

# SAP ADOBE Forms

## Overview

Forms are used for mass printing in SAP systems. Besides using the printer for standard output you can also select the Internet (by using a generated HTML output), a fax, or e-mail as the output medium.

### Tools Delivered by SAP for Form Designing

- SE71 – Sapscripts
- SmartForms – Smart Forms (introduced in SAP Basis Release 4.6C)
- SFP – Adobe Form (As of SAP NetWeaver '04 )

As of SAP NetWeaver '04 (in SAP Web Application Server), you can use a new solution to create interactive forms and print forms for the optimization of your form-based business processes. This solution uses Portable Document Format (PDF) and software from Adobe Systems Inc. that has been integrated into the SAP environment



# Overview - Features

Create form templates for the layout that include logos or pictures

Edit forms online or offline

Forms can be filled in advance automatically with specific data from SAP applications and then sent to the correct recipients using secure methods

Automatic consistency checks for forms

Activate enhanced functions such as comments

Digital signatures and form certification

User-friendly tools reduce the time and costs associated with creating form layouts.

The usage of the PDF format means that forms retain their appearance regardless of the environment they are used in.

# Overview - Advantages Over Smart Forms/SAPscript

Adobe Livecycle Designer is an easy to use, flexible tool for designing forms

Full integration into the SAP development environments for Java and ABAP

Graphics (BMP, JPEG, GIF, PNG, EXIF) can be included into forms directly no conversion required

Objects (including texts) can be rotated

Different page orientations (landscape, portrait) are possible within one form

Graphical elements can be included in forms

Existing PDF or Word documents can be imported

Barcodes can be printed on all printers of types Postscript, PCL, PDF, or Zebra

Mailing and faxing is easier

Scenarios and integration into browser-based applications are possible (Web Dynpro for Java or ABAP)

# Overview - Adobe Designer: - Technical Pre-requisites

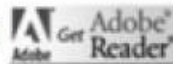
SAP 6.40 GUI patches are installed



## **SAP GUI for Windows**

- Make sure that Designer is installed and check on your hard drive. Default location for Windows: C:\Program Files\Adobe\Designer 7.1 (depending on version)

Adobe Reader 7.0 (incl. update to 7.0.8) - The most current version should always be used, in particular for interactive features. Check SAP Note 834573 for details.



## **Adobe Reader**

- used to be called Acrobat Reader



## **Microsoft Windows**

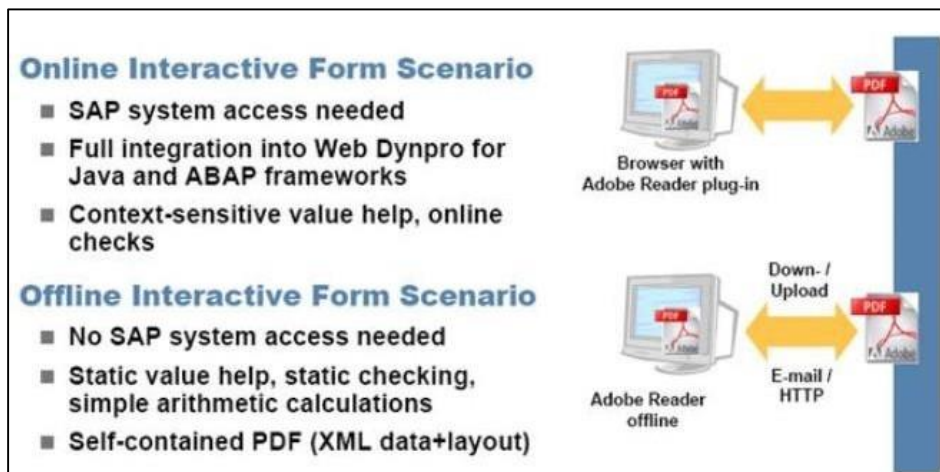


Microsoft Windows 2000 or higher

# Overview - Adobe Form – Types

There are two basic types of forms:-

- 1. Interactive Forms ( Basic scenarios )
  - Online
    - The user is logged on to the SAP system when he or she fills out the form.
  - Offline
    - The user is not logged on to the SAP system when he or she fills out the form. Once the form has been filled out, the user sends it to the issuer of the form, for example by e-mail. The SAP system of the issuer then extracts the data from the form.



# Overview

You can use the following development environments:

- Interactive Forms in Web Dynpro for Java in SAP NetWeaver Developer Studio
- Interactive Forms in Web Dynpro for ABAP in ABAP Workbench

These forms can contain dropdown menus, pushbuttons, text fields, and other elements that enable users to enter data.

The form is generated in PDF format, which, for example, can be displayed by the user in a browser.

The user uses Adobe Reader or Adobe Acrobat to fill out the form and saves the changes made to the form in XML format.

The SAP system extracts the data from the form and saves it to the database, where it can be processed further.

# Overview

There are basically two scenarios how PDF based forms can be used in an SAP system:

Integration into classical ABAP programs:

- This is typically the case when mass processing of forms is required, e. g. for printing, mailing or faxing them. Data retrieval, user communication, or result processing (like spool processing) is done with the logic of ABAP screens (using SAP GUI). Technically speaking, interactive scenarios are also possible with SAP GUI integration, but you would typically have a browser based UI for interactive scenarios

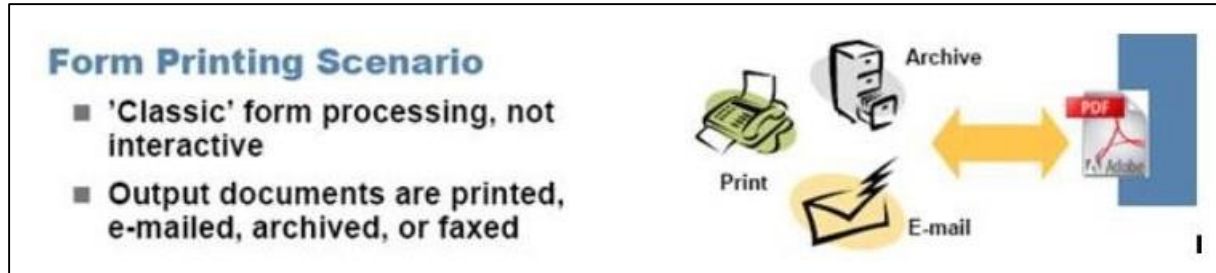
Integration into browser-based, interactive scenarios

- In interactive scenarios, individual forms are processed and displayed in a web browser. The user can then enter data into the form and trigger the further processing. You can use Java Web Dynpro or (starting with SAP NetWeaver 2004s) ABAP Web Dynpro.



# Overview

## Print Forms - Used normally for Printing/Fax/Email



- Forms for Printing, E-Mail, or Fax
- You can use the Form Builder (integrated into ABAP Workbench) to create
- PDF-based print forms that you can then print or send by e-mail or fax. When you create these print forms, you can rely on the tried and trusted principle of separate data retrieval and form layout processes.
- This enables you to make changes to either one of the processes, without affecting the other.
- PDF-based print forms can be used for the following: Order confirmations  
Invoices , Account statements , Checks etc

# Overview

## Online Interactive Form Scenario

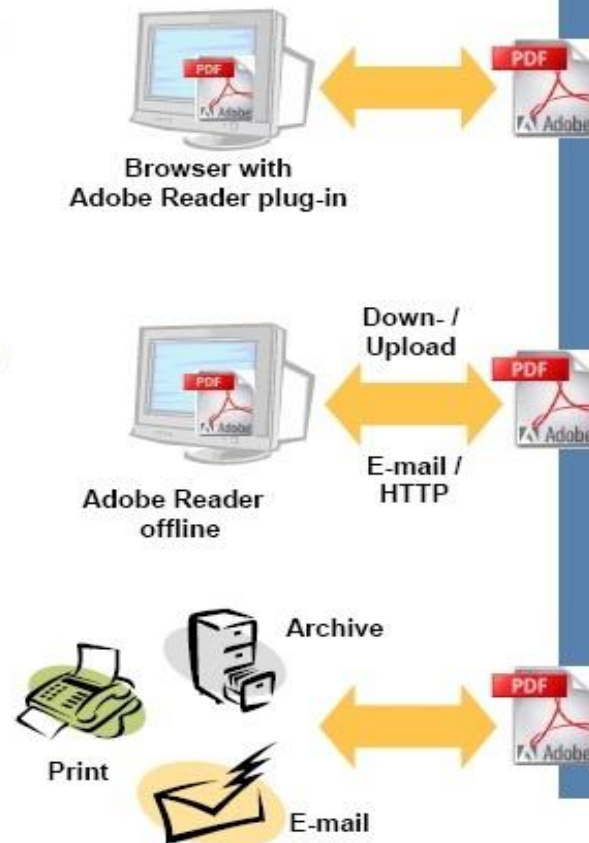
- SAP system access needed
- Full integration into Web Dynpro for Java and ABAP frameworks
- Context-sensitive value help, online checks

## Offline Interactive Form Scenario

- No SAP system access needed
- Static value help, static checking, simple arithmetic calculations
- Self-contained PDF (XML data+layout)

## Form Printing Scenario

- 'Classic' form processing, not interactive
- Output documents are printed, e-mailed, archived, or faxed



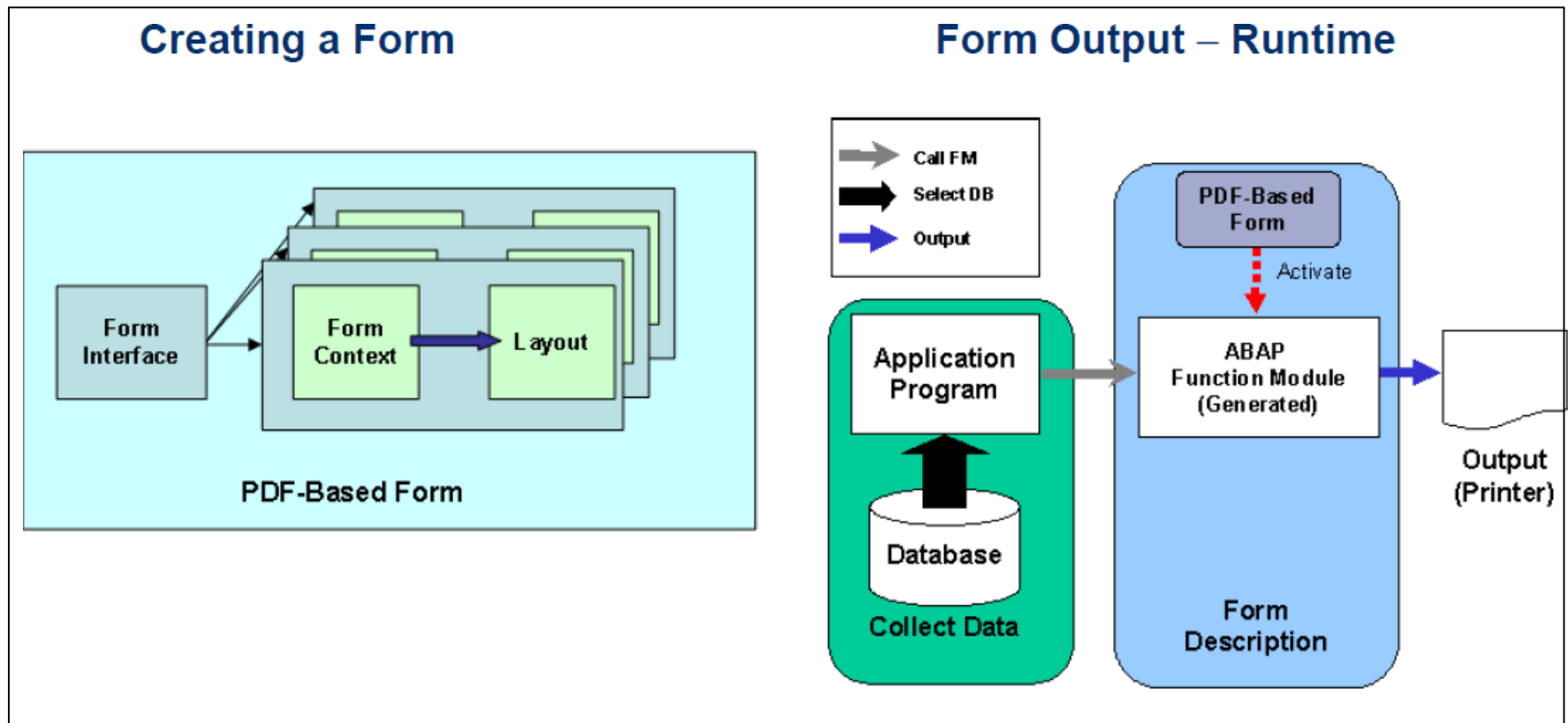
# Architecture - Structure

A PDF-base print form has the following attributes:

- **A form interface** that sends the application data to the form.
- **A form context** that contains the form logic. This logic controls the dynamic formatting of the form. For example, it enables variable fields to be displayed; it specifies that certain texts appear only under certain conditions (one text for a first warning and a different text for a second warning); and it can specify that invoice items can be processed repeatedly in a table.
- **A layout.** In the layout, you define how the output data is positioned, its appearance in graphics, and the design of the pages.

# Architecture

The following graphics show you the architecture that is implemented when you create and print a PDF-based form.



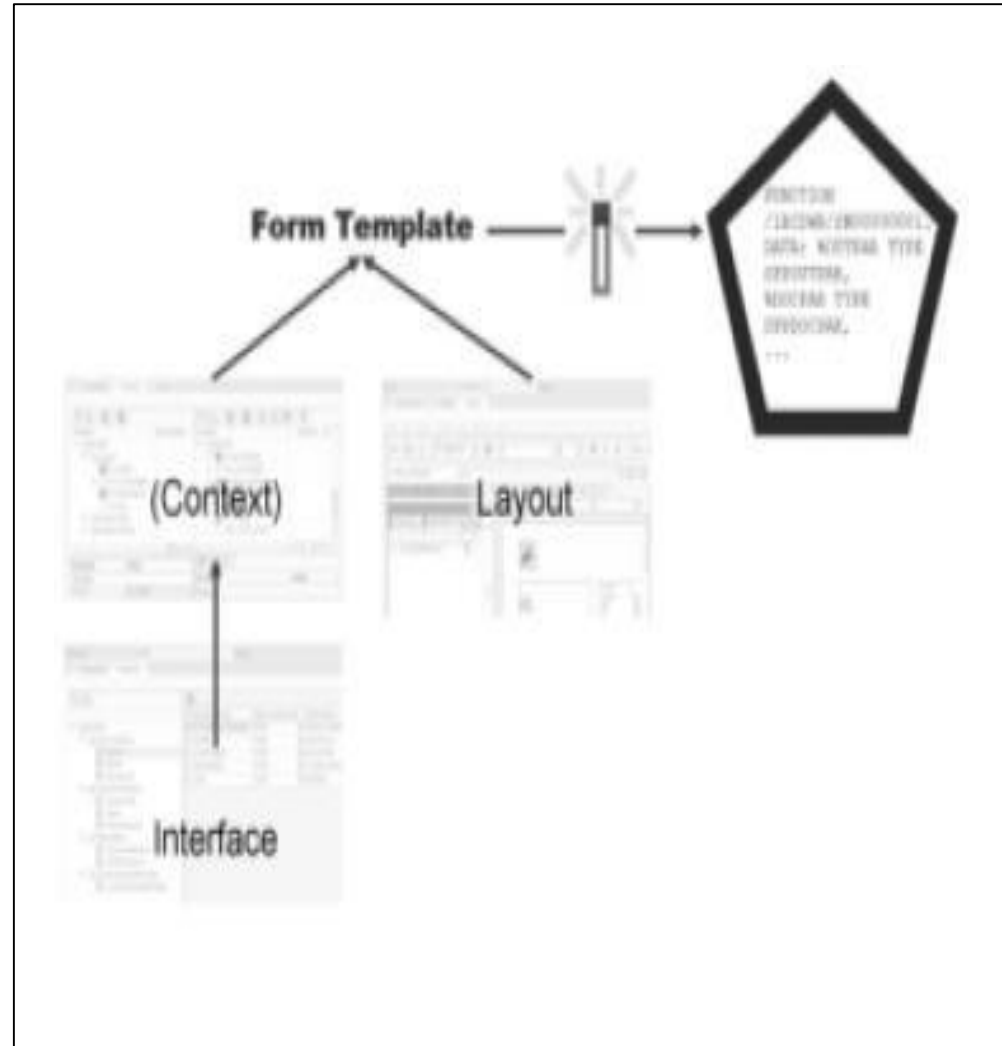
# Architecture - The Tools Involved (Design Time)

## 1. Interface (transaction SE80 or SFP)

- Reusable
- The interface defines which data a
  - Program can possibly pass on to a form.

## 2. Form template (transaction SE80 or SFP)

- Consists of a context and the layout.



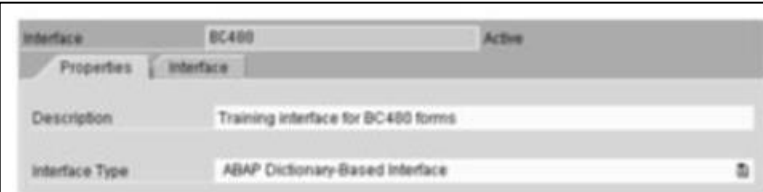
# Interface

Every PDF-based form needs to have an interface; it is the link between the ABAP program and the form.

The program can pass data to the form only if it is defined in the interface, and (most of) the dynamic data used in the form layout will be defined in the interface.

To access interface maintenance, use transaction SFP. Alternatively, use transaction SE80 and choose Other object.

## Types of an Interface



The screenshot shows the SAP SFP (Smart Forms) interface maintenance screen for interface BC400. The interface is active. The description is "Training interface for BC400 forms". The interface type is "ABAP Dictionary-Based Interface".

- **ABAP Dictionary based**
- **Smart Forms compatible**
- **XML schema based (mainly Web Dynpro)**

# Interface

The types “ABAP Dictionary based” and “Smart Forms compatible” are used for print scenarios.

“XML schema based” (which was introduced in SAP NetWeaver 2004s) is primarily used for Web Dynpro scenarios.

# Interface - Parts ABAP Dictionary Or Smart Forms

Properties Interface

BC480

Form Interface

- Import
- Export
- Exceptions

Global Definitions

Initialization

Currency/Quantity Fields

- ABAP types
- Dictionary types

Parameter Name	Type assignment	Type Name	Optional Flag	Pass Value	Default Value
/1BCDWB/DOCPARAMS	TYPE	SFPDOCPARAMS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
IS_CUSTOMER	TYPE	SCUSTOM	<input type="checkbox"/>	<input type="checkbox"/>	
IT_BOOKINGS	TYPE	TY_BOOKINGS	<input type="checkbox"/>	<input type="checkbox"/>	
IT_SUMS	TYPE	FLPRICE_T	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
IV_IMAGE_URL	TYPE	STRING	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
IV_SENDING_COUNTRY	TYPE	ADRC-COUNTRY	<input type="checkbox"/>	<input type="checkbox"/>	'DE'



# Interface - Form Interface

## Form Interface

- Parameter name
- For type assignment, TYPE is the only option (in interfaces that result from a Smart Form migration, you might also use LIKE.)
- Type name: You can enter ABAP types here (c, i, n, etc.) and Dictionary types (like data elements or tables).
- If you set the Pass Value flag, a copy of the parameter will actually be passed from the program to the form (not just the address). Such parameters can be changed in the interface coding; the original value remains untouched.
  - Parameters with this option checked will slow down performance, particularly in the case of large parameters (such as internal tables).

Note: The import parameters defined in the interface can be passed from the application program to the form at runtime, and vice versa for export parameters.

# Interface - Form Interface

A form interface of the type ABAP Dictionary has only one default import parameter (/1bcdwb/docparams of type sfpdocparams). It is used to determine a form's locale (language and country) and whether the form will allow interactive features.

Export parameters can be added only for those interfaces that are compatible with Smart Forms.

Exceptions that you declare in an interface can be raised in ABAP coding of a form. They are based on the traditional exception concept (not the class-based concept that was introduced in SAP Web Application Server 6.10).

You raise an exception as such: `RAISE <exception>.` (Alternative: `MESSAGE <message_type><message>(<message_class>) RAISING <exception>`

# Interface – Global Definitions

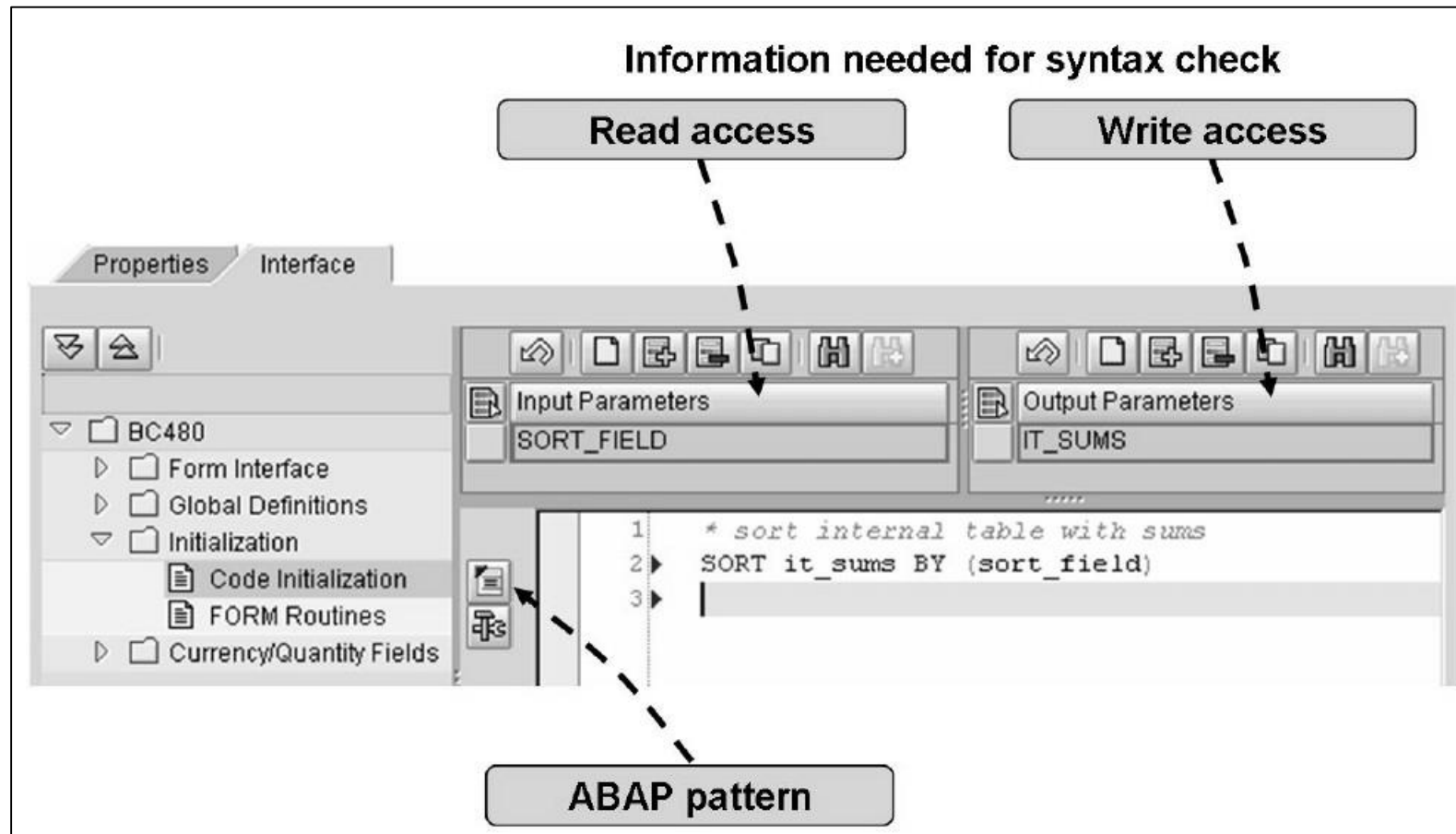
Global Definitions - There are also parts to an interface that are actually invisible from outside, that is, they cannot be accessed from the application program. Among them are the global definitions, initialization coding, and currency/unit fields.

Global fields: Global fields can be integrated in the form layout.

Field symbols: Field symbols might act as placeholders for variables. They are useful in dynamic programming and for speeding up the processing of internal tables.

Global types: If your global fields (or any fields that you might declare within ABAP coding) need types other than ABAP types (i, n, f, c, p, and so on) or Dictionary types, you can create local types in the editor that opens when you choose *Types*.

# Interface - Initialization



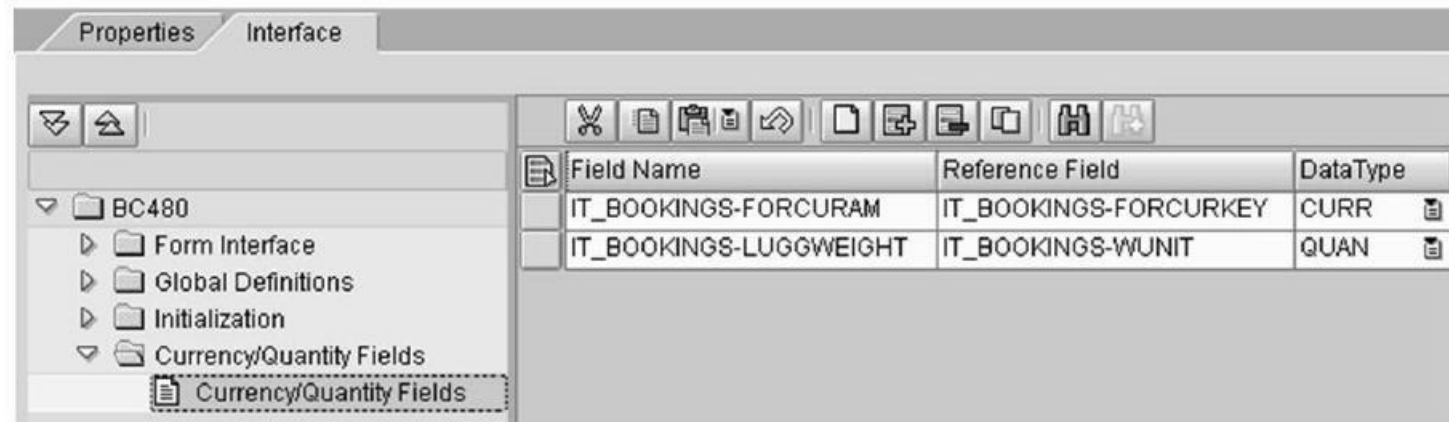
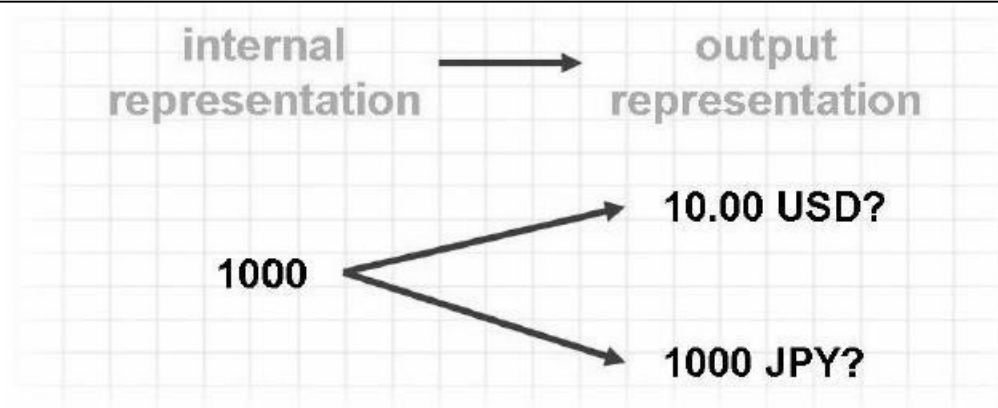
# Interface - Initialization

During initialization, data coming from the program can be changed before it is sent to the form. In forms that have not been migrated from Smart Forms, initialization is the only time when ABAP coding can be executed.

Even if the initialization coding makes use of form interface fields or global fields, you still have to make them known to the initialization coding. Enter those fields that you read from under Input Parameters, and those that you set under Output

In the initialization coding, you can call form routines that you have created in the interface: `PERFORM ...` Form routines make sense for coding that needs to be executed several times. You define form routines by using the ordinary ABAP syntax `FORM xyz... ENDFORM`.

# Interface – Currency/Quantity Fields



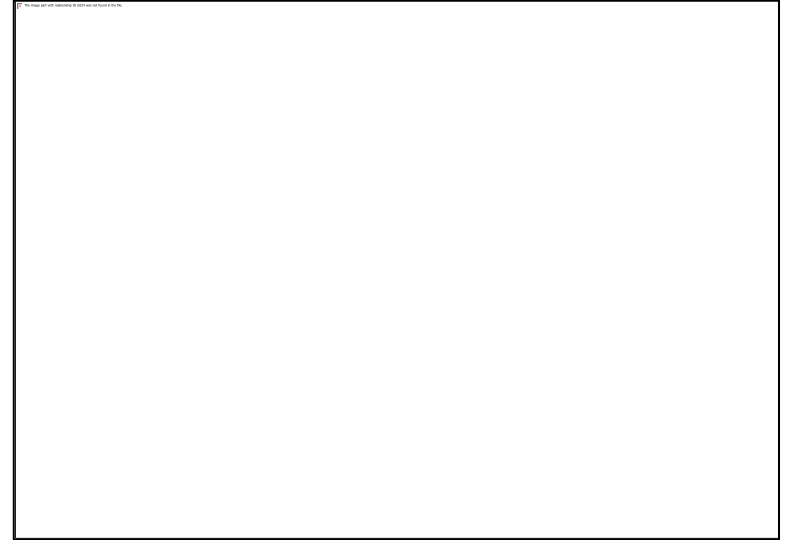
# Interface – XLS Schema



# Context

The context is essential, as it provides the source for data for a form. Apart from static elements, only those texts, fields, images, and so on can be included in the layout of a form that have been integrated into the form's context. However, the context should not be overloaded, as this will have a negative impact on printing performance.

It can be seen as a subset of the interface enriched with some form specific information.

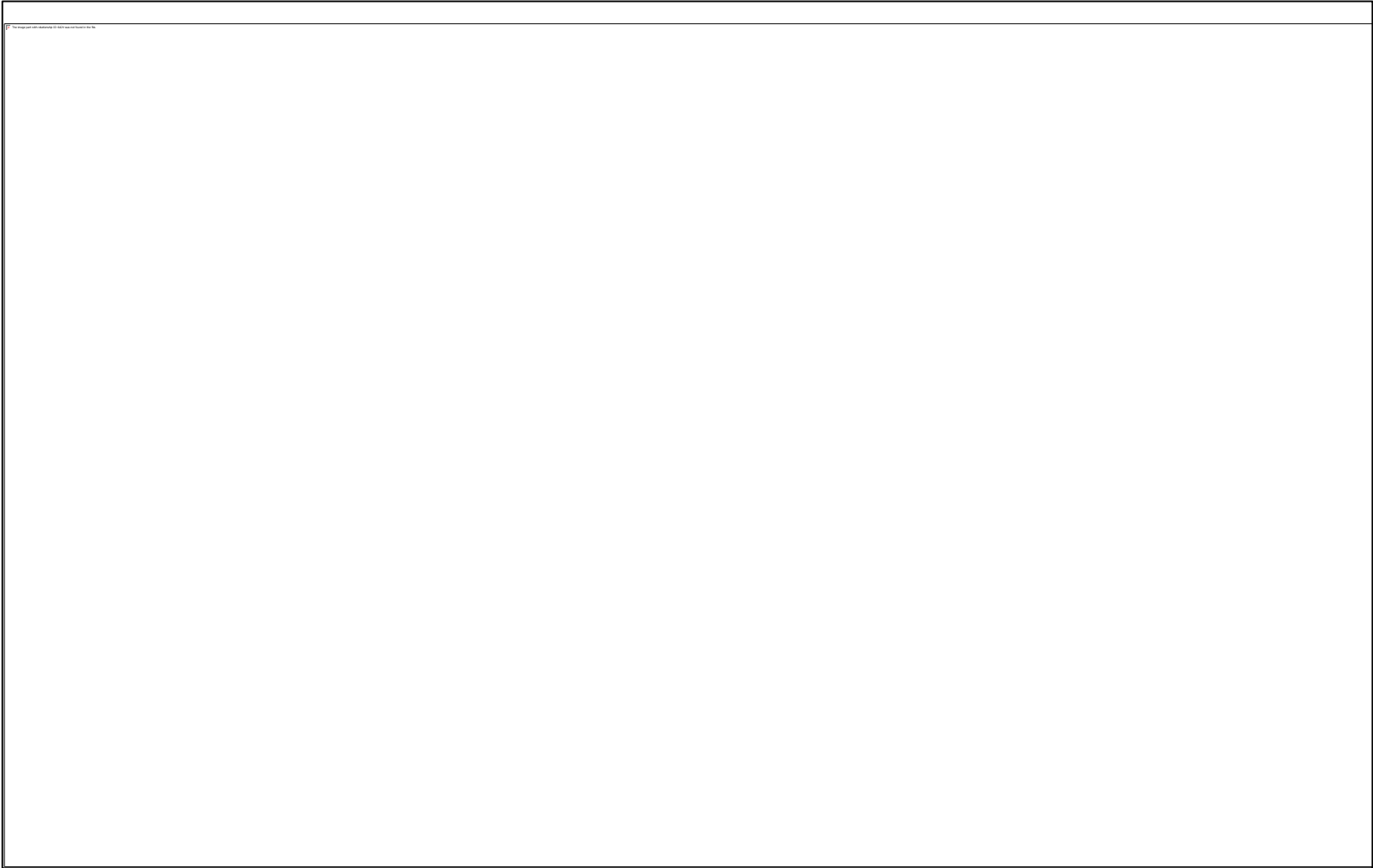




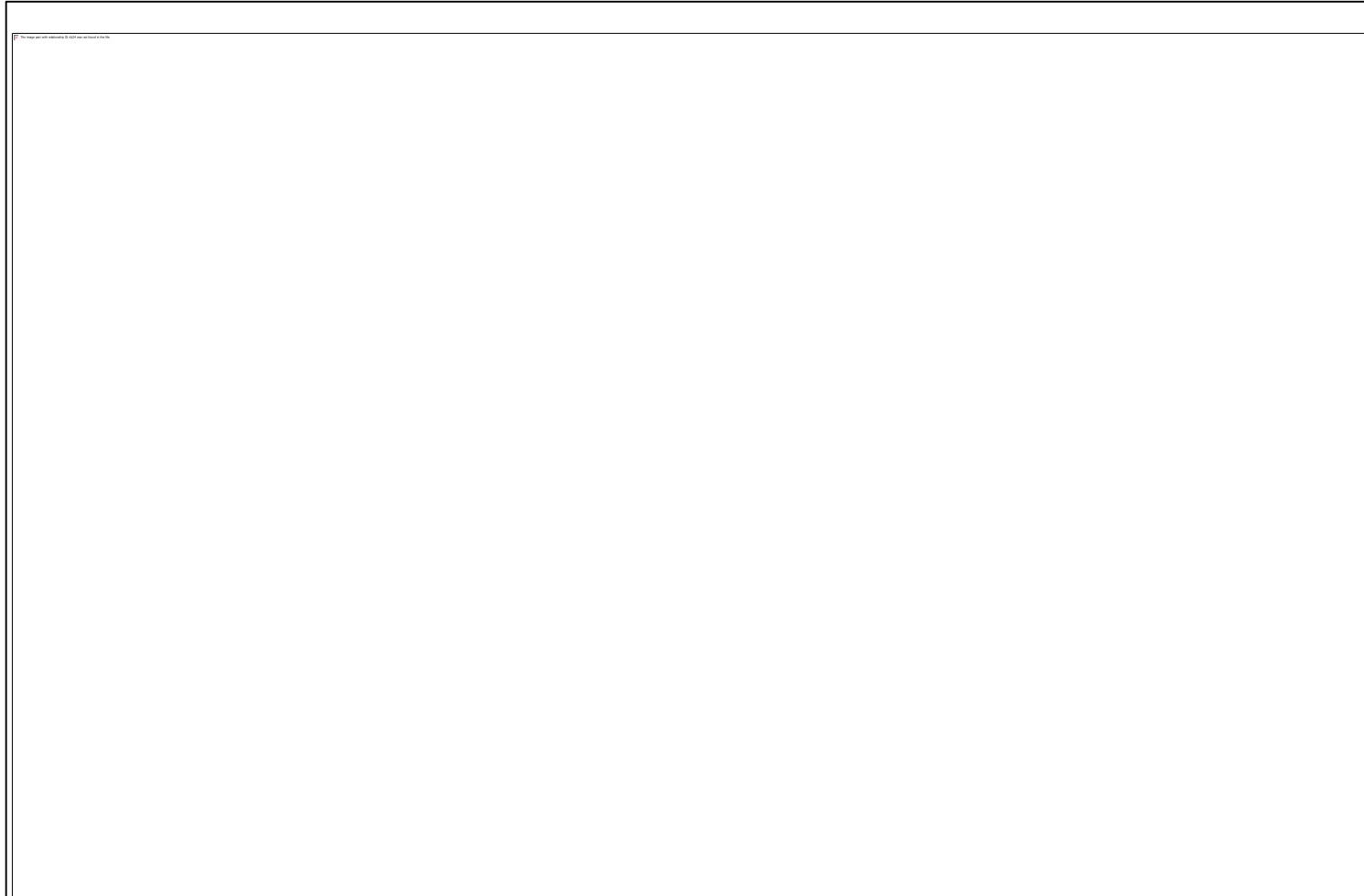
# Context - General Handling of the Context



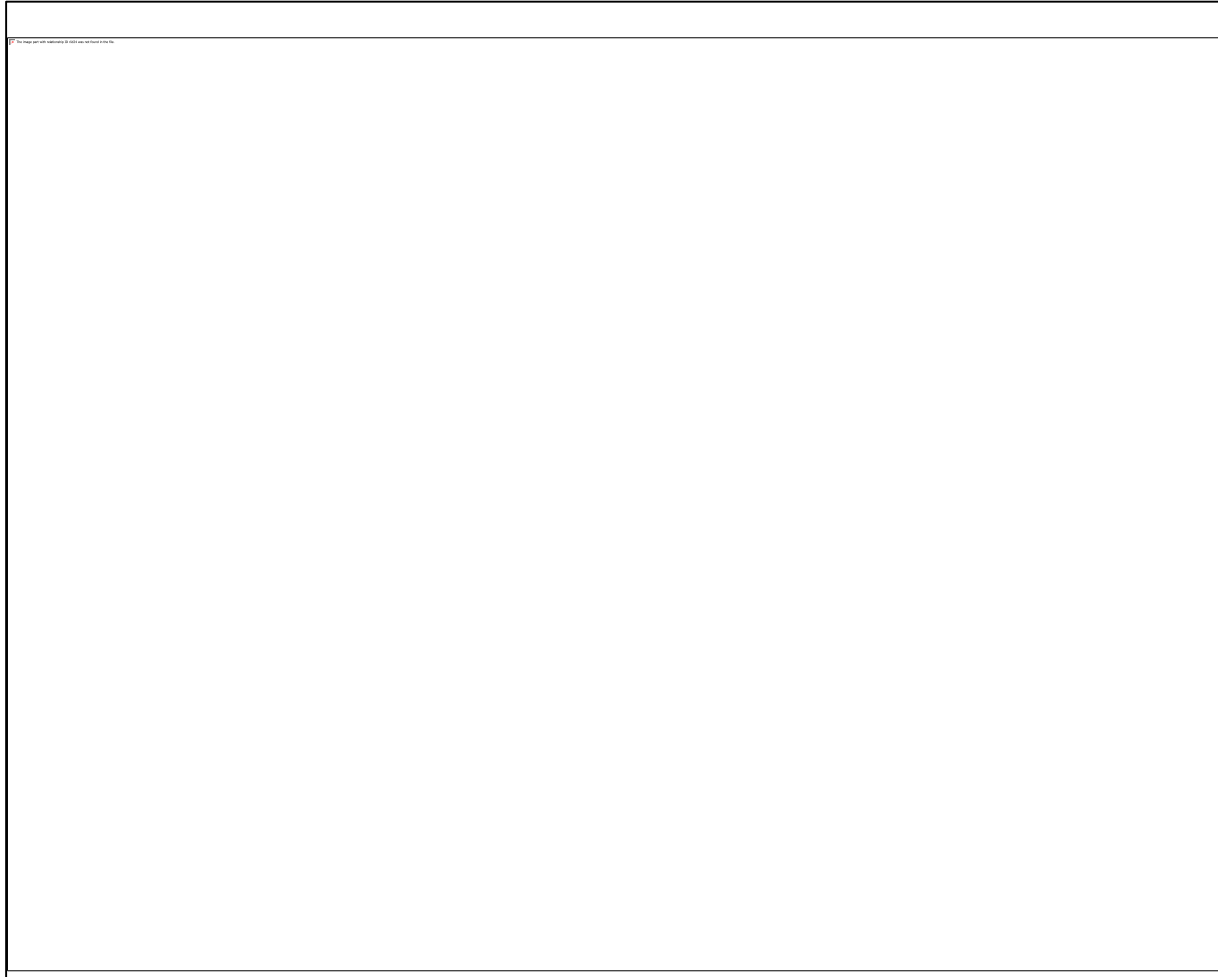
# Context - Using the Interface



# Context - XML Representations of Internal Tables



# Context - Internal Tables (Loops) in the Context



# Context - Conditions and Alternatives

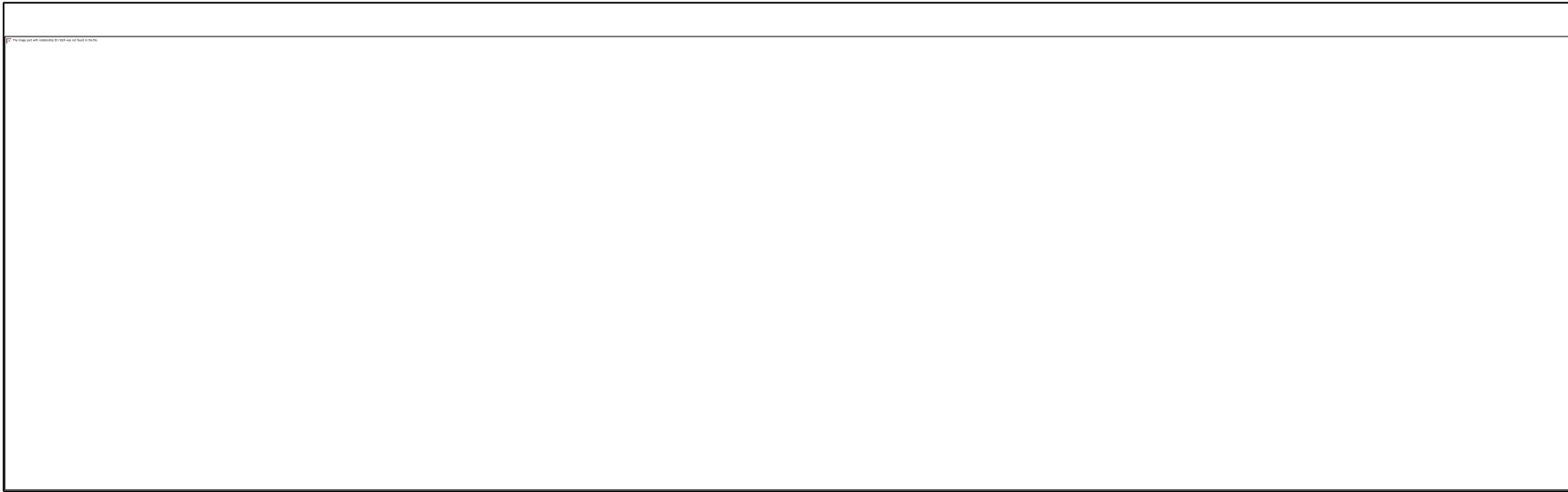
If a condition for a context element fails to be correct at runtime, the element field will not be part of the data stream.



# Context - Folders and Graphics

Folders make working with Adobe Lifecycle Designer easier, too, as they help you to organize your layout or nodes.

In the context, you can also define graphics. To do so, right-click on the top node of the context and choose *Create* → *Graphic*



# Context - Including Texts

In a form context, four types of long texts can be included:

- 1.Addresses from the Business Address Services
- 2.Text modules (Smart Forms texts)
- 3.Include texts (SAPscript texts)
- 4.Dynamic texts

Note: All these texts will automatically be converted by transaction SFP into a special format that can be evaluated by Adobe document services:XHTML.







# Context - Addresses

© 2014 Pearson Education, Inc. All rights reserved.

# Context - Text Modules and SAP script Texts



# Context - Text Modules and SAP script Texts

You need to create the texts only once and can then reuse them as required.

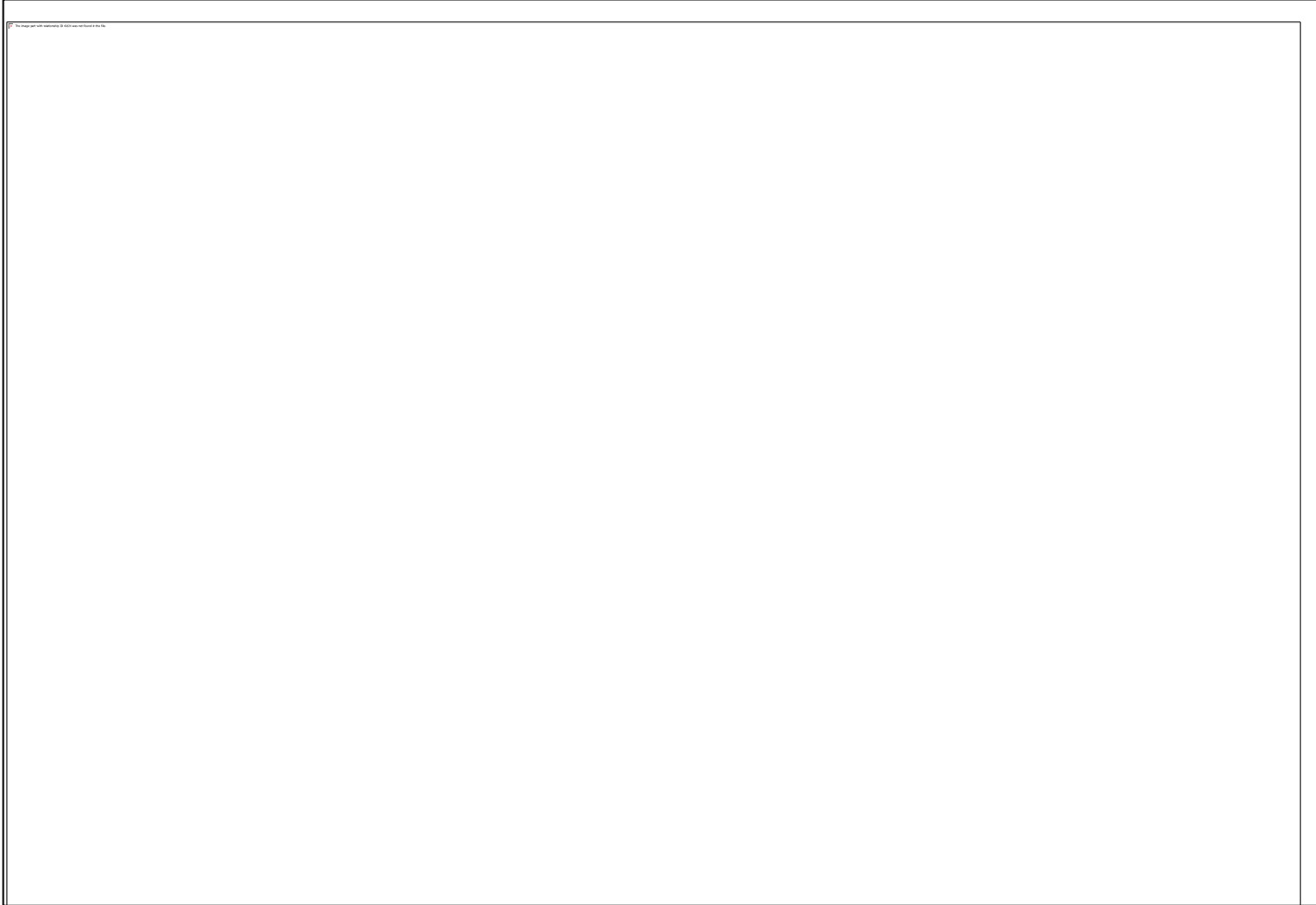
You make changes centrally only once without having to modify the actual forms.

Typical examples include headers (company address), footers (company information like board members and so on), and whole pages containing introductions or terms of trade.

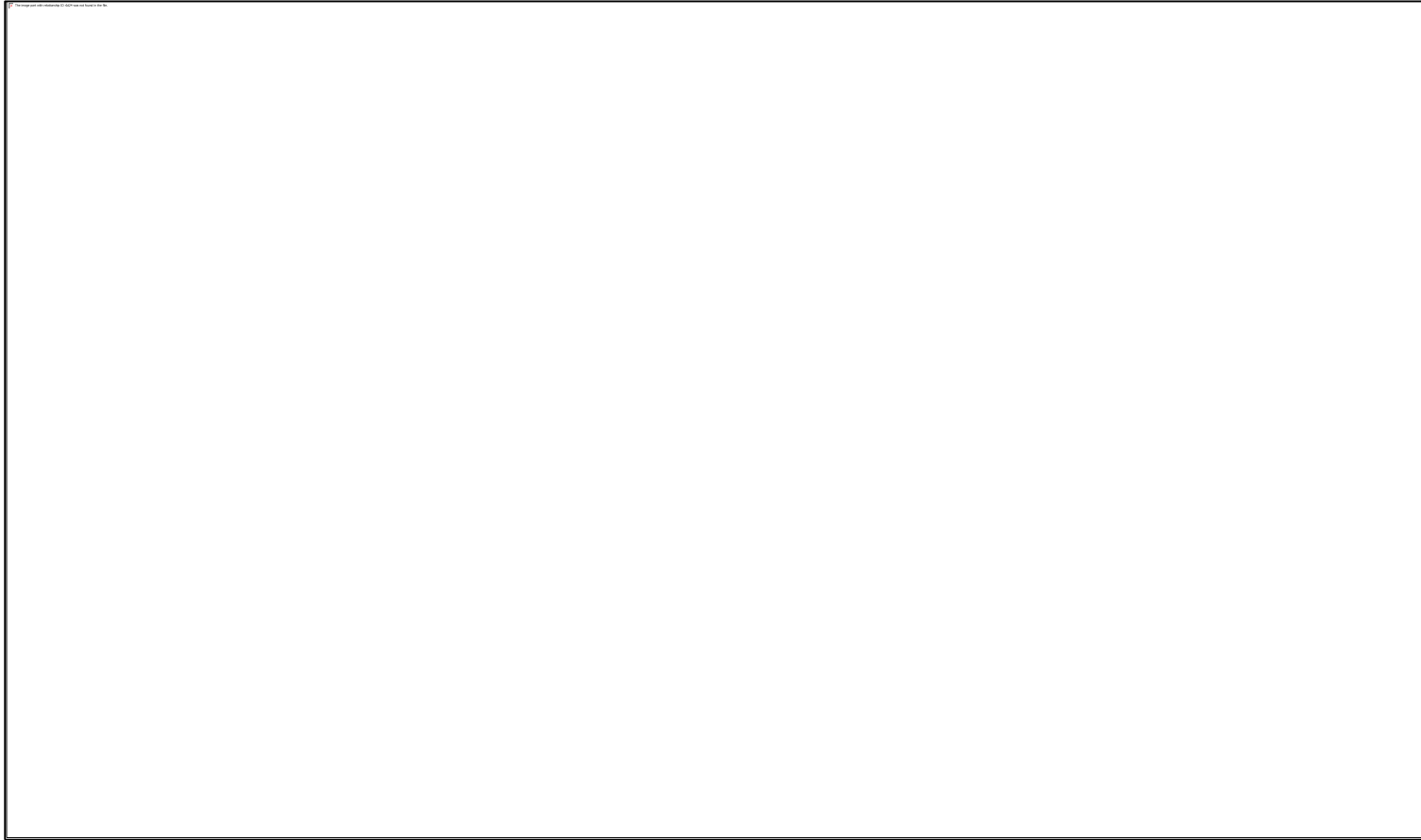
Use text modules of Smart Forms and also include texts (that is, SAPscript texts).

Run transaction SMARTFORMS (for text modules) or SO10 (for include texts

# Context - Creating Text Modules



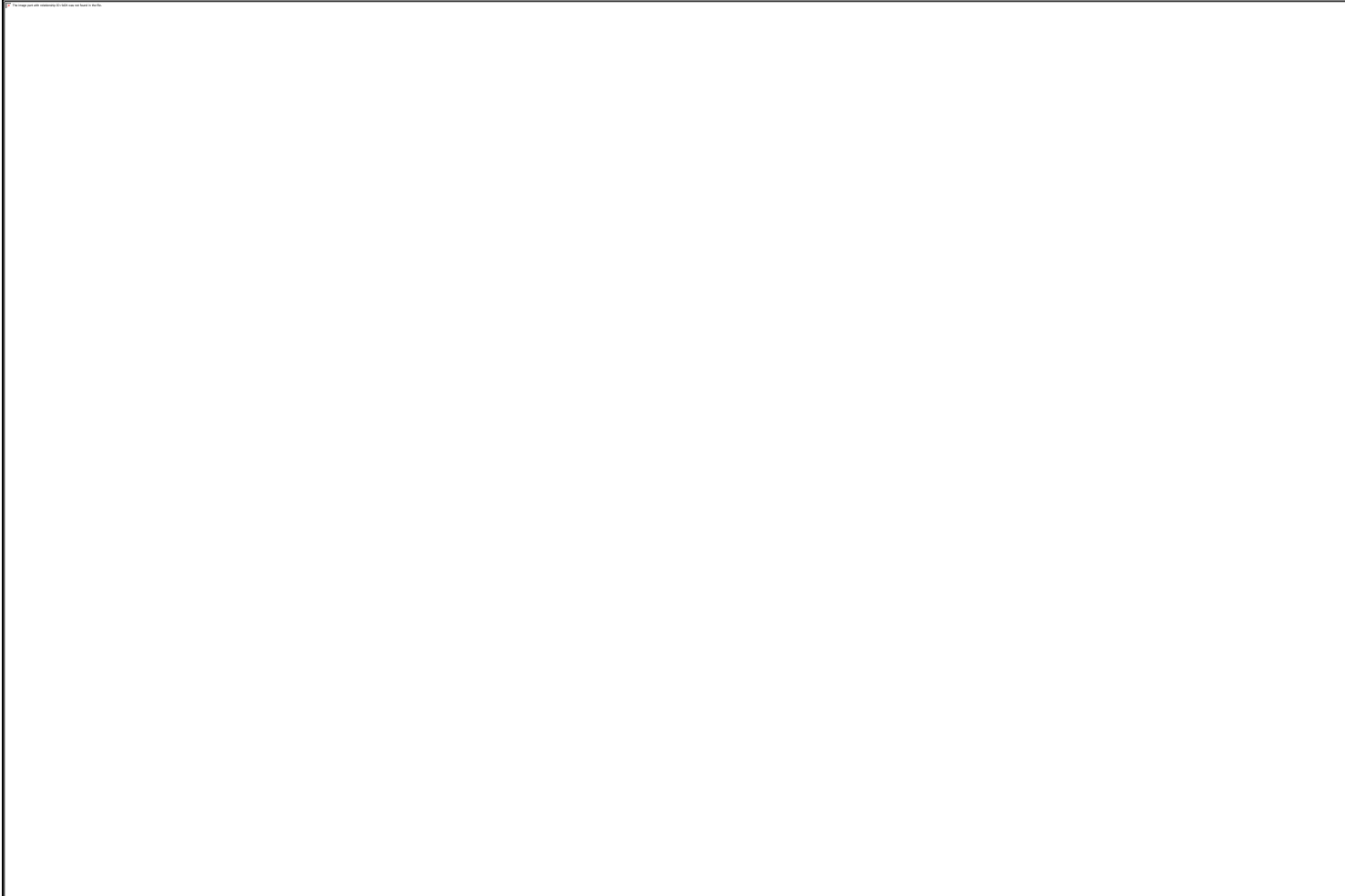
# Context - Including Text Modules



# Context - Including SAP script Texts



# Context - Including Dynamic Texts



# Designer - Adobe Lifecycle Designer: Overview

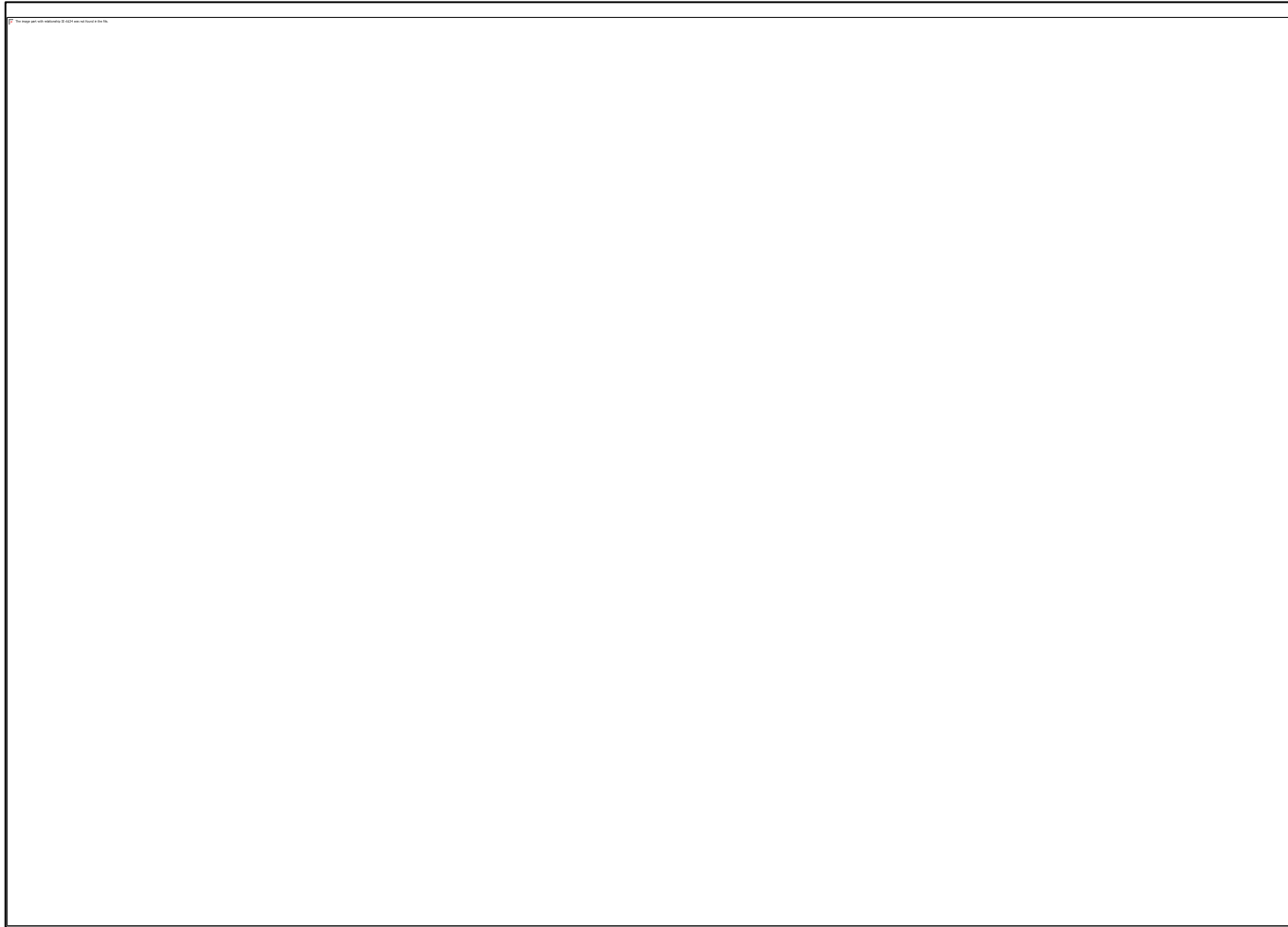




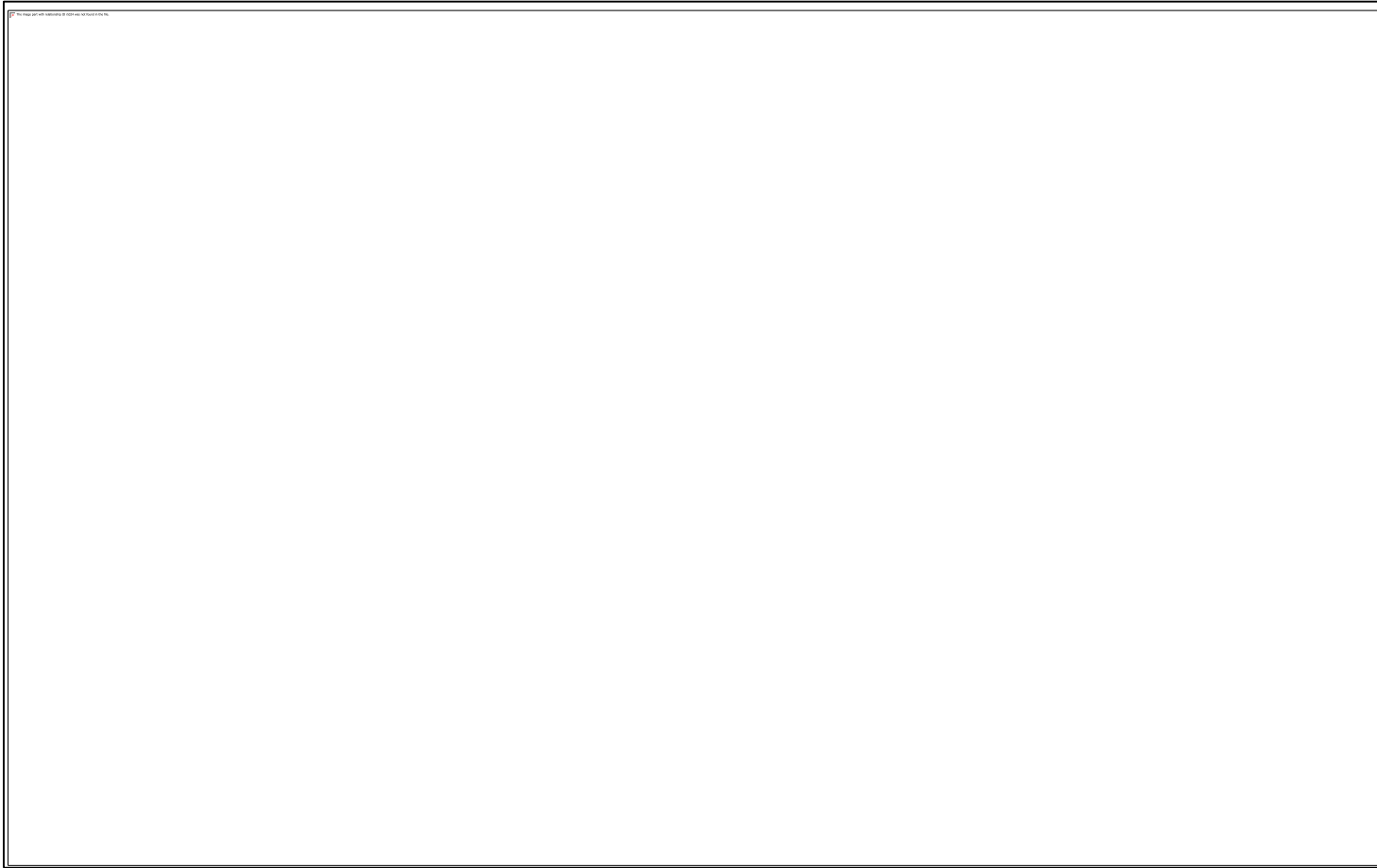
# Designer - Adobe Lifecycle Designer: Overview

1. In SAP NetWeaver 2004s, a pushbutton Layout was added to transaction SFP, which displays Designer in a full screen
  2. The Designer workspace consists of four main areas. All but the central one (the Layout Editor) can be closed by choosing Palettes → Workspace (Palettes → Manage Palettes in some versions).
  3. In the top area, the Script Editor can be displayed. It allows you to enter scripts for calculations. You can choose between JavaScript and Adobe's FormCalc.
  4. The subdivisions of the left and right areas are called palette windows with further subdivisions of palettes
- Note: You can always return to the standard by choosing Palettes → Workspace → Reset Palette Locations.

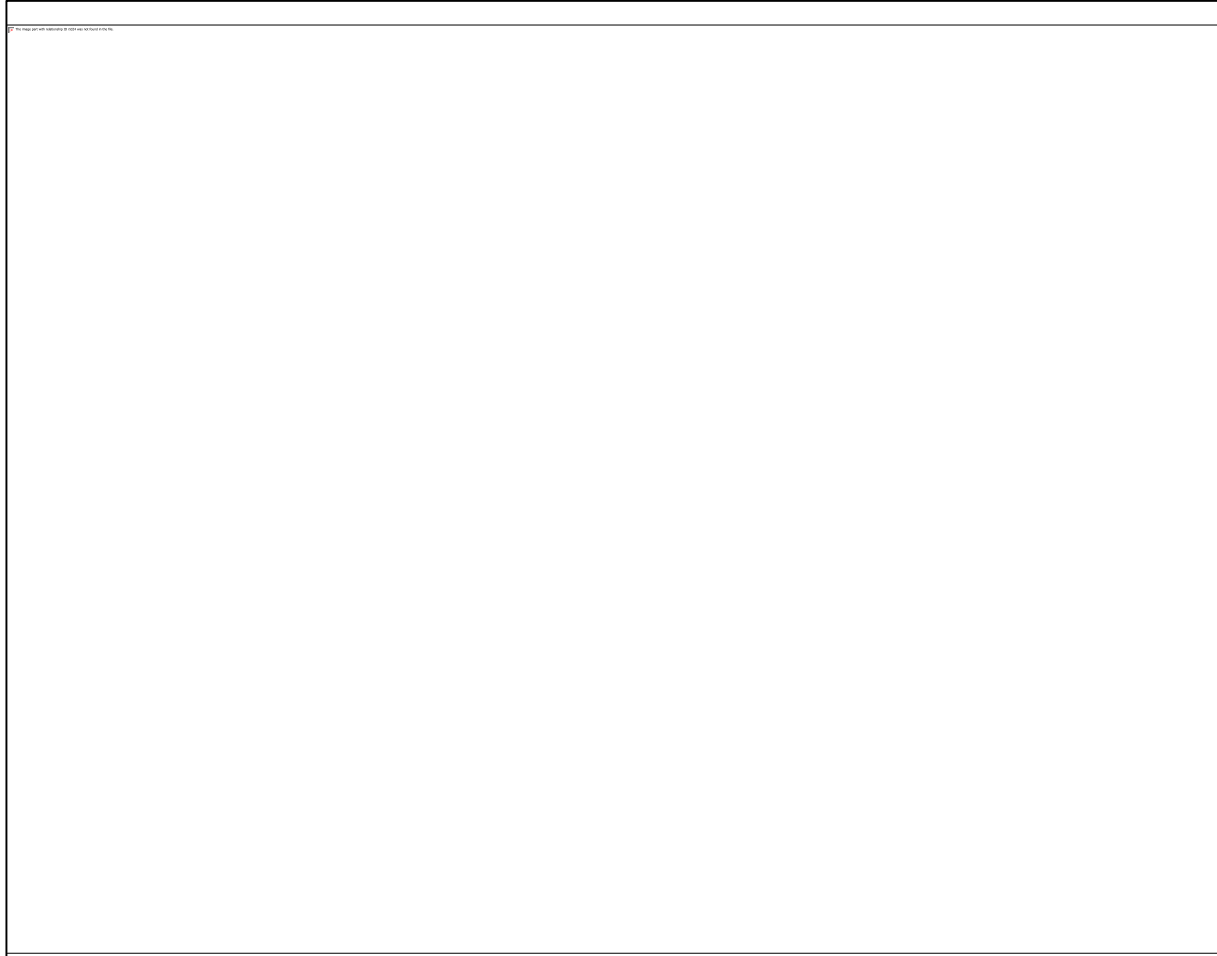
# Designer - Toolbars



# Designer - Palette Windows: Handling



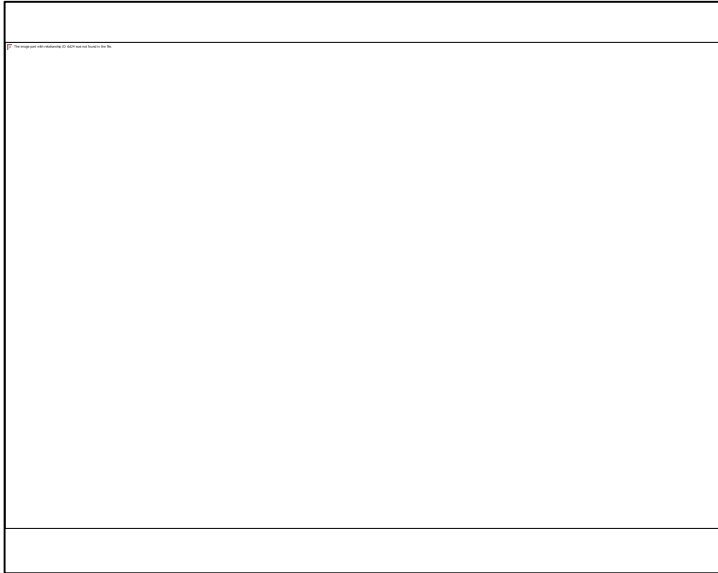
# Designer - Palettes: Overview



# Designer - The Hierarchy Palette



# Designer - The Layout Palette



1. You can position and size an object by clicking the resize handles of the Layout Editor and moving them. You can achieve the same by typing in the coordinates and the width/height in the Layout palette.

2. For dynamic elements (like dynamic texts that come via the application program) you can select Expand to fit the width and/or height to avoid the disappearance of lines.

for 3. You can set the margins, for the space between text and the borders instance,

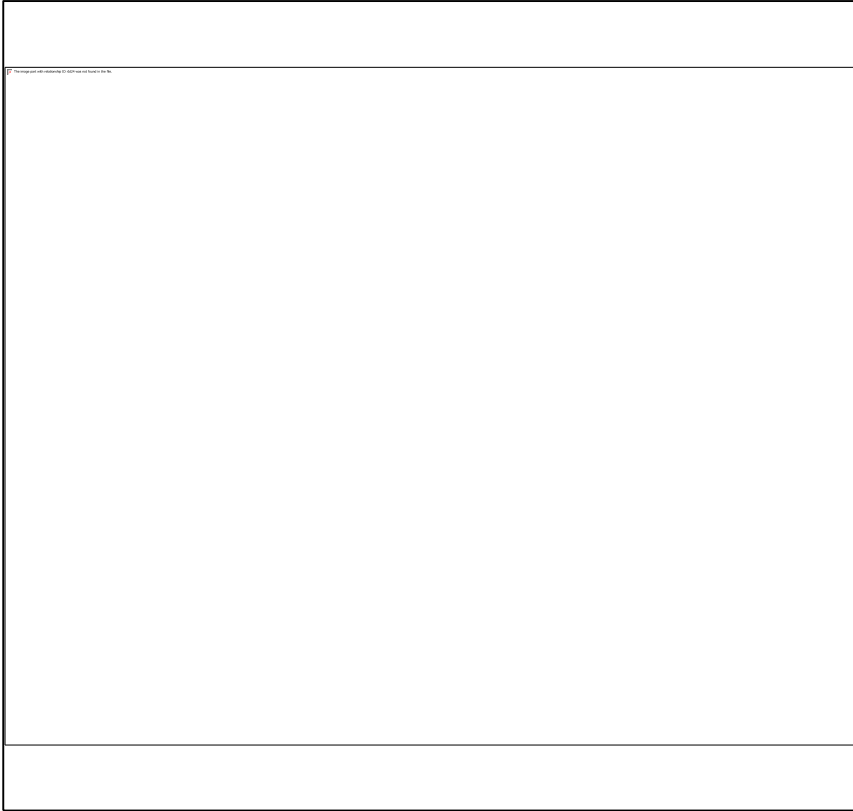
of the text object.

4. Dynamic and interactive objects (like text fields or checkboxes) will normally need to have a caption. You can determine its size and its position with regards to the object itself.

5. Objects can be rotated in 90° steps. You must specify around which anchor point the object should be rotated



# Designer - Borders and Background Colors



1. On the Border palette, you can determine edges and/or background fills.

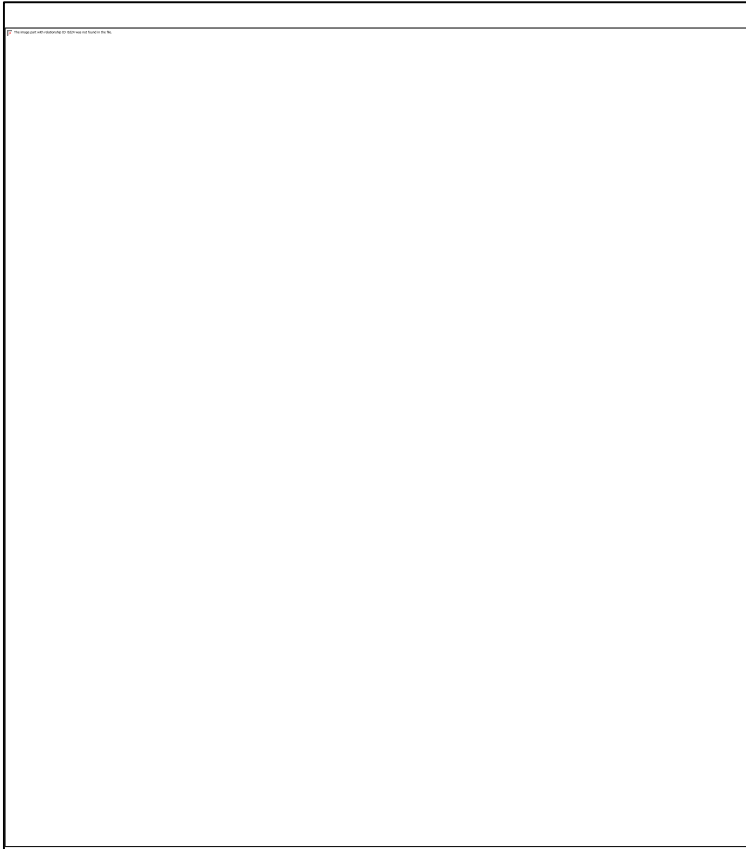
2. Edges can be edited together or individually.

3. For a background fill, you can choose between none, solid (one color) and various patterns for two colors.

4. For objects that are non-static (like a text field), you can also specify the border properties of the fillable areas. For example, you might choose to have a background color for a text field that differs from its caption color. To achieve this, select the object. In the Object palette, choose the Field tab. From the Appearance list, select Custom.

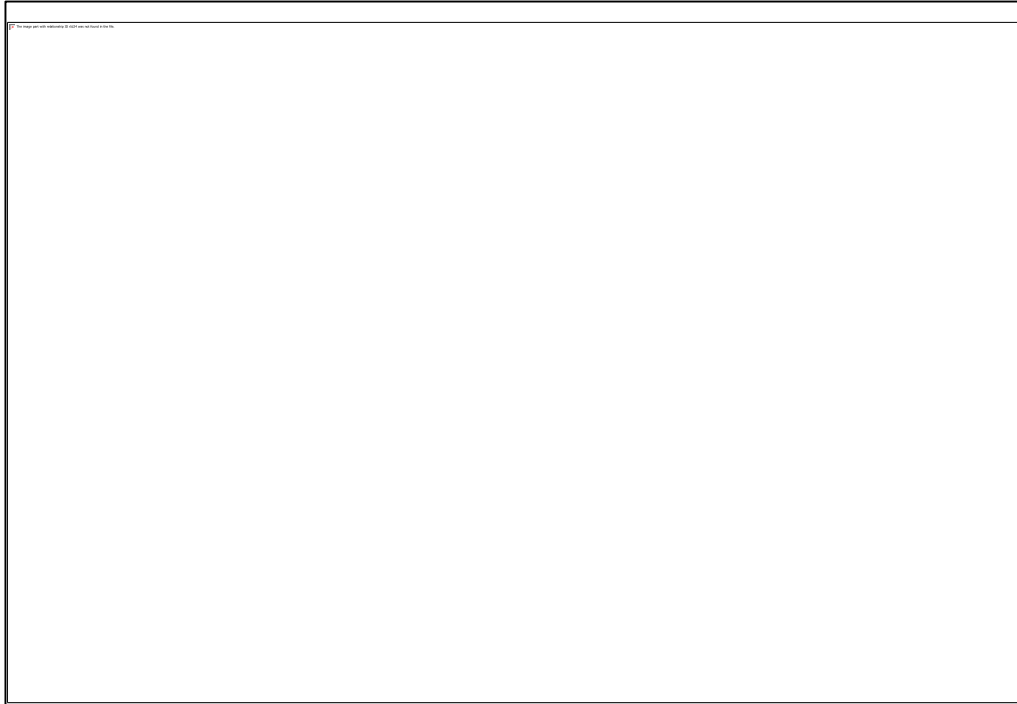


# Designer - Library



If you create an element and need to use it several times in your layout, it can be added to a tab page of the *Library palette*. You can then drag and drop your element from the *Library*, just like all predefined elements. All standard objects that come with Designer can be restored to any *Library tab* by selecting *Restore Standard Objects from the palette menu*. A library with its objects can also be published on a server

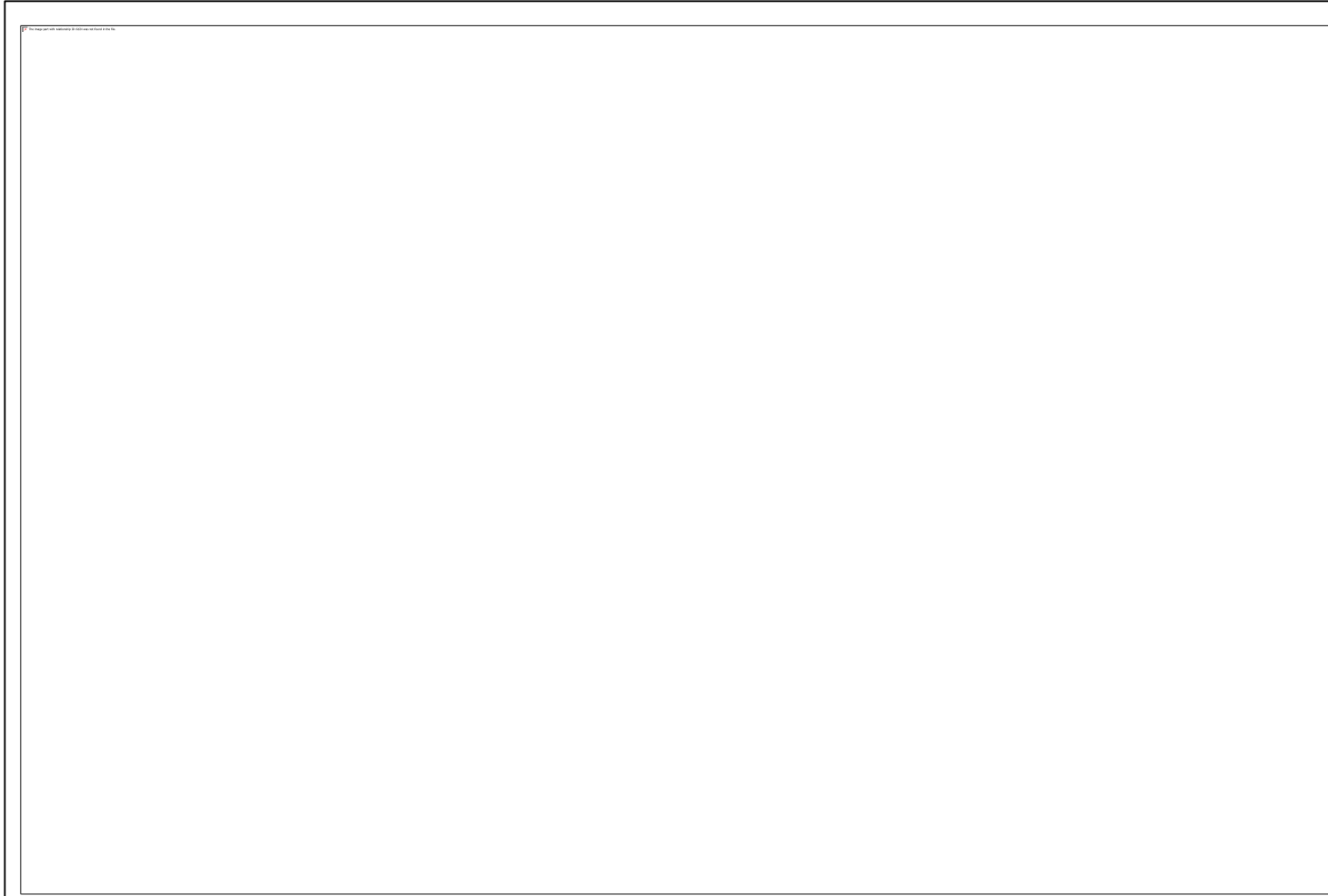
# Designer - Form Properties



# Designer

- 1.Master Page
- 2.Body Page
- 3.Content Area
- 4.Sub forms

# Designer



# Designer

Every form design contains at least one master page that Adobe Lifecycle Designer creates automatically.

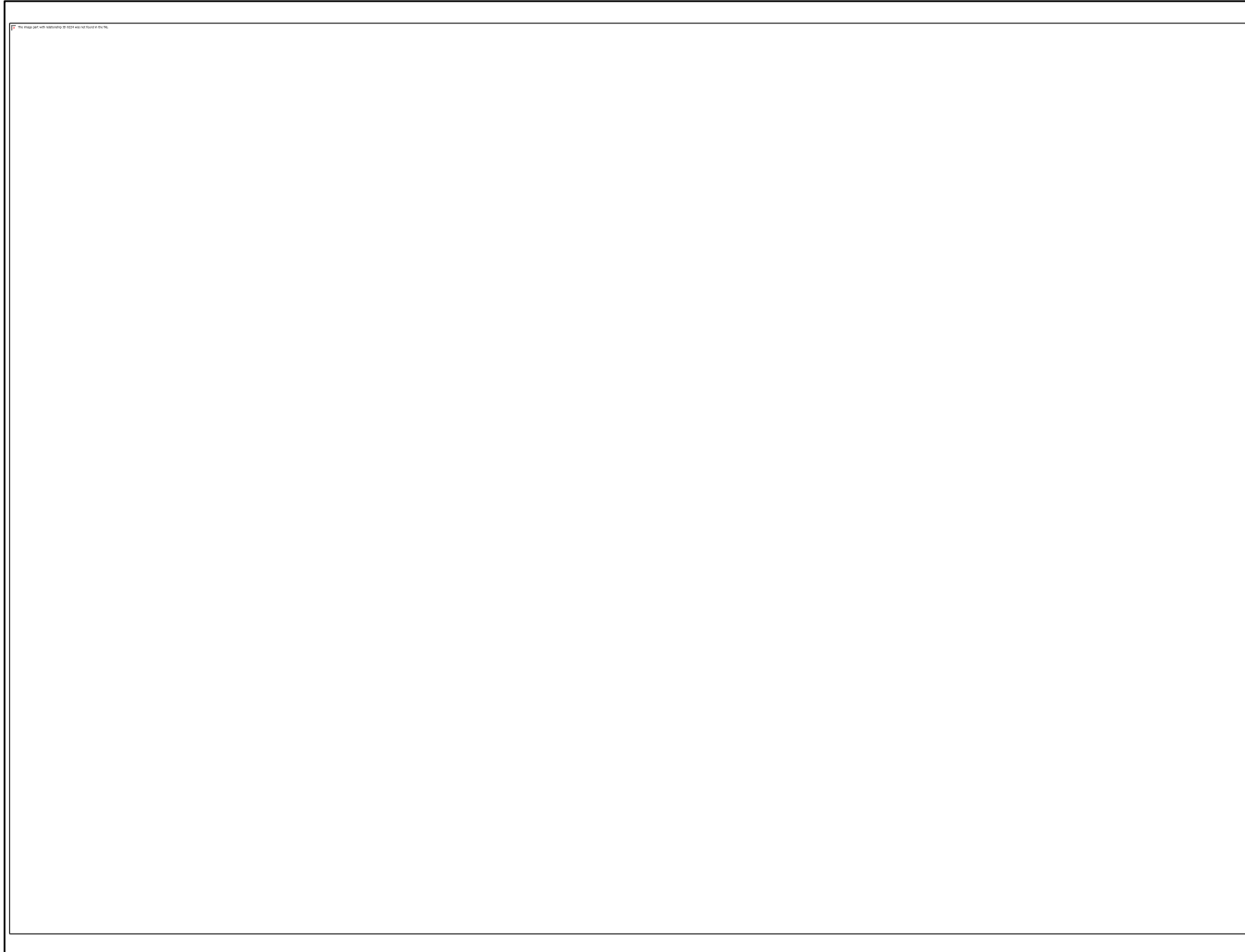
You can put objects on a master page that will appear on any resulting page at runtime, for example, your company logo. Such objects that never change are called boilerplate objects.

To some extent, the boilerplate objects of a master page could be compared to secondary windows in SAPscript or Smart Forms.

On a master page, you must include at least one content area. This defines the size to be used for dynamic output Content

Content areas can be included only on master pages.

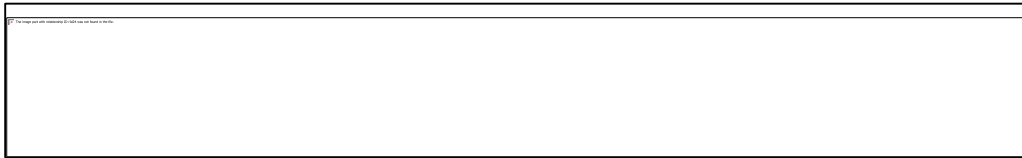
# Designer - Inserting Several Master Pages



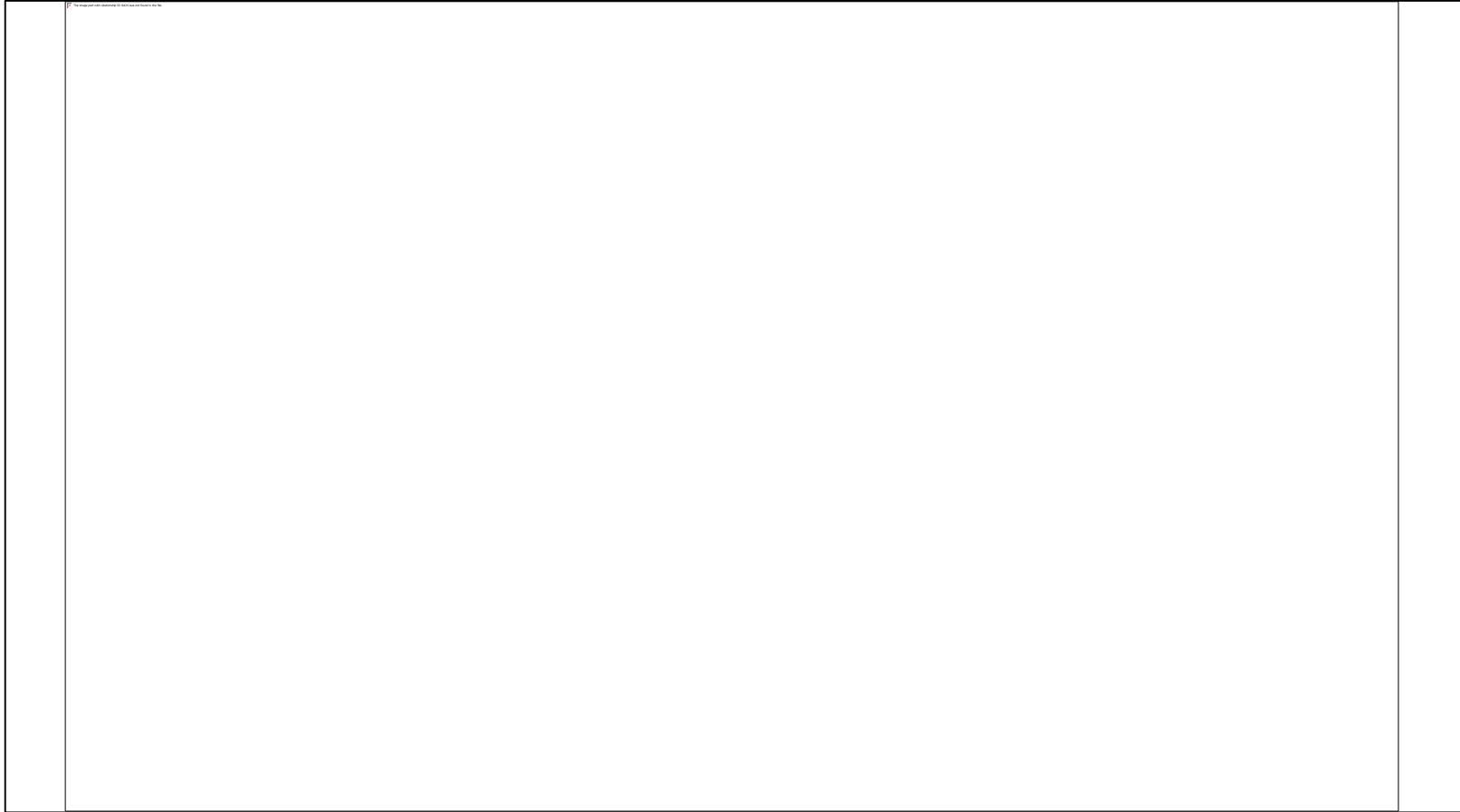
# Designer - Inserting Several Master Pages

For every master page, go to the *Object palette in order to set:*

- 1.The page size, for example, letter or A4
- 2.The page orientation, for example, portrait or landscape
- 3.Whether the page occurrence should be restricted: For example, you could restrict the occurrence for your first master page (which contains addresses, company logo, and so on) to 1. As a consequence, at runtime this master page would be taken for one output page only. The second master page in the *Hierarchy would by default automatically be taken for following output* pages if more data is laid down than can be displayed on a single page. If you restrict the maximum page occurrence without having another master page, this setting will be ignored at runtime if more data needs to be displayed.



# Designer - Body Pages and When to Include Them

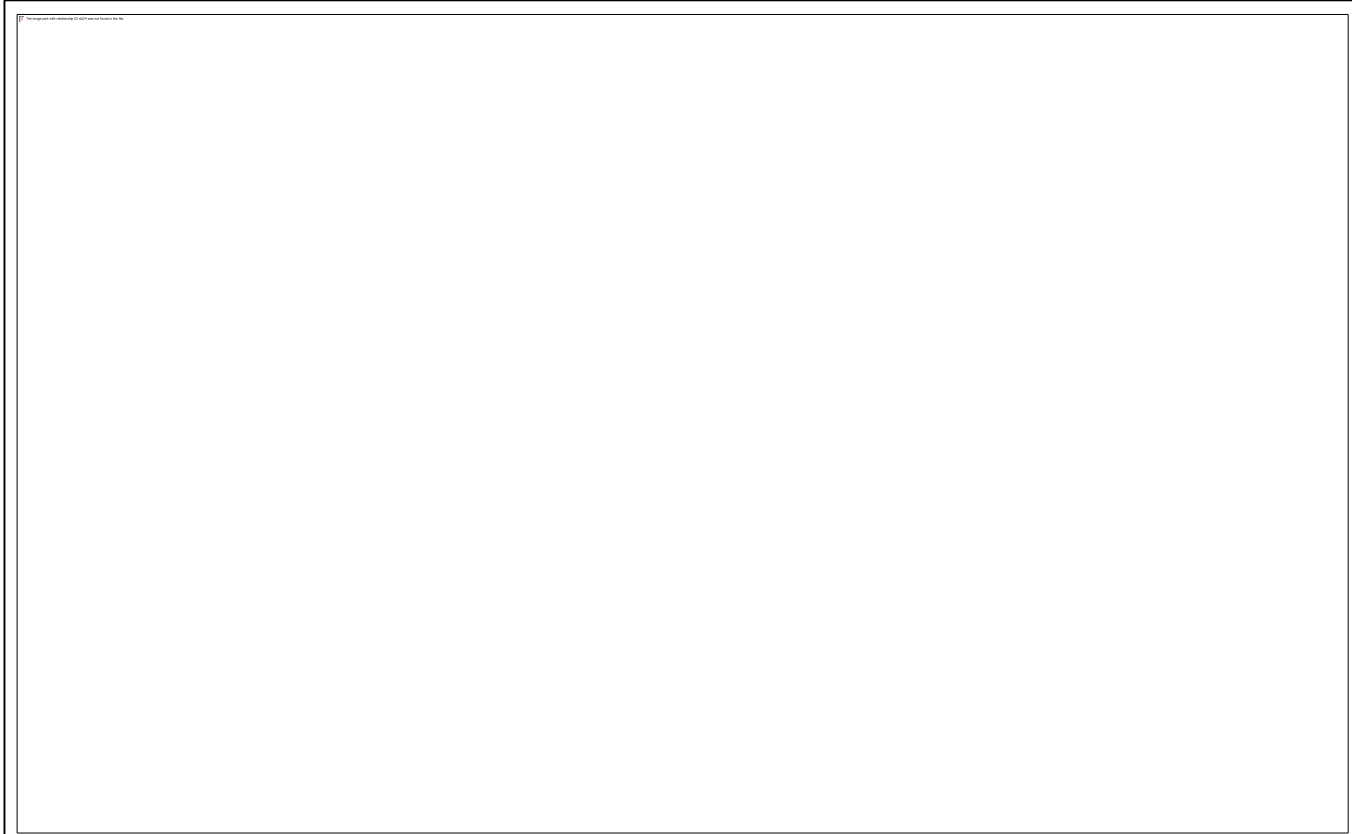




# Designer - Body Pages and When to Include Them

1. A body page is a top-level sub form. It serves as an organizing unit for dynamic content and can be laid down only in a content area of a master page.
  2. Dynamic content is wrapped up in body pages.
  3. A body page will use the space provided by a content area. If, at runtime, this happens not to be sufficient due to a large amount of data, the body page will look for the next content area (which might involve an automatic page break).
  4. To some extent, the combination of a content area and a body page included there could be compared to the main window in SAPscript or Smart Forms. (Note, however, that this is only a very rough equivalent!)
  5. For further organizing body pages (or, though rarely, master pages), they can contain subforms.
- A body page is a wrapper or a container for dynamic content

# Designer - New Page: Inserting the Terms and Conditions



# Designer

If you set a body page's place to *On Page <Master Page>*, there are two possible scenarios:

- 1.If the preceding sub form was already laid out on that master page, the body page will follow on the same output page (provided there is enough space).
- 2.If the preceding sub form was laid out on a different master page, a page break will be inserted and a new output page will begin using the desired master page.

To display the terms and conditions on a separate page with a special layout, you need to define an extra master page (let's call it *TERMS*) and create a new body page.

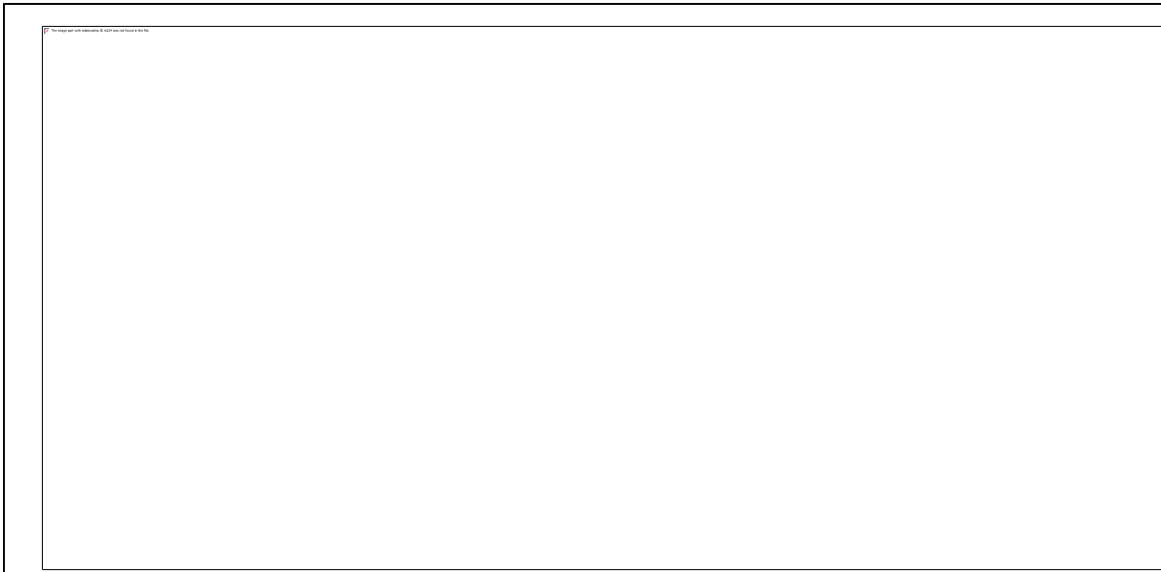
For the new body page (which is, like all body pages, a sub form), choose *Top of Page → TERMS for Place*

Instead of creating a new body page, you could tell the previous sub form to go to the top of page TERMS after it has been laid down.

# Designer

Subforms can be thought of as folders containing several objects. They can be used for the simple reason of keeping order in the Hierarchy, as it is possible to expand and compress subforms.

Subforms also help to rearrange several objects at a different place in the form.

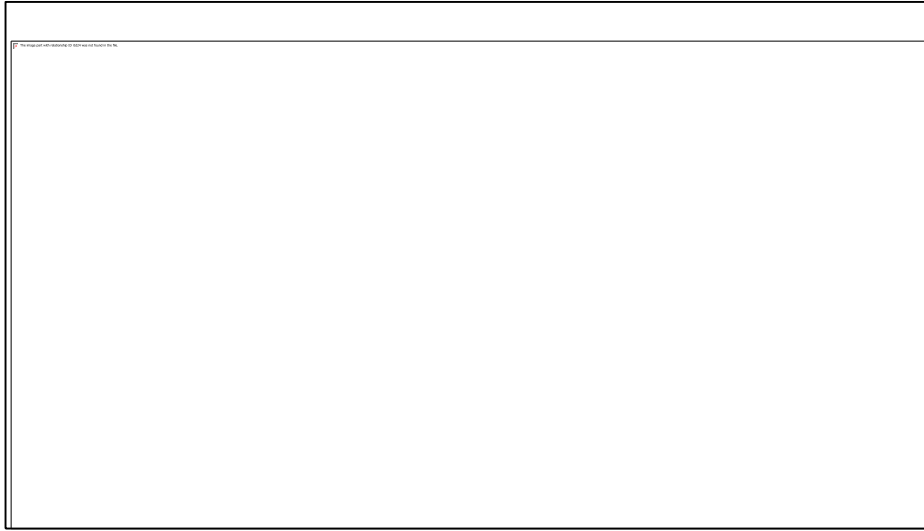


# Designer - Types of Subforms

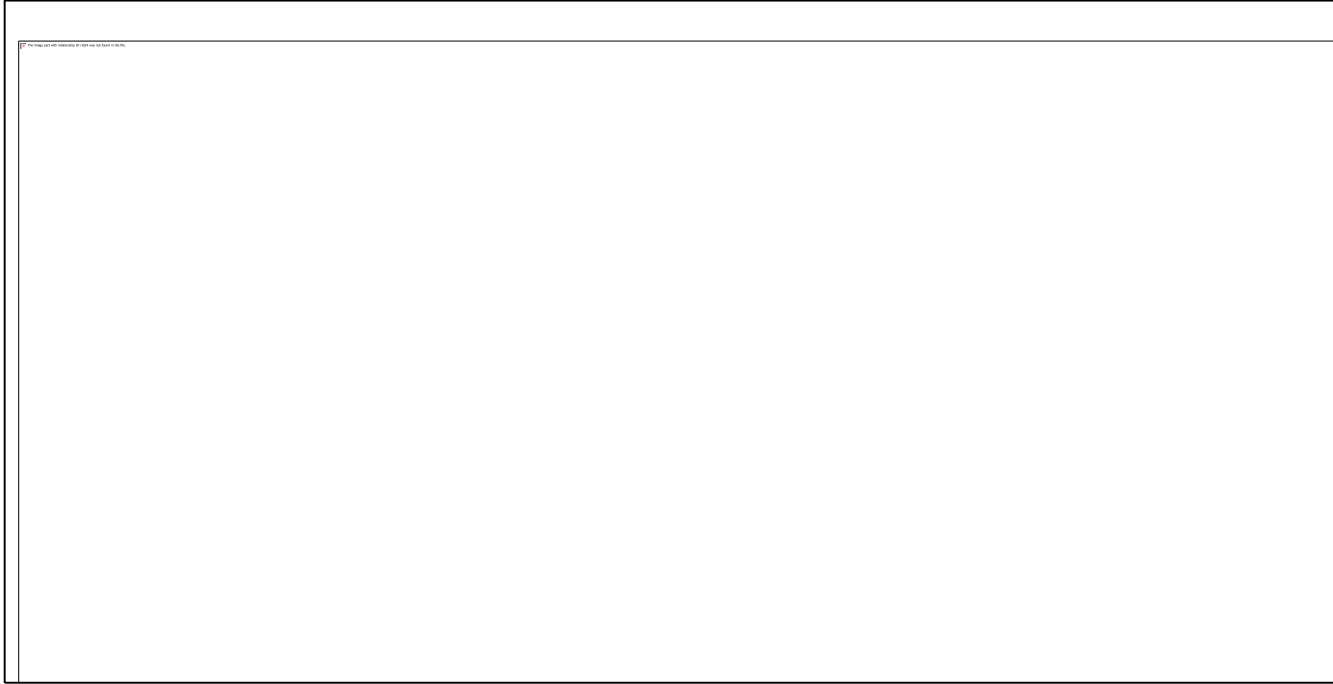
1.If of type *Positioned*, *objects of subforms can be laid down at their exact position at runtime, relative to the subform. For example, if a text field has been positioned at the top left corner of a subform of type Positioned, it will always be positioned at the top left corner of the subform, independent where on a page this subform is included. (The Hierarchy position of an object within a subform of type Positioned is irrelevant for its layout position.)*

2.If of type *Flowed*, the objects will follow each other, depending on the space they require at runtime. A body page (as the topmost subform) is typically of this type

# Designer - Types of Subforms



# Designer - Preventing Page Breaks

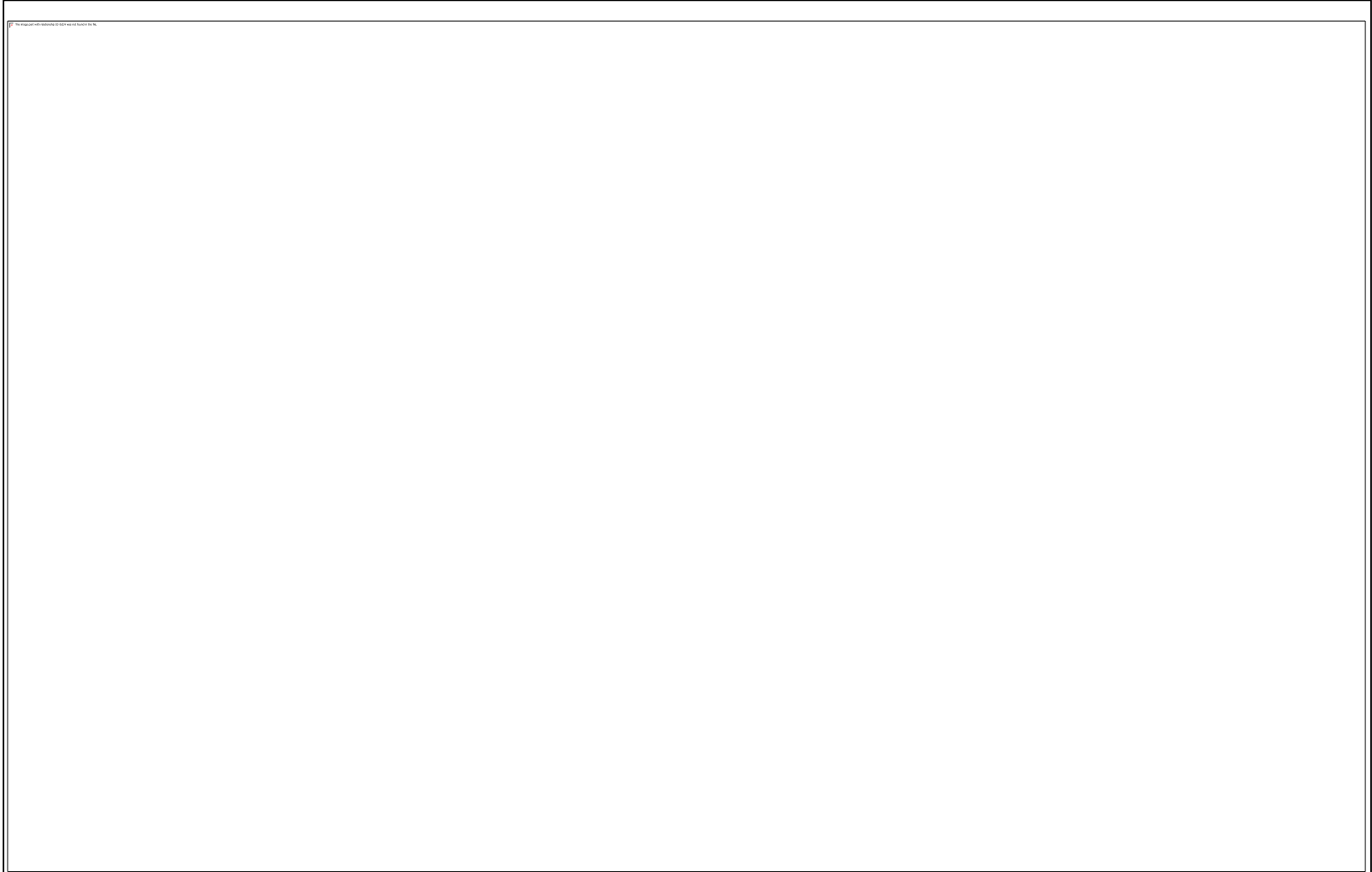


# Layout

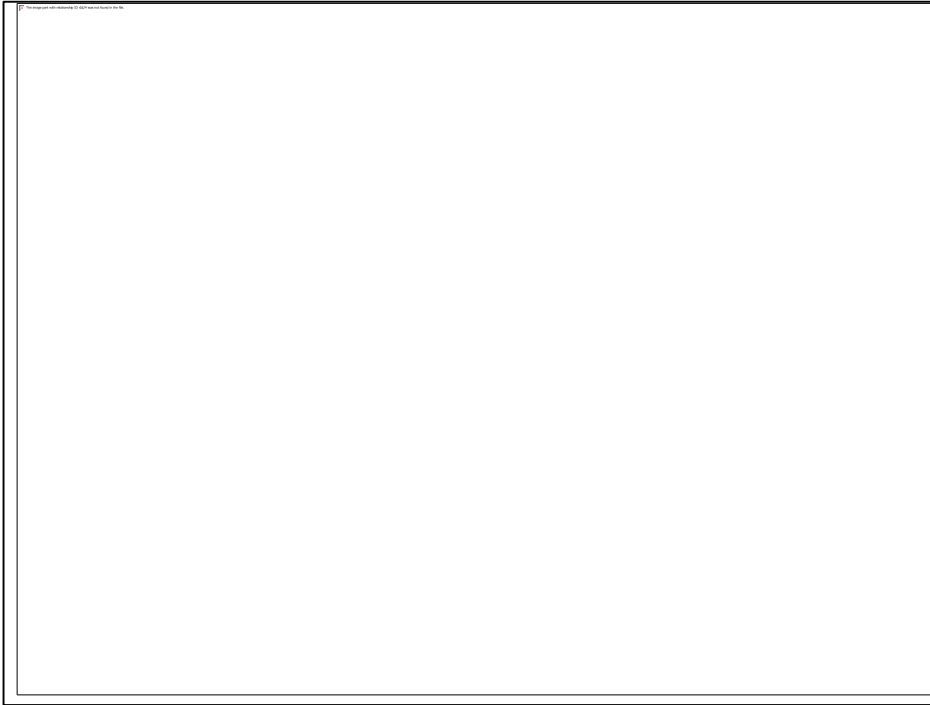
1. Insert static elements into a form: images, texts, and graphical objects
2. Set object properties for static form elements
3. Insert dynamic elements into a form: text fields, image fields, date/time fields, floating fields
4. Set the data binding (the connection between the layout fields and the business data)
5. Apply patterns (picture clauses) to influence field output
6. Insert tables into a form
7. Format tables
8. Set a header for a table



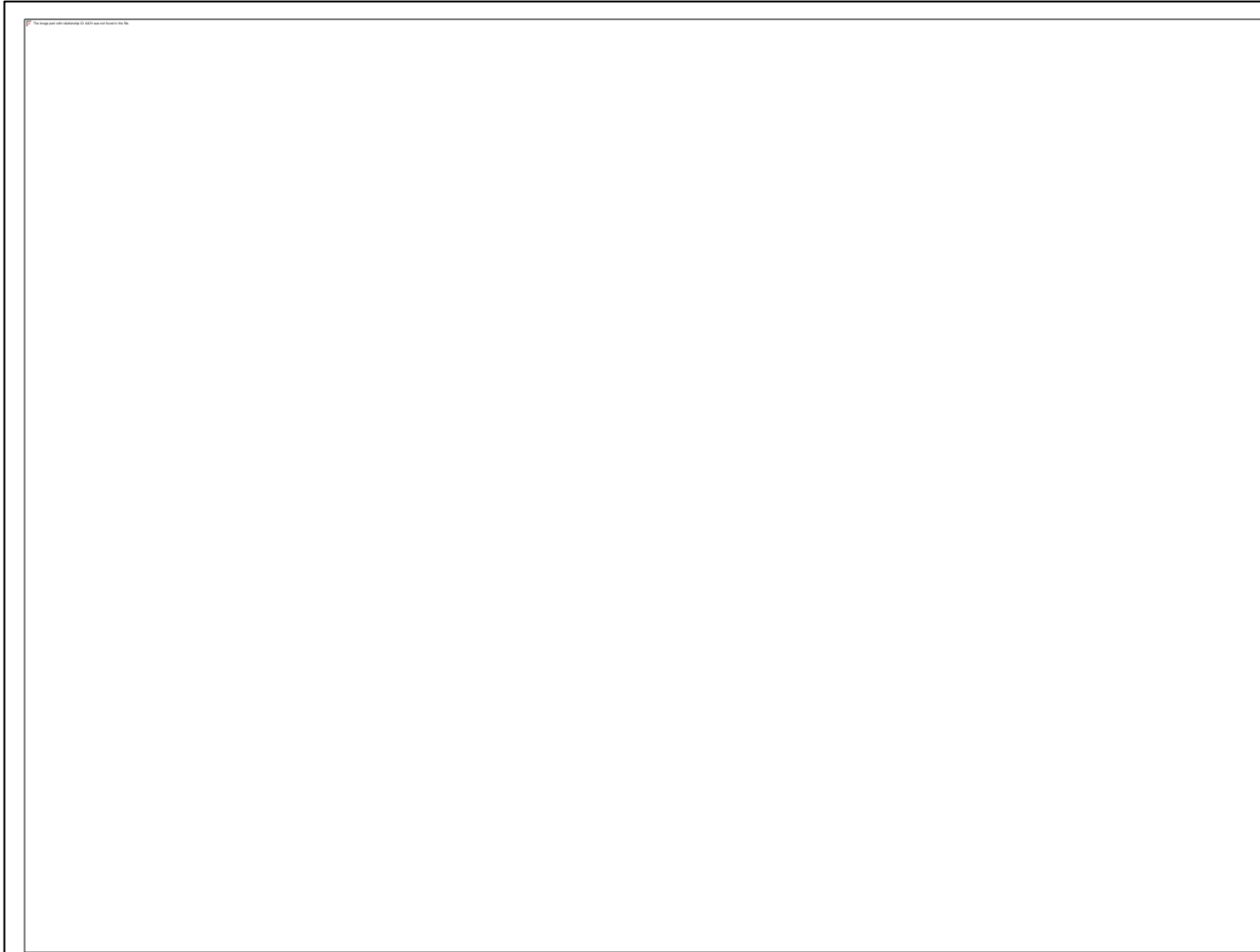
# Layout - Inserting Static Images



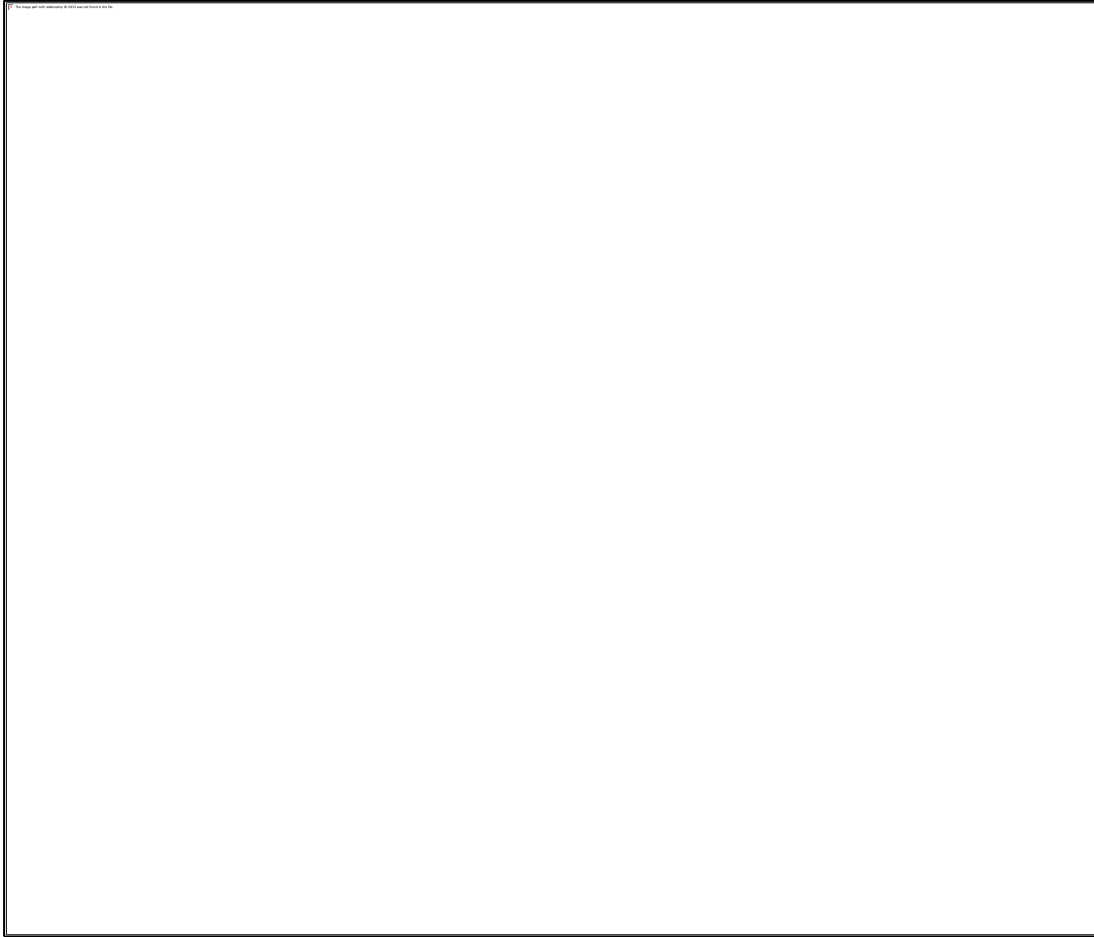
# Layout - Inserting Static Text and Geometric Objects



# Layout - Inserting Fields with the Data View

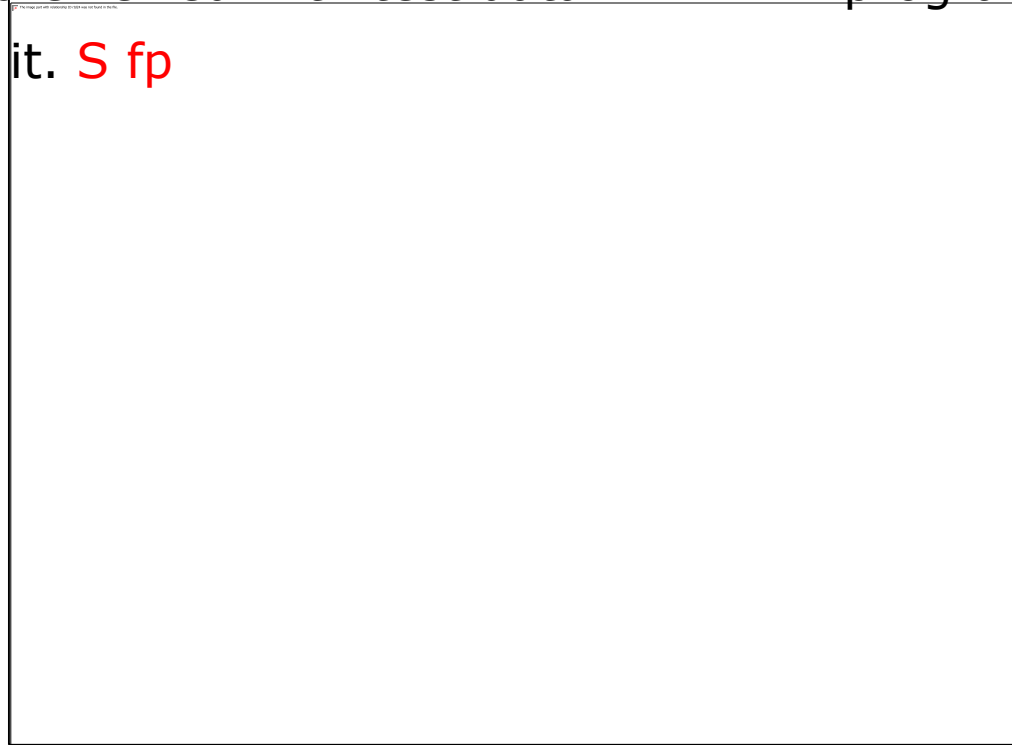


# Layout - Text Fields

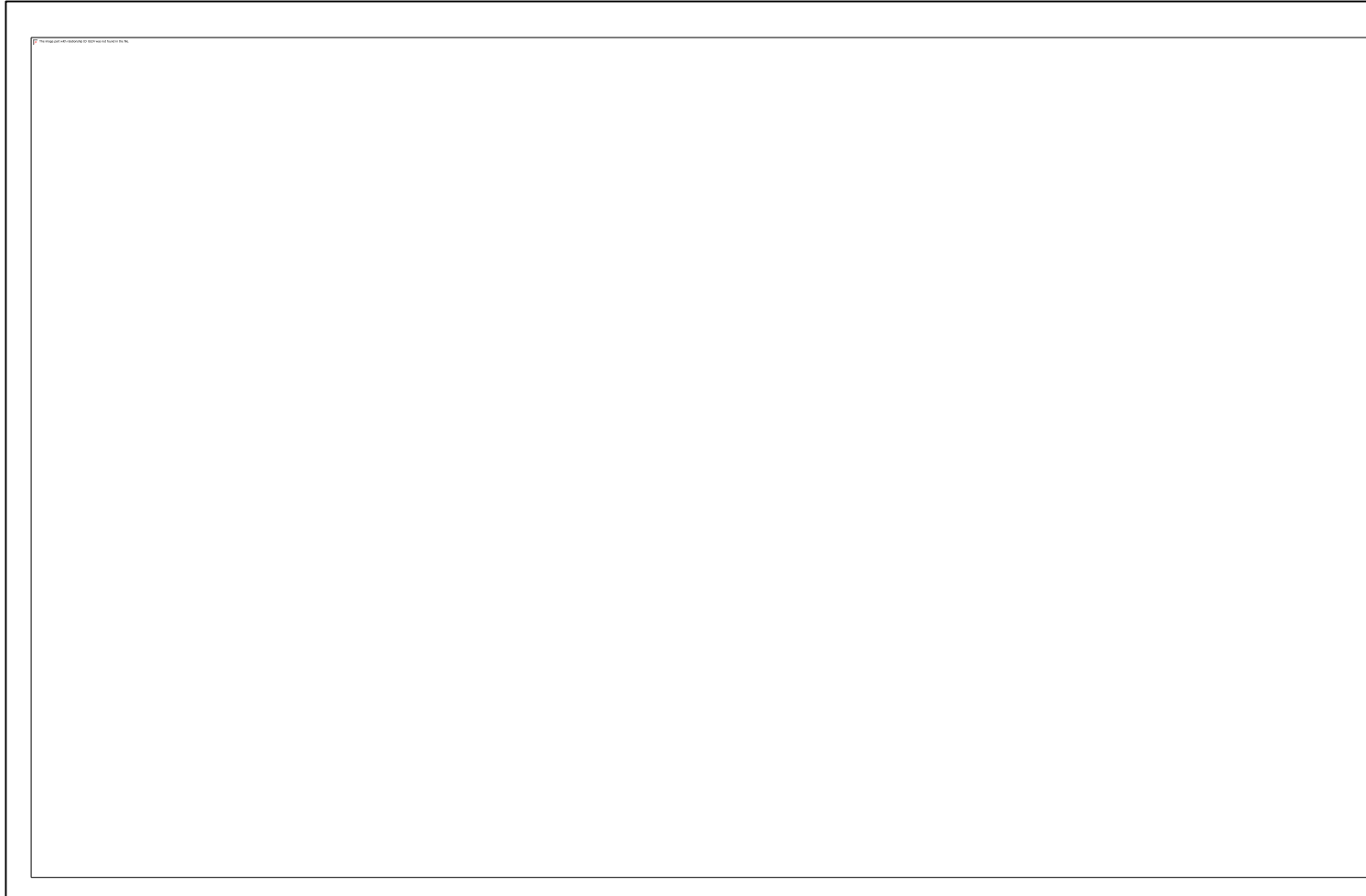


# Integration into ABAP Programs

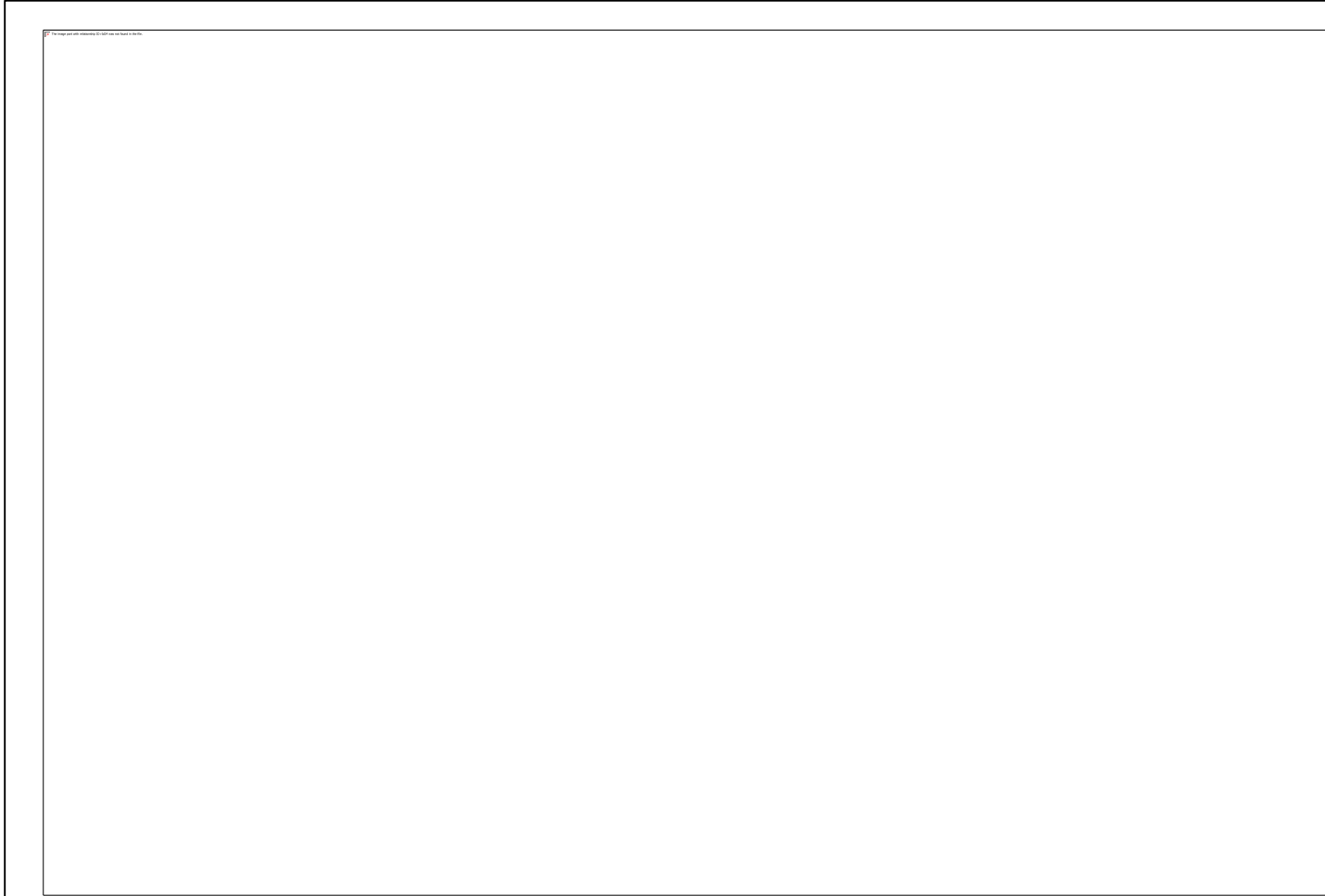
Designing the layout of a form is the most time-consuming part of administering a printing scenario. However, a form itself cannot be run; it can only be previewed with test data. An ABAP program is required to process it. S fp



# Integration into ABAP Programs



# Integration into ABAP Programs



# Integration into ABAP Programs





# Integration into ABAP Programs

