101
Status schema 101
Status transition 101
Status value 101, 103
Status variable 101, 103
Cancellation Approval Status 109
Consistency Status 109
Release Status 109
Sales Processing Status 109
Site Reservation Status 109
Supplemental business object 286, 305, 325
Definition 305
Supplier business object 418
Support 34
Switch statement 214
Synchronizer 102, 104
Synchronous call-based communication 379
Synchronous web service 399, 407
System message 117
Message ID 118
System message (Cont.)
Text 118
System-to-system communication 357

T

Task type 370
Category 370
Tax Authority business object 418
Technical and Support Documentation 450
Technical objects (TecO) 78
Technical support cost 435
Technical user 400
Template/projection concept 418
Tenant 48, 49
Test deployment 449
Test user 443
Text search 112
Tight coupling 195
Total cost of ownership 435
Trace
Data volume 258
Developer 256
Enable for a business user 260
Features 257
For test users 256
Incident 256
Transaction 234, 473
Interaction phase 234
Save phase 235
Transactional behavior 58
Transactional RFC 473
Transformation 51
Transformation rules 331
TREX 473
Trigger test deployment 449
Button 450
Type inference 212
Type system 211

U

UI Designer 48, 64, 135, 207, 462, 473
BO Browser/Data Model 136
Configuration Explorer 136
UI Designer (Cont.)
    Drag-and-drop 136
    Extensibility Explorer 136
    Preview mode 137
    Properties 136
    Toolbox 136
UI extension 287, 320
    Key user tool 289
    Lifecycle stability 288
UI pattern
    Advanced find form 113
    Basic find form 113
    Find form 113
UI → User interface
UI Style Guide 462
Unified modeling methodology (UMM) 70
Uniform Resource Locator (URL) 473
Unit conversion 343
Universal Unique Identifier (UUID) 83, 260, 473
Update development 457
User 424
User interface 473
    Architecture 60
    Code data type 149
    Controller engine 51
    Data model 140
    Data type identifier 148
    Dedicated interaction paradigm 131
    Development 48, 463
    Display type 144
    Embedded component 302
    Engine 55, 60
    Extension type 287
    Floorplan 62, 131, 137, 181, 201, 207, 287, 312
    Model controller 141
    Pattern engine 61
    Pattern hierarchy 137
    Pattern-based 131, 207
    Property 145
    Reusable building blocks 131
    Site map 134
User interface tutorial
    Adding a button to a form 151
    User interface tutorial (Cont.)
        Adding a common task to a work center view 184
        Adding a field to a form 150
        Adding a work center view to an SAP work center 183
        Adding an embedded component to a floorplan 199
        Assigning a work center view to a work center 182
        Configuring a preview pane 158
        Configuring the find form 164
        Creating a data list for a code list 170
        Creating a list pane 156
        Creating an embedded component 196
        Creating screens for a business object 133
        Defining a segregation of duties conflict 194
        Defining a value set and the default value set mapping 163
        Defining an OBN create scenario 174
        Defining an OBN create with reference scenario 178, 180
        Defining an OBN Show Details scenario 175
        Defining dynamic behavior of an action button 166
        Defining the screen title 154
        Filling a list with data from a query 159
        Implementing an instance authorization check 192
User-to-system communication 357
UUID 473

V

Variable 210
    First assignment 211

W

WAN 473
Web report 331
Web service 53, 355, 473
Web service authorization 403
Index

Web service definition language 403
Web service endpoint 403, 408
Web service engine 51
Web service interface operation 409
Web Service Provisioning Wizard 399
Web Service Reliable Messaging 404
While loop 215
Whitespace character 210
Wide area network (WAN) 473
Wireframe 40
Wiring 382
Work center 53, 131, 183, 208, 444

Work center view 132, 183, 444
Adding or removing 321
Authorization 401
Extension 322, 323
World Wide Web (WWW) 473

X

Xcelsius 48
XML 473
XML-based message 357