SAP® Business One

Version 1.0

English

SAP Business One Software Development Kit (SDK)

Standards and Guidelines

SAP AG
Neurottstr. 16
69190 Walldorf
Germany

Copyright

Contents

Preface

This document describes the Standards & Guidelines for working with the SAP Business One Software Development Kit.

The document consists of six sections:

- General Information about the SAP Business One SDK
- Specific Information for working with the Data Interface API
- Specific Information for working with the User Interface API
- Registering an Add-on SAP Business One
- Tips and Tricks
- Glossary

All statements with reference to SDK version 6.5 are **preliminary** and **subject to change**.

Document History

Version	Section	Description
1.0	1.4.1	Updated naming reservation process
1.0	3.1	Added connect string mechanism
1.0	4	Added register/deregister Add-on
1.0	1.13	Added rules for event handling
1.0	3.4	Added Screen Painter
1.0	5	Added Tips and Tricks

How to read this document

The target group for this document includes both ISVs and Customers. In addition to the general Information relevant for both parties, customers should focus on the User Interface API.

Icons

Icon	Meaning
	Caution
	Example
	Note or Tip
	Recommendation
	Syntax

1 General Information

1.1 Introduction

The SAP Business One SDK consists of two Application Programming Interfaces:

1. The **SAP Business One Data Interface API (DI API)**
2. The **SAP Business One User Interface API (UI API)**

The **SAP Business One SDK Data Interface API -** is a collection of COM objects called business objects. These objects describe various methods for updating, retrieving, and manipulating data in the SAP Business One database.

For example, you can use the Data Interface API to connect an external data warehouse management system to SAP Business One.

The **SAP Business One SDK User Interface API -** is a collection of DCOM objects that provide access to forms, controls within these forms, and menus. DCOM (distributed component object model) is a standard interface for object communication in distributed applications.

From SDK 6.2 you can use the User Interface API to integrate customer specific programming or third-party products into SAP Business One.

In addition helpful tools and important documentation (such as *Standards and Guidelines*) are included with the SAP Business One SDK.

1.2 SAP Business One Architecture

SAP Business One is implemented as a two-layer architecture. The client software consists of a graphical user interface and business object classes that connect to the database. The server software consists of a fax, mailing, and dialing service as well as the database itself.

The SDK business objects (DI API) connect to the database directly through the ODBC-protocol. They do not communicate with any of the SAP Business One client components.

1.3 Development Tools

Microsoft Visual Basic 6.0 and Microsoft Visual C++ 6.0 are the recommended development tools.

The examples in this document refer to MS Visual Basic 6.0.

For information on recommended development environments for use with the SAP Business One SDK, see Note 615987 in the SAP Service Marketplace.

1.4 Naming Conventions

1.4.1 Namespaces for Partners

To avoid conflicts between SAP Business One Add-ons programmed by different partners or software vendors, you must use naming conventions for new objects that do not belong to the SAP Business One standard software. The following objects in particular can cause problems and need to be considered:

- Database tables and indices
- Delivered files (dll's, executables, ocx, resource files, etc.)
- Names for projects, modules, classes and their properties
- IDs for forms and screens
- Filenames for projects, modules and classes
- User-defined fields within SAP Business One

The following naming conventions were established to avoid conflicts between object names. When you register as a (software-) partner for SAP Business One SDK, you need to apply for a namespace as described here:

To obtain a reserved name prefix:

1. Create an OSS message for component SBO-SDK and title "PREFIXRES ". Provide the following data in the detailed text:
    - The name of the planned partner solution
    - The contact person (e-mail) in the partner's company
    - The name of the responsible contact in field (partner manager)
    - Your preference for the prefix:

      Due to technical restrictions in SDK6.01 and SDK6.2 the prefix should be limited to 3 characters. Note that the prefix must have a minimum of 3 characters for ABAP name spaces.

      The syntax of the java project name should be adapted to the usual internet syntax (such as: '.com.sap') and should use the internet address of the partner's company. The usual prefix (e.g. 'XYZ') is added to the internet syntax in lower case (e.g. '.com.sap.xyz') as a suffix. If you also need a java prefix, please specify your internet address.

      

      You must wait until you receive a response to your message in step 1 before attempting step 2.
2. Reserve a name for name server (= ABAP name space):
    a. Go to SAP Service Marketplace: `http://service.sap.com/namespaces` and choose: *reserve an accepted namespace*.
    b. Give feedback about your reservation by sending the message (created in step 1) back to SAP.
3. Please confirm that the message has been closed by SAP.

1.4.2 Using the Partner Namespace:

The following naming conventions apply for the partner namespace. Assume that your namespace is XYZ: For development with the SAP Business One SDK, use the delimiter '_'. This denotes the end of a namespace identifier string:

- Name database tables XYZ_<user-defined in upper case>
- Name any delivered files XYZ_<user-defined in upper case>
- Name the file names for projects, forms, modules and classes XYZ_<user-defined in upper case>
- Name property names of projects, modules, forms and classes as well as ID's for forms and items XYZ_<user-defined in upper case>.
- Name Project files / packages of the JavaConnector <internetsyntax>.xyz (in lower case)
- User-defined fields in SAP Business One should be named U_XYZ_<user-defined in upper case>

It is possible, but unlikely, that some files with the same name already exist in the 'Windows World'. Following the recommendations given above, you can at least be sure that you won't have conflicts between SAP-qualified solutions.

1.5 Partner Package

A partner package is made up of the following components:

- SAP Business One - deployed to the customer through the usual SAP distribution channel
- Add-on from partner is delivered on a separate CD. The partner is responsible for ensuring the Add-on fit to the versions of SAP Business One SDK and the version of SAP Business One used at the customer site.

An Add-on delivery must include the following items:

- Components developed by the partner
- An appropriate installation procedure must be used for the installation and de-installation of the partner components including registration of SAP Business One, (see Section 4)
- Appropriate documentation for installation and usage of the partner components. For more information, see Section 1.14

The partner is responsible for the upgrade consistency of the Add-on components delivered to the customer.

The SAP Business One runtime SDK is part of the standard SAP Business One installation.

1.6 Multilingualism

The logon language is determined by the language property of the company object. The different language keys have logical constants. They are members of the enumeration class SAPBobsCOM.BoSuppLangs, that is described within the online documentation of the SAP Business One SDK Data Interface API.

You need to ensure that external components use the same character set as SAP Business One.

When you work with the SAP Business One User Interface API, SAP recommends that you use resource strings to keep your development more language independent.

1.7 Modifications to the Standard

In general, modifications to standard SAP Business One objects are not allowed. You can insert new table fields using *Tools* → *Manage user fields* → *Select category* → *Remove/Change/Insert user fields*. Most business objects have a 'UserFields property', for example the business partner object. This property contains a reference to a UserFields object, which is designed for accessing the user fields. For more information about user-defined fields, see Section 2.1.3.3. Ensure that you adhere to the naming conventions when creating new objects (Section 1.4).

It is **very important** that you document any modifications made to screens using the UI API. This will allow you to easily adjust your solution to future SAP Business One releases.

Take care to always add your new module callout to the SAP Business One *Command Center*. This will ensure the ability to upgrade in future versions of SAP Business One, should changes occur in the structure of standard lower level menus.

1.8 Authorization Checks

There are two types of SAP Business One SDK object:

1. **Business Objects** - grant access to different business data.
2. **Raw Data Access Objects** - can be used to access data on a generic level.

Authorization checks are performed for all SAP Business One business objects. If a user does not have permission to view objects in SAP Business One, the user will not have permission to access the data belonging to the business objects using the Data Interface API. For example, the Items object has the same authorization as the material master data form in SAP Business One.

Please exercise caution when using the RecordSet object (see Section 2.3.1). It does **not** perform authorization checks as it is a raw data access object.

1.9 Platform Independency

SAP Business One is restricted to the MS SQL Server, and cannot be used on other platforms. However, to ensure independency for upcoming platforms, use the existing business object classes for data access rather than the RecordSet object (see 2.3.1).

1.10 Compatibility of SAP Business One SDK Versions

The Data Interface API consists of methods that allow access to business objects (could be compared to the BAPI approach for SAP Net Weaver) and the interface to access raw data. Compatibility, however, can only be assured for the business object part of the interface.

Applications using the User Interface API depend on the layout of the SAP Business One version it was developed for. As a result an application may fail when SAP Business One is upgraded.

For details, see the *Release Strategy* document for the SAP Business One SDK and Note 626733 in the SAP Service Marketplace.

1.11 Built-In Functionality for Partner Solutions

Using the SAP Business One SDK Data Interface API, you can access

- Additional database tables (SAP recommends that you **do not** use additional tables. You should use User-defined database tables instead)
- User-defined database tables
- User-defined fields

1.11.1 Additional DB-Tables from Partners

If you create additional data base tables, follow the naming conventions (see Section 1.4). There is no tool support to create new database tables.

Additional database tables are not visible within the SAP Business One data dictionary.

The standard business objects in the Data Interface API cannot access additional database tables. In this case, use the RecordSet object instead.

1.11.2 User-Defined Database Tables

You can create and maintain user-defined database tables in SAP Business One (from SDK 6.2). User-defined database tables are also visible in the SAP Business One data dictionary.

At SDK 6.5 the length of user-defined table names was extended to 20 characters. This particular length allows you to implement name prefixes. The advantage of user-defined tables is that they are fully integrated into SAP Business One. For example, in the case of queries. For this reason, SAP recommends that you should use user-defined tables instead of solution specific database tables.

1.11.3 User-Defined Fields

You can use user-defined fields to expand the SAP Business One data model without defining new database tables. These are not affected by release or support package level upgrade (see Section 2.1.3.3). Since user-defined fields can be accessed by standard business objects in the data interface API, SAP recommends that you create user-defined fields instead of new database tables.

It is not possible to deliver these fields automatically to the customer site in the current version. You must create them manually on the customer's system. For example, you can create these items using the Data Interface API.

1.12 Qualification

To ensure qualification of your Add-on, check that you have complied with the following prerequisites:

- Apply the naming conventions described previously.
- Follow the information rules and recommendations given in this document.

For more information about the qualification of Add-on solutions, and the certification of Add-on suppliers, see alias ISV in the SAP Service Marketplace.

1.13 Handling Events

You need to document event handling with care to prevent misleading side effects involving other Add-on solutions also developed with the UI API.

Take care to document each event and form, and in every case include the following information:

- Expected situation (prerequisite)
- Action that is performed (change of data)
- Condition for a break in event chain (set parameter bubblevent)
- Situation that can be expected by possible successors (other Add-on's) handling the same event

1.14 Creating SAP-like Documentation

1.14.1 Motivation

The SAP Business One Software Development Kit (SDK) allows you to integrate third-party applications with SAP Business One both at the database and user interface level. It is an ideal platform to change and extend the functionality of SAP Business One with specific functionality required by your particular business.

This section offers standards and guidelines to help you write user-oriented documentation for the functionality you develop using the SDK. The SDK provides a template that will assist you to write your documentation: *Offline Template for Writing SAP Documentation*. This template is based on the *Standards and Guidelines for Writing at SAP*. The *Standards and Guidelines for Writing at SAP* and the offline template may be subject to change. To find the most current version, see SAPNet under the alias **S&G.**

1.14.2 Writing Documentation

1.14.2.1 Who is Your Target Audience?

The target audience for your technical documentation is the user who will use the functionality you have developed using the SDK. This person might be a technical consultant or an end user.

When writing technical documentation, keep your target audience in mind. Ask yourself the following questions:

- What information does the target audience need?
- How does this information help the target audience?

1.14.2.2 Structure of Your Documentation

The following section provides you with an overview of the keyblocks (parts of the template) you should use when you write technical documentation. It shows whether the keyblock is required or optional, and specifies the standard keyblock headings. If you decide not to use an optional keyblock, delete the heading for that keyblock.

Keyblocks Available

Information Class	Page
Background	15
Component	12
Example	14
Field	14
Function	13
Message	14
Method	15
Object	12
Procedure	14
Process	12
Report	13

Component	Answers the questions	Use
Title	What is this component called?	Required
Purpose	What does the component do (in general)? Why do I need the component?	Required
Implementation Considerations	Under what conditions (business or technical) should I implement and use this component?	Optional
Integration	How is the component related to other components or functions? Where does the component fit in the component hierarchy? What other components do I need?	Optional
Features	What does the component do (details, specifics)? What capabilities does it provide?	Optional
Constraints	What does the component not do?	Optional
Example	Can you give me an example of the use of this component?	Optional
Object	**Answers the questions**	**Use**
Title	What is the object called?	Required
Definition	How is the object defined?	Required
Use	Where and how can I use the object (in general)? What constraints apply to the use of this object?	Optional
Structure	What are the parts of the object? What is each part used for?	Optional
Integration	How is this object related to other objects?	Optional
Example	Can you give me an example of this object?	Optional

Process	Answers the questions	Use
Title	What is the process called?	Required
Purpose	What does the process do? Why should I use the process? What are the business or technical benefits?	Required
Prerequisites	What do I need to do before the process starts? What are the preceding processes?	Optional
Process Flow	When does the process start? How does the process work?	Required
Result	What is the result of the process? What are the follow-on processes?	Optional
Example	Can you give me an example of this process?	Optional

Function	Answers the questions	Use
Title	What is the function called?	Required
Use	What does the function do (in general)? Why do I need the function?	Required
Integration	How is it related to other functions? What other functions are needed for this function to work?	Optional
Prerequisites	What has to be in place or has to happen before I can use this function? What events or system settings (Customizing) trigger this function?	Optional
Features	What does the function do (details, specifics)?	Optional
Activities	What does the user do? What does the system do?	Optional
Example	Can you give me an example of the use of this function?	Optional

Report	Answers the questions	Use
Title	What is the report called?	Required
Use	What is the purpose of the report (general and commercial)? Why do I need the report?	Required
Integration	What other reports can I call up? What are the options for navigating to other reports?	Optional
Prerequisites	What are the necessary prerequisites or parameter settings in order to run this report? What are the system settings required to run this report?	Optional
Features: Selection Standard Variants Output	What is the specific purpose of this report? What changes are written by the report to the database?	Optional
Activities	What are the report-specific navigation options?	Optional
Example	What would be a specific instance of this report?	Optional

Procedure	Answers the questions	Use
Title	What task am I performing?	Required
Use	What are the purpose and result of this procedure? Why should I use this procedure?	Optional
Prerequisites	What do I need to know before I start this procedure? What are the preceding procedures?	Optional
Procedure1: Step Step result	 What specific action do I take? What is the result of this action?	 Required Optional
Result	What is the result of completing the task? What are the follow-on procedures?	Optional
Example	Can you give me an example of this procedure?	Optional

Field	Answers the questions	Use
Short Text	What is this field called?	Required
Definition	What is this field for? What does this field mean?	Optional
Use	What happens when I enter a value or set an indicator? Under what circumstances do I use the field? What do I enter in this field and why (no specific values)?	Optional
Dependencies	What other fields or functions does this field influence? Are there other effects that I might not expect? Are there restrictions? Can field values be defined in Customizing?	Optional
Example	If this field has a certain value or setting, what happens?	Optional

Message	Answers the questions	Use
Short Text	What happened? What must I know or what must I do before I can continue?	Required
Diagnosis	What caused the message to appear? What happened	Optional
System Response	How does the system respond? What processes does the system carry out? What functions can the system not complete?	Optional
What To Do	What must I do about the message? What can I do to prevent this happening again? (Links to other transactions.)	Optional
Procedure for System Administrator	What can I do, assuming I have the required Customizing or administrative functions? (Links to IMG or other transactions.)	Optional

Example[2]	Answers the questions	Use
Title	What is this an example of?	

[1] If you use the keyblocks *Use* or *Prerequisites* and if you are describing a **single** procedure, use the keyblock *Procedure*. If you are describing more than one procedure, use a specific heading for each procedure rather than the generic keyblock heading *Procedure*.

[2] An example has keyblocks only when it serves as an example for a step-by-step procedure or for a process. If this is the case, use the keyblocks for the relevant step-by-step procedure or process.

Background	Answers the questions	Use
	What kind of background information may be useful for the user?	

Method	Answers the questions	Use
Title	What is the Method called?	Required
Use	What does the method do?	Required
Prerequisites	What are the prerequisites for executing this method?	Optional
Result	What is the result of executing this method?	Optional
Parameters	Which parameters are transferred?	Required
Exceptions	What exceptions exist?	Optional
Notes	What special features exist for certain topics (commits, performance, buffering using internal tables, and dialog functions)?	Optional

For more information on documentation standards, see alias **S&G** in SAPNet.

1.15 Creating Documentation for SAPEasy

You must provide documentation describing your new Add-on to the appropriate implementation projects. See alias **sbo-support** in SAPNet.

2 Programming an External Client using the Data Interface API

2.1 Logon Procedure

In SAP Business One, the company object represents a company database for SAP Business One and is the highest object in the object-hierarchy. This means that the company object controls access to SAP Business One and logon takes place using this object.

2.1.1 The Company Object

The company object is an instance of the class SAPbobsCOM.Company. The client program has created a company object for you to logon to SAP Business One for the first time:

```
Public vCmp as SAPbobsCOM.Company
Set vCmp = New SAPbobsCOM.Company
```

You then need to set the following company object properties (connection parameters): *Server, UserName, Password, language,* and *CompanyDB* , for example:

```
vCmp.Server = "My_DB_Server"     'Name of the MS SQL DB Server
vCmp.CompanyDB =  “My_Company“   ‘Enter the name of your company
vCmp.UserName = "dagobert"
vCmp.Password = "secret"
vCmp.language = SAPbobsCOM.ln_English
```

Set the *Server* property to the name of the MS SQL database server.

Set the *CompanyDB* property to the name of the company you want to work with.

The properties *UserName, Password,* and *language* must contain the name of the SAP Business One-User, the user’s password, and the language for the session to be opened. Logical constants exist to specify the logon language. They are members of the enumeration BoSuppLangs and are described in the language property documentation for the company object.

To connect to the company database, you call the company object `Connect()` method:

```
Dim l as Long
l = vCmp.Connect()
```

If the connection to the database was successful, the method returns 0, otherwise it returns a negative value. You can obtain a more extensive error description using the company object method `GetLastError` (see Section 2.1.4).

To close the connection, you call the company object `Disconnect()` method:

```
vCmp.Disconnect
```

2.1.2 Accessing Business Data

Having established a connection to a SAP Business One Company database, you can access the data using the business objects contained within the SAPBobsCOM component.

To create a business object and to get a reference to it, you use one of the company object methods `GetBusinessObject`(object as `SAPBobsCOM.BoObjectTypes`) or `GetBusinessObjectFromXML`(Filename as String, index as Long).

`GetBusinessObject` always creates a new, empty business object. Before any business data can be retrieved or manipulated using this object, you need to set appropriate object properties. You must set the parameter objtype to the type of business object to be created (for the values defined, see the documentation. They are defined as members of the enumeration `SAPBobsCOM.BoObjectTypes`).

In contrast, `GetBusinessObjectFromXML` creates a new business object with predefined properties from an XML file contents. The document structure of the file must be be the same as for the concrete business object saved using the `SaveXML` method. For more information, see the online documentation for the company object method `GetBusinessObjectFromXML`.

The following example refers to the Business Partner's object:

```
'Declare a suitable reference variable:
Dim vBP As SAPbobsCOM.BusinessPartners
'Get a new business partners object
Set vBP = vCmp.GetBusinessObject(oBusinessPartners)
```

Business objects have methods for creating (Add), changing (Update), deleting (Remove), and reading a data object of this kind (GetByKey). For more information about these and other methods of accessing or changing business data, see the online documentation for the relevant business object.

Example - How to create a new customer in the company database:

```
Dim ret As Long
Dim errmsg As String, _
        bpCardCode as String

vBP.CardCode = "C00003"
vBP.CardName = "Ronald G. Tailor"
vBP.CardType = "C"
ret = vBP.Add()                                    ' ret = 0 if and
only if Add() was successful
vCmp.GetNewObjectCode bpCardCode  ' Here we want to know the key of the
last
                                                                    '
modified object. GetNewObjectCode is a
                                                                    '
standard method of the company object
If ret Then
    vCmp.GetLastError ret, errmsg
    msgbox errmsg
End If
```

Example - Reading data for a specific customer with a dedicated key:

```
Dim bpCardeCode as String, _
        bpCardName as String, _
        bpCardType as String
If vBP.GetByKey("C00003") = True Then
           bpCardCode = vBP.CardCode
           bpCardName = vBP.CardName
           bpCardType = vBP.CardType
Else
     msgbox "No matching customer record was found!!!"
End If
```

2.1.2.1 Special Objects for Data Access

In general, business data within an SAP Business One company database is accessed using predefined business objects contained within the SDK. The reasons for this are:

1. The objects perform authorization checks.
2. Any data access using business objects is platform and release-independent. An external client program that accesses data exclusively using the predefined business objects does not need to be adapted should the SAP Business One system be upgraded to a higher release version or support package level.

There may be situations where you cannot reach the desired business data using a business object, or you may want to access entire lists of business objects. In such situations, you can use two special objects, the RecordSet object and the SBObob object. These objects are described in Sections 2.3.1 and 2.3.2.

2.1.3 Transaction Handling

When a data operation is performed on a business object a transaction is started. If the operation is successful, then a *Commit* is issued and the data is saved. If the operation fails, then a *Rollback* is issued and the data is discarded.

This kind of transaction handling is sufficient as long as only one business object is being modified.

If you want to perform a consistent transaction that contains changes to more than one business object, you must specify the start and end of the transaction using the company object methods `StartTransaction()` and EndTransaction(endType as SAPBobsCom.BoWfTransOpt).

2.1.3.1 StartTransaction

Use this method to start a “global” transaction that can contain multiple changes to several business objects.

When you call `StartTransaction`, all business object changes that are issued after this call and before a following call to the method `EndTransaction` (see 2.1.3.2), belong to one logical unit of work.

If one of the business object changes fails during any process, the transaction ends and a rollback is issued.

If the changes are successful, you **must** use `EndTransaction` to run the commit, to dequeue the locked records and to allow other users to access them.

2.1.3.2 EndTransaction

Use this method to mark the end of a global transaction opened by `StartTransaction` (see 2.1.3.1.).

You must pass a parameter to `EndTransaction` to indicate whether the changes contained within this transaction are to be committed or rolled back:

`EndTransaction(endType as SAPBobsCOM.boWfTransOpt)`

Possible values of `endType`:

`SAPBobsCOM.wf_Commit`: Commit transaction

`SAPBobsCOM.wf_Rollback`: Discard all changes contained in the transaction

2.1.3.3 Accessing User-Defined Fields

SAP Business One allows you to define additional fields for the business data categories. These fields are called user-defined fields. They are not affected by any release or support package level upgrade.

You can access the contents of user-defined fields using the `UserFields` object. Unlike business objects, you cannot create this object with the `GetBusinessObject` method. All business objects that can be extended with user-defined fields have a `UserFields` property. This property is read-only, and contains a reference to a `UserFields` object that grants access to the contents of the user-defined fields for that business object.

Example - How to access the user-defined fields for a given business partner

Preconditions for this example:

- `vBP` has been already defined as a reference variable of type `SAPBobsCOM`.BusinessPartners and points to an existing object)
- `TxtValue` is a Textbox-Control defined at design time
- `LblName` is a Label-Control defined at design time

```
Dim oUserFields as SAPBobsCOM.UserFields, _
         ii as integer
Dim oFields as SAPBobsCOM.Fields
Dim oField as SAPBobsCOM.Field

Set oUserFields = vBP.UserFields
Set oFields = oUserFields.Fields

For each oField in oFields
       If ii > 0 Then
          AddUF
       End If
       TxtValue(ii) = oField.Value
       LblName(ii).Caption = oField.Name
       ii = ii + 1
Next oField

Sub AddUF
 ‘ Adds a new label and a new textbox control to the actual form
    Dim newnum As Long

    newnum = LblName.Count
    Load LblName(newnum)
    Load TxtValue(newnum)

    LblName(newnum).Left = LblName(newnum - 1).Left
    TxtValue(newnum).Left = TxtValue(newnum - 1).Left

    LblName(newnum).Top = LblName(newnum - 1).Top +  _
       LblName(newnum - 1).Height + 5
    TxtValue(newnum).Top = TxtValue(newnum - 1).Top + _
       TxtValue(newnum - 1).Height + 5
End Sub
```

The SAPBobsCOM.UserFields object has a Fields property that points to a SAPBobsCOM.Fields object. This object is a collection of several SAPBobsCOM.Field objects.

2.1.4 Error Handling

When an error occurs in SAP Business One during data access, you can call the company object method `GetLastError` (`errCode` as Long, `errMsg` as String) to get the return code and error description.

You have to call the method as soon as the error occurs. Once you have called subsequent methods, the error information is lost.

If no errors occur, `errCode` contains the value 0 and `errMsg` is an empty string. Otherwise, `errCode` contains a value other than 0 and `errMsg` contains the error description.

2.2 Data Interface API Business Objects

For an overview of the business objects in the DI API, see the DI API help file.

2.3 Data Interface API Raw Data Access Objects

Raw Data Access (RDA) means accessing data using SQL statements. This access method does not take into account security factors such as authorization or business objects. You should use this method of access with care.

2.3.1 The RecordSet Object

2.3.1.1 Introduction

The RecordSet object is a special object for generic access to existing database tables in the SAP Business One company database. The method DoQuery can run any valid SQL command against the database.

Starting with SDK 6.5, the first command word in an SQL statement **must** be one of the following:

- SELECT
- DELETE
- UPDATE
- INSERT

The following command words are **not allowed** as the first word in an SQL statement:

- EXECUTE
- CREATE

This has the following consequences:

1. DB tables can be accessed directly
2. Arbitrary inserts, updates and deletes can be carried out without authorization or business logic checks

The structure of DB tables can change during release or support package level upgrade; the coding of client programs using the `RecordSet` object may have to be invalidated after an SAP Business One release or support package level upgrade

Possible application scenarios include access to DB tables that do not belong to the SAP Business One standard because no corresponding business objects exist.

The `SBObob` object (see 2.3.2) contains several data access methods that are not possible when using standard business objects. These methods always return a `RecordSet` object.

2.3.1.2 Usage

The `RecordSet` object contains a result set which consists of one ore more identical rows. The object has special methods for processing the data contained in the result set:

`MoveFirst`	Moves to the first row of the result set
`MoveLast`	Moves to the last row of the result set
`MoveNext`	Moves to the next row of the result set
`MovePrevious`	Moves to the previous row of the result set
`SaveXML`	Writes the result set with XML format into an ASCII file

Additional properties for processing the resulting data:

`Bof`	Returns a Boolean value. If true, the actual row index points to the first record set row
`Eof`	Returns a Boolean value. If true, the actual row index *points behind* the last record set row
`Fields`	This property returns a `SAPBobsCOM.Fields` object for accessing fixed columns in a row
`RecordCount`	Returns the number of columns of the result set as a long value

For more information about the `RecordSet` object and its methods and properties, see the online documentation for the SAP Business One SDK Data Interface API.

2.3.2 SBObob Object

The `SBObob` object is an accessory that enables the user to get information from the company database using methods that are not supported by ordinary business objects such as `BusinessPartners`, `Documents`, and `StockTaking`.

Its methods for read access to business data return a `RecordSet` object because, in most cases, these methods select several data records of identical structure.

For example:

- Method `GetBPList` - retrieves a list of business partners defined in the company
- Method `GetUserList` - retrieves a list of users defined in the company

3 Programming an Additional Screen using the User Interface API

Note that the User Interface API has been delivered with the Software Development Kit since version 6.2.

3.1 Establishing the Connection to SAP Business One

Use the Application instance to use its containers (Menus & Forms), event manipulation and property settings.

In order to connect to COM UI, you must supply a connection string to the Add-on.

The connection string provides the Add-on with essential information regarding the SAP Business One application it is connected to.

- **Development Mode**: During development you will receive the connection string from SAP. For example, the connection string may like this:
  0030002C0030002C00530041005000420044005F004400610074006500760 02C0050004C006F006D0056004900490056
  When you are developing you should supply the string to your code.
  This is done by copying the string into VB environment as shown:
  Project → Project Properties → Make tab → Command Line Arguments.
- **Customer Mode**: When your Add-on is installed on your customer's desktop the Add-on will receive the connection string from SAP Business One.
  When SAP Business One launches an Add-on, it supplies the connection string as a command line parameter (argument).

```
CAUTION: The development connect string may be subject to change!
sample coding:
'Declare a new instance of the SboGuiApi object
Dim oSboGuiApi As New SAPbouiCOM.SboGuiApi
Dim oApp       As New SAPbouiCOM.Application
'Variable for the connection string
Dim sConnectionString As String

    'Get the connection string
    #If Release = True Then
        sConnectionString = Split (Command)(0)
    'SBO is running the addon exe with a command line parameter,
    'which is the connection string
    #Else
        sConnectionString = "String supplied from SAP"
    'connection string supplied from SAP for development only
    #End If
```

3.2 Creating the Form

The SAP Business One SDK UI API **Form** object represents a Form in SAP Business One.It is uniquely identified by its `InstId` field (form pointer address in the application in current life cycle). The **Form** object holds all data of the referenced form that is open to manipulations through the API (except the identifier).

```
Dim oForm As SAPbouiCOM.Form

Set oForm = oApp.Forms.Add("myForm" & oApp.Forms.Count)

oForm.Title = "Hello World"
oForm.Visible = True
```

3.3 Creating Items on the Form

The SAP Business One SDK UI API **Item** object represents an Item in SAP Business One forms. It is uniquely identified by its item number and holds all data of the referenced form (except for the identifier and type).

Use this object to manipulate item properties such as size, string, location, etc.

```
Dim oItem As SAPbouiCOM.Item
Dim oTxt As SAPbouiCOM.StaticText

Set oItem = oForm.Items.Add("3", it_STATIC)
oItem.Top = 10
oItem.Left = 20
oItem.Width = 200
Set oTxt = oItem.Specific
oTxt.Caption = "Hello SDK 6.2 world "
```

3.4 Creating a Form Using Screen Painter

Using the SAP Business One SDK Screen Painter (from SDK 6.5) you can quickly and easily create forms with the same look and feel as those already available in SAP Business Business One. Screen Painter is a SAP Business One Add-on. It is started automatically by SAP Business One. For more information, see the Screen Painter help system.

3.5 User Interface API - Objects

For an overview of available objects see the UI API help file.

4 Installing and Registering an Add-on

You need to register your Add-on application to ensure that it runs automatically when SAP Business One is launched. Registered Add-on applications connect to SAP Business One via the UI API.

This section describes the steps you must follow to register your Add-on with SAP Business One.

For general information about SAP Business One Auto Start (from SDK 6.2), refer to:

- The UI API release note for your particular version
- Note 653022 in the SAP Service Marketplace

4.1 Overview

Installing and registering your SAP Business One Add-on consists of the following four steps:

- **Step 1 – Generating a Registration File** – you need to run Add-onRegDataGen.exe
- **Step 2 – Verifying the Installation Directory** – done by the Add-on installation routine
- **Step 3 – Copying the Add-on System Files** – done by the Add-on installation routine
- **Step 4 – Registering the Add-on** – done by the Add-on installation routine

4.2 Step 1 – Generating the Registration File

You need to generate an Add-on specific registration data file to run your Add-on together with SAP Business One. This file is generated once only, and must be shipped together with the Add-on. It contains essential information that is written to the registry during the installation process.

Procedure

To generate your registration file:

1. Run the tool: `AddOnRegDataGen.exe` (shipped together with the SDK UI API).
2. Enter: *Partner name, Add-on name,* and *Add-on exe file name*.
3. Choose *Generate file*.

4.3 Step 2 – Verifying the Installation Directory

The installation routine calls the function `GetInstallPath()`, located in the `SBOAddonReg.DLL`[3], and uses the URL of the Add-on registration file as the first parameter.

The function then returns the path of the Add-on installation directory to SAP Business One so that your Add-on executable can be found and automatically started.

`GetInstallPath` - Function Definition:

```
Function GetInstallPath(ByVal installDataFile$, ByVal outStr$, lLen As
Long) As Long
```

Parameters:

1. `installDataFile$` – full path of the registration data file (created in Step 1 Generate a Registration File).
2. `outStr$` - returns the full path to which you should copy your Add-on exe file.
3. `lLen` – the length of the `outStr$` parameter.

The value returned by the function is a long value.

If `lLen` is too small and `outStr$` can't contain the path string – the value returned would be the necessary length. In this case you should call the function again with the correct `outStr$` size.

Otherwise the function returns 0.

The call `GetInstallPath("E:\AddOns\Datev\BD_ADDONREG_SLD")` would return the path of the Datev Add-on installation directory for example, `C:\Program Files\SAP Manage\SAP Business One\AddOns\SAP\BD_Datev`

[3] SBOAddonReg.dll is shipped together with the UI API SDK

4.4 Step 3 - Copying Add-on Files

The installation program copies all Add-on files from the installation CD to the Add-on installation directory (determined by the previous step).

For example, all files for running the SAP Business One Datev Add-on will be copied to `C:\Program Files\SAP Manage\SAP Business One\AddOns\SAP\BD_Datev`

4.5 Step 4 – Registering the Add-on

The installation program calls `RegisterAddOn()`(part of `SBOAddonReg.DLL`), and transfers the Add-on Name as a parameter to register the Add-on with SAP Business One.

This function verifies that the Add-on executable is located at the installation path given in step 2. If the executable file is present, the Add-on will be registered in SAP Business One for automatic starting.

`RegisterAddOn` - Function Definition:

The function does not have any parameters and returns a boolean value that indicates success or failure.

The call to `RegisterAddOn(<BD_Datev>)`would register the SAP Business One Datev Add-on with SAP Business One and enable the Autostart function. Check the return value using for example:

```
If RegisterAddOn = 1 Then
        MsgBox "Add-on Was Registered Successfully"
    Else
        MsgBox "Failed To Register Add-on"
End If
```

4.6 Unregistering an Add-on

Calling the function `UnRegisterAddOn()` - (contained in `SBOAddonReg.DLL`), using the URL of the registration data file as the first parameter, will remove the Auto Start functionality and delete all files in the Add-on directory.

`UnRegisterAddOn` - Function Definition:

Unregisters an Add-on from SAP Business One.

```
Function UnRegisterAddOn Lib "SBOAddonReg.dll" (ByVal installDataFile As String) As Boolean
```

Parameter:

`installDataFile$` – full path of the registration data file

Calling `UnRegisterAddOn("E:\AddOns\Datev\BD_ADDONREG_SLD")` will remove the Auto Start for the SAP Business One Datev Add-on and delete all files from the Datev Add-on installation directory `C:\Program Files\SAP Manage\SAP Business One\AddOns\SAP\BD_Datev.` Check the return value using for example:

```
If UnRegisterAddOn(InstallStrFile) = 1 Then
        MsgBox "AddOn Was UnRegistered Successfully"
    Else
        MsgBox "Failed To UnRegister AddOn"
End If
```

5 Tips and Tricks

This section contains tips and tricks that may prove useful when developing your Add-on for SAP Business One.

5.1 Preventing Performance Problems when Filling Matrices

To prevent performance problems, use the **data source** (from SDK 6.2 UI API) to fill matrix object cells instead of filling the matrix cell by cell.

Example

[Visual Basic 6.0]

```
        'Define different user data sources
    Set us_col1 = frm.DataSources.UserDataSources.Add "us_col1",
SAPbouiCOM.dt_SHORT_TEXT, 10
    Set us_col2 =     frm.DataSources.UserDataSources.Add
"us_col2", SAPbouiCOM.dt_SHORT_TEXT, 10
    Set us_col3 =     frm.DataSources.UserDataSources.Add
"us_col3", SAPbouiCOM.dt_SHORT_TEXT, 10
    Set us_col4 =     frm.DataSources.UserDataSources.Add
"us_col4", SAPbouiCOM.dt_SHORT_TEXT, 10
    Set us_col5 =     frm.DataSources.UserDataSources.Add
"us_col5", SAPbouiCOM.dt_SHORT_TEXT, 10
    'associate columns with data sources
    Set col1 = mtx.Columns.Add("col1", SAPbouiCOM.it_EDIT)
    col1.DataBind.SetBound True, "", "us_col1"
    Set col2 = mtx.Columns.Add("col2", SAPbouiCOM.it_EDIT)
    col2.DataBind.SetBound True, "", "us_col2"
    Set col3 = mtx.Columns.Add("col3", SAPbouiCOM.it_EDIT)
    col3.DataBind.SetBound True, "", "us_col3"
    Set col4 = mtx.Columns.Add("col4", SAPbouiCOM.it_EDIT)
    col4.DataBind.SetBound True, "", "us_col4"
    Set col5 = mtx.Columns.Add("col5", SAPbouiCOM.it_EDIT)
    col5.DataBind.SetBound True, "", "us_col5"

    'Fills matrix
 For ii = 0 To 19
 For jj = 0 to colNum
      frm.DataSources.UserDataSources(jj).Value = "stam" & ii &&
jj
      Next jj
   'After setting all the relevant data sources and the line to
matrix
    mtx.AddRow
    Next ii
```

5.2 Using XML with LoadBatchActions to Create a Form

You can pass XML as input to the `LoadBatchActions` method to create forms. Note that you can also activate menu items at the same time.

Example

```
<menus>
        <action type="enable">
           <menu uid="1290" />
           <menu uid="1291" />
        </action>
        <action type="disable">
               <menu uid="6657" />
           <menu uid="6658" />
        </action>
</menus>
```

5.3 Creating Radio Buttons Using the UI API

The following sample code shows you how to program radio buttons using the UI API.

Example

(Example code in Visual Basic)

```
Dim optBtn As SAPbouiCOM.OptionBtn
          Dim oFrm As SAPbouiCOM.Form
           Dim oItem As SAPbouiCOM.Item
           Dim oUserdatasource As SAPbouiCOM.UserDataSource
           ' set oFrm = ..

           'Option 1
           Set oItem = oFrm.Items.Add("BD_rbRes", it_OPTION_BUTTON)
           oItem.Left = 240
           oItem.Top = 10
           oItem.Height = 16
           oItem.Width = 220
           Set optBtn = oItem.Specific
           optBtn.Caption = "Button One"
           Set oUserDataSource =
      oFrm.DataSources.UserDataSources.Add("BD_resDS", dt_SHORT_TEXT, 1)
           optBtn.DataBind.SetBound True, , "BD_resDS"

           'Option 2
            Set oItem = oFrm.Items.Add("BD_rbPost", it_OPTION_BUTTON)
            oItem.Left = 240
            oItem.Top = 30
            oItem.Height = 16
            oItem.Width = 220
            Set optBtn = oItem.Specific
            optBtn.Caption = "Button Two"
            oItem.Visible = False
            Set optBtn = oItem.Specific
            optBtn.GroupWith ("BD_rbRes")
```

6 Glossary

API

Application Programming Interface: Is the specific method prescribed by a computer operating system or by an application program by which a programmer writing an application program can make requests of the operating system or another application.

(D)COM

(Distributed) Component Object Model: A standardized foundation for the communication and integration of software components. Introduced by Microsoft.

DI API

Data Interface Application Programmers Interface: Is a collection of COM objects called business objects. These objects describe various methods for updating, retrieving, and manipulating data in the SAP Business One database.

ODBC

Open Database Connectivity: This is an API for database access introduced by Microsoft.

SAP Business One SDK

Software Development Kit to enhance and/or to build additional components for the SAP Business One System.

SAP Service Marketplace

SAP's worldwide information and communication center.

SQL

Structured Query Language: SQL is a language used to access relational database management systems.

UI API

User Interface Application Programmers Interface: Is a collection of DCOM objects that provide access to forms, controls within these forms, and menus within SAP Business One.

XML

Extended Markup Language: XML is a markup language that can be used to pass structured information between different Web based applications.