



Creating Accessible Forms With Adobe PDF Forms Access



Copyright ©2001-2002 Adobe Systems Incorporated. All rights reserved.

[Terms of Use](#)

[Online Privacy Policy](#)

Table of Contents

Adobe PDF Forms Access: Introduction to Accessibility Issues 1

Introduction 1

Section 508 and Electronic Documents 6

Accessibility Design Considerations 8

How Acrobat Helps With Accessibility 11

How PDF Forms Access Helps With Accessibility 13

Brief Review of PDF Forms 13

Exercise: PDF Form Field Properties 15

Summary 21

Adobe PDF Forms Access: Tagging PDF Forms 22

Introduction to PDF Forms Access 22

Overview of PDF Forms Access 24

Exercise: Initializing a Form Using PDF Forms Access 32

Modifying the PDF Forms Access Structure Tree 36

Exercise: Adding Elements to the Structure Tree 38

Adding Accessible Text to Form Fields 45

Exercise: Adding Accessible Text to Form Fields 51

Testing Accessible Forms in Adobe Acrobat 57

Introduction to Accessibility Testing 57

Opening a Tagged Form File in Adobe Acrobat 58

Exercise: Examining the PDF Tags Palette 60

Accessibility Testing 65

Exercise: Testing Form Accessibility 67

PDF Form Structure and Layout 72

Form Structure and Layout Guidelines 72

Form Structure and Layout 73

Exercise: Form Layout Guidelines 74

Creating Accessible Form Tables 90

Introduction to Form Tables 90

Form Tables 91

Table of Contents

Exercise: Creating Accessible Form Tables 93

Importing Forms Structure 109

Importing Forms Structure and Concluding Topics 109

Importing Structures 110

Exercise: Importing a Form Structure 111

Wrapping It Up and Putting It All To Use 119

Exercise: The Final Test 122

PDF Forms Access Summary 124

Introduction to Accessibility Issues



Introduction

[Instructor Notes: The course is comprised of two modules: this short Introduction module, and a second much larger module on using PDF Forms Access. The Introduction module covers general issues relevant to creating accessible documents, not just forms, explains how the Acrobat Tagged PDF format facilitates that, and what PDF Forms Access does in regard to making forms accessible. Finally, the Introduction module briefly reviews the basics of creating PDF forms. Students who are very familiar with accessibility issues and are experienced with creating PDF forms can safely skip this section if they wish to do so.](#)

This course will guide you through the process of creating PDF Form documents that can be accessed and used by individuals who are blind or otherwise visually disabled. This first section will provide an overview of some of the general issues and challenges involved in creating accessible electronic documents. To some extent, this may require a rethinking, or at least an awareness of, the way we present information. It requires using software to produce accessible documents that preserve the intended content ordering of the original document, and that provide alternate descriptions for visuals such as pie charts, tables, and graphics, and, in the current context, for fillable fields in interactive forms documents. Adobe Acrobat's Tagged PDF capabilities provide the functionality needed to meet these requirements. PDF Forms Access converts non-tagged PDF forms to Tagged PDF format, automatically creating accessible PDF Tags for fillable fields in the forms document.

The starting point for the exercises and examples used in the course is an existing, untagged PDF file to which fillable or interactive form fields have been already been added using the Adobe Acrobat 5 Forms Tools. For information on creating non-form Tagged PDF files from various authoring tools, such as FrameMaker®, InDesign®, and MSWord®, refer to Adobe's *Authoring for Accessibility* series of courses. For information on creating PDF Forms documents from scanned sources, refer to documentation for Acrobat Capture. References for each are provided below in the [References](#) section.

Through the hands-on exercises included in the course, you will become familiar with the PDF Forms Access interface and functionality, you will use PDF Forms Access to produce an accessible PDF Form document, and you will make adjustments and modifications to the PDF Forms Access tag structure in order to meet the particular requirements of the sample document. You will test the accessibility of the processed document using a screen reader and other methods.

For further material in addition to this course, you can also refer to the Web sites and other resources listed in the [References](#) section below.

Learning Objectives

The objectives of this section are as follows:

- To discuss some of the basic issues and challenges involved in creating accessible documents.
- To describe how the PDF Tag structure provides information that is used by screen readers.
- To introduce Adobe PDF Forms Access® and its capabilities.
- To summarize a typical workflow for creating and troubleshooting accessible PDF form documents using PDF Forms Access.

References

- <http://access.adobe.com>
- <http://www.adobe.com/products/acrobat/solutionsacc.html>
- *How To Create Accessible Adobe PDF Files*
 - PDF Version: <http://www.adobe.com/products/acrobat/pdfs/accessbooklet.pdf>
 - HTML Version: <http://access.adobe.com/booklet1.html>
- *Advanced Techniques for Creating Accessible Adobe PDF Files*
 - <http://www.adobe.com/products/acrobat/pdfs/CreateAccessibleAdvanced.pdf>
- White Paper: *Enhancing the accessibility of the Web with Adobe Acrobat software*
 - PDF Version: <http://www.adobe.com/products/acrobat/pdfs/accesswhitepaper.pdf>
 - HTML Version: <http://access.adobe.com/whitepaper1.html>
- White paper: *Accessible Design Guidelines*, Sarah Horton, Dartmouth College, May 2, 2002. Good treatment of many aspects of designing accessible documents: not specific to product. Many useful resource links. <http://www.dartmouth.edu/~webteach/resource/download.html>
- Section 508 Analysis: http://www.adobe.com/products/acrobat/pdfs/Sect_5085.pdf
- *Acrobat 5.0 and Accessibility Frequently Asked Questions* <http://www.adobe.com/products/acrobat/pdfs/Acro5AccessFAQ.pdf>
- Section 508 Press Release <http://www.adobe.com/aboutadobe/pressroom/pressreleases/200106/20010621508.html>
- FOSE Press Release (discussing Acrobat 5.0 as it relates to Government and Accessibility) <http://www.adobe.com/aboutadobe/pressroom/pressreleases/200103/20010320fose.html>
- Adobe on-line accessibility courses: *Authoring for Accessibility*. <http://partners.adobe.com/access/acroaccess.html>.

Product Demonstrations and Documentation

- Adobe Acrobat 5.0 and Accessibility Video Series
 - Working with Existing PDF files
<http://www.adobe.com/products/acrobat/movie2.html>
 - Working with Forms
<http://www.adobe.com/products/acrobat/movie3.html>
 - Usability Enhancements of Acrobat 5.0
<http://www.adobe.com/products/acrobat/movie4.html>
 - Working with Screen Readers
<http://www.adobe.com/products/acrobat/movie5.html>
- Acrobat Capture
http://partners.adobe.com/asn/developer/acrosdk/docs/Capture_API_Overview.pdf
http://partners.adobe.com/asn/developer/acrosdk/docs/Capture_API_Reference.pdf

Downloads

- Acrobat 5.05 Update (you must have Acrobat 5.0 installed to use this)
 - <http://www.adobe.com/products/acrobat/update.html>
- Acrobat Reader 5.0 Download (the version that includes accessibility)
 - standard presentation page:
<http://www.adobe.com/products/acrobat/readstep.html>
 - text-only page:
<http://www.adobe.com/products/acrobat/alternate.html>
- Additional Adobe product downloads for accessibility
<http://access.adobe.com/downloads.html>

Section 508 Information

- <http://www.access-board.gov/sec508/guide>
- <http://www.section508.gov/>
- <http://www.access-board.gov/news/508-final.htm>
- <http://www.itpolicy.gsa.gov/cita/fap.htm>

Vendor Information

- Screen readers:
 - Freedom Scientific Jaws (<http://www.freedomscientific.com>)
 - GW Micro WindowEyes (<http://www.gwmicro.com>)
 - Dolphin Oceanic HAL (<http://www.dolphinoceanic.com>)
- <http://www-3.ibm.com/able/overview.html>
- <http://www.microsoft.com/enable>

Course Contents

Topics	Exercises
Section 508 and Electronic Documents	
Accessibility Design Considerations	
How Acrobat Helps With Accessibility	
How PDF Forms Access Helps With Accessibility	
Brief Review of PDF Forms	PDF Form Field Properties
Creating Accessible Forms with Forms Access	Initializing a Form Using PDF Forms Access
Modifying the PDF Forms Access Structure Tree	Adding Elements to the Structure Tree
Adding Accessible Text to Form Fields	Adding Accessible Text to Form Fields
Testing Accessible Forms in Adobe Acrobat	Examining the PDF Tags Palette
Accessibility Testing	Testing Form Accessibility
PDF Form Structure and Layout	Form Layout Guidelines
Creating Accessible Form Tables	Creating Accessible Form Tables
Importing Forms Structure	Importing a Form Structure
Wrapping It Up and Putting It All To Use	The Final Test
PDF Forms Access Summary	

File to Download for Exercises in this Course

FormsAccessExamples.zip, which is normally found in the *Exercises* folder, contains the following files:

- **PersonalData.pdf**
- **PersonalData_underline.pdf**
- **PersonalData_unfinished.pdf**
- **ProblemForm1.pdf**
- **ProblemForm2.pdf**
- **ProblemForm3.pdf**
- **ProblemForm4.pdf**
- **Table1.pdf**
- **Table2.pdf**
- **Table3.pdf**
- **Table1_ext.pdf**
- **Table1_org.pdf**
- **Table1_org-new.pdf**

- `Jaws1.wav`
- `Jaws2.wav`
- `form_1040.pdf`
- `f1040_accessible.pdf`

You should extract these files from the zip archive if they are not extracted already. You will use `PersonalData_unfinished.pdf` in the exercise accompanying this section.



Section 508 and Electronic Documents

Although the World Wide Web has revolutionized the flow of information to people around the globe, individuals who are blind, have low visual acuity, or have motor impairment can find interacting with computer technologies challenging and frustrating.

[Instructor Notes: Mention that Section 508 covers a broad range of disabilities in many situations; access to electronic documents is just one area. Because documents are by nature visual, the focus of this course is toward providing access to electronic documents for blind and visually impaired persons.](#)

The Architectural and Transportation Barriers Compliance Board (Access Board) issued accessibility standards for electronic and information technology on December 21, 2000. This document is known as Section 508 and requires Federal agencies' electronic and information technology is accessible to people with disabilities. Information technology also encompasses electronic documents and the software used to read them.

Accessibility should be a relatively easy, low-effort addition to the documentation production process. This can be true provided that you consider accessibility during the document design phase. Most documents are geared towards sighted individuals. Layouts of information that interweave graphics and text with the intention of being read in a particular reading order can result in a confused presentation from the screen reader. A screen reader is a voice-synthesized program that converts the text of a document to an audible stream. People who rely on screen magnifiers to read information can only view a small portion of the document at a time and may have problems navigating a document with an artistic layout. Sighted people can easily grasp trends from data presented in graphs and tables, but that visual context is not available for a visually-impaired reader.

The following document illustrates some of these points. Should the screen reader ignore the logo? Where should it start reading? Does the picture convey something that is important for non-sighted viewers to understand? If so, what is the best way to explain this in text? How can the information presented in the table be summarized effectively? In order for a vision-impaired reader to make sense of this document, we must somehow convey the structure as well as the content of the document. For example, what order should information be read. What should be said about the pictures and tables?

The screenshot shows a document layout with several labeled components:

- Logo:** Line-A-Sight logo in the top left corner.
- Title:** 'Line-A-Sight LensPrep' at the top right.
- Subtitle:** 'High-speed automated lens finisher, center and surface' below the title.
- Column 1:** The left column containing introductory text and a table.
- Column 2:** The right column containing a scanned image of a lens and descriptive text.
- Table:** A table titled 'Test Results on Various Surfaces' located in the lower part of Column 1.

Product	Units per hour
Resin on a lens	175 units
UV-cured glass	177 units
Glass inside lens	173 units
Acrylic Polyvinyl	184 units

Proper presentation of information depends on several factors. Encoding the proper, or logical, reading order for a document is one of the most fundamental steps towards accessibility. This is especially critical for documents with characteristics such as columned formats, multiple distinct text blocks, or fillable fields with accompanying text descriptions or instructions. Designing a document for accessibility may also mean rethinking any messages conveyed solely with visuals. Presentations involving large tables of data or other information may need to be assessed for readability: processing a table with a screen reader is much more cumbersome than processing a table with the benefit of sight. In general, the the simpler a document’s format and layout is, the easier it will be to make it accurately and usefully accessible. When that is not possible or desirable, the process of making the document accessible also becomes somewhat more complex.

[Instructor Notes: It is possible to make visually complex documents accessible, but it can take a great deal of work depending on the complexity of the document and the authoring software.](#)



Accessibility Design Considerations

This course is focused on PDF Form document accessibility. Although form design issues, such as page and field layout, can affect the way PDF Forms Access processes a form, and although some recommended practices, and practices to avoid, when creating forms will be covered in a later section, a detailed treatment of interactive form documents is well beyond the scope of this course. However, if you are in the position designing and creating PDF forms from scratch, there are some basic guidelines you may wish to keep in mind. Whether you are publishing a document in HTML, XML, Adobe PDF, or some other format, creating accessible documents often requires more than simply representing the original document accurately. Sighted people can look at a printed page and easily discern the difference between titles, subtitles, columns of text, headers, footers, fillable fields and so on. If the document is a form, they can determine which block or set of instructions goes with each fillable field, sometimes regardless of the distance or position of each to the other. Visual clues, such as location of the text on the page, bold text, and large font sizes, as well as arrows, pointers, and lines, help sighted persons determine the structure of a document so they can read and navigate it easily.

Unfortunately, assistive technologies such as screen readers cannot depend on these visual clues. They must rely instead on the underlying computer-based information to provide that same structure. As a result, making documents accessible depends on two things:

- Authoring the original documents so that they contain not just content, such as the text in the document, but also information about the structure of the content, such as how the text flows within the page and from page to page.
- Using publishing tools that can retain and encode both the content and the structure so that it can be interpreted by assistive technology.

In order to do this:

- Authors need to be aware of the importance of writing with the intent of creating accessible documents, and how to accomplish that.
- Authors need to be aware of the features within their authoring applications that support accessibility and make full use of them.

We strongly recommend that you design for accessibility as much as possible in your documents and select publishing tools that support the generation of Tagged PDF files.

[Instructor Notes: Software development in the area of structuring and tagging documents for repurposing and accessibility is probably at the first generation stage and will continue to evolve rapidly. If accessibility is a major concern in your documentation efforts, we suggest always using the latest versions of both your authoring software and Acrobat.](#)

We also recommend that you use the most recent versions of your authoring software. Older software does not do a good job creating Tagged PDF. Often, the

resulting Tagged PDF requires a significant amount of work to fix problems that occur during the conversion process.

Document Design

In terms of actual document design, the most problematic areas are complex layouts and visual presentation of information. Visual presentation of information includes elements like charts, pictures, and graphics, but also any relevant content presented in bitmapped or other non-text format. If you intend on using a multi-columned layout, make sure that your authoring software has built-in tools to support this and that the content converts accurately to Tagged PDF. Layouts found in newspapers and magazines, where multiple unrelated articles appear on a page and continue on different pages, pose particular challenges. The authoring software must provide a mechanism to allow you to indicate the read order. In other words, you must be able to define where the continuation of each story can be found such that it is converted to Tagged PDF. Without this capability, you are faced with a very time-consuming task of defining the read order in the Tagged PDF so that it is accessible.

You should evaluate what kind of information is being presented by graphics in the document. Is the information already present in the text? Is the visual simply providing color and images that are not essential to the message conveyed by the document? If the graphic is a table, bar chart, pie chart, and so on, you should summarize the findings. For example, if a bar chart is showing the projected income over the next four quarters, state that and include the actual income figures in the alternate text. If a table is summarizing information, such as product requirements on different operating system platforms, you may want to consider including it in the text of your document.

Because accessibility is such a new area, the available authoring software varies in its capability to produce optimal Tagged PDF. For example, the work required to retain the reading order for a visually complex document, such as a brochure, may vary from application to application. Unless the logical structure of such documents is carefully prepared, usually involving iterative testing, it is not unusual for the sequence of the Tagged PDF structure to be almost but not quite right in terms of conveying the logical flow of the document's contents. It is always better to adjust the original document to produce the desired PDF Tag sequence rather than adjusting the Tagged PDF structure using the Acrobat Tags Palette.

Workflow for Creating Accessible Documents

Creating an accessible document is an iterative process. Accessibility is still evolving. Expect that the techniques and guidelines will change over time as well as the capability of various software applications to support this process. However, the authoring software that you use as well as the types of documents your department produces will influence the workflow.

Within the authoring application, the user creates a document template that defines the overall look and appearance of the document. The template also includes the paragraph and character styles that will be used by the author.

If the document is to be a form, the user would also use graphics tools to create underlines or borders for fillable fields, if desired, and position them in regard to labels or text instructions. A non-tagged PDF file would then be generated using the method particular to the authoring software.

Once a PDF version of the file has been generated, the form developer uses the Acrobat Forms Tool to create fillable fields at appropriate positions on the form. This step includes specifying field characteristics such as type, appearance, actions, and format. A Short Description can also be provided for fields at this stage, or after loading the form into PDF Access. Short Descriptions are read by screen readers when they encounter fields in a form. Any Short Description you provide for a field when you create it in Acrobat will be retained when the form is loaded into PDF Forms Access, where it can be edited further if desired.

The form should be reviewed and tested. At that point, the author can use PDF Forms Access to generate a Tagged PDF file which will include a unique tag for each active field in the form.



Note: PDF Forms Access cannot be used with *Tagged* PDF files. You must *uncheck* the Acrobat Distiller option for creating Tags in the *Tags tab* when given the opportunity just prior to the generation of the file.

Once the Tagged form has been generated, you must verify that the content is actually accessible. Shortcut methods can and should be used much of the time during the tagging process to verify that necessary elements are in the text stream, which almost certainly guarantees they will be accessible by a screen reader. Using a screen reader can be tedious and confusing, and you will probably want to use one somewhat infrequently during development of the document. However, no other testing method is more useful due to the insight it will provide you about how the document will actually be presented to a visually impaired person.



How Acrobat Helps With Accessibility

As mentioned before, an accessible document must be able to give a visually-impaired person cues about the reading order of the text and alternate explanations of elements such as fillable fields, graphics, and tables. Adobe Tagged Portable Document Format (PDF) provides these capabilities.

Converting documents to Tagged PDF is a robust, long-term solution for creating accessible documents. Tagged PDF represents various components of a document, such as chapters, heading styles, blocks of text, fields, tables, graphics, and so on, as tag elements. The tag structure is similar to markup languages such as HTML and XML. A document's structure is represented as a hierarchy of tag elements. The order in the hierarchy represents the reading order of the document. Since the content is represented with tag elements, other applications can extract the information and reuse it for other purposes. For additional technical information about Tagged PDF, please refer to section 9.7 Tagged PDF in the PDF Reference, Third Edition, version 1.4.

[Instructor Notes: PDF Tags are essential for accessibility because they provide the hooks into the document that other applications, such as screen readers, can use. However, tags are only part of the total equation in the sense that the application must decide how to make use of the tag information provided to it. The document developer has no direct control over this part of the equation; screen readers and other accessible devices evolve and become more sophisticated over time. The primary goal of the document developer in terms of accessibility is to make sure that the available hooks are in place so that applications can access them for information about the document. PDF Tags generally provide more document information than is currently used by alternate access devices. Tags placed in a document today will be utilized in increasingly sophisticated ways as access technology evolves over time.](#)

Tagged PDF offers the following benefits

- You can associate additional information with a particular tag element, such as a fillable field or a graphic, by using its alternate text property.
- Other applications can automatically reflow text and associated graphics to fit a page of a different size than was assumed for the original layout.
- The document's content can be converted to other common file formats (such as RTF, HTML, and XML) while preserving the structure and basic style information.
- It retains one of the primary advantages of PDF, the ability to preserve the exact look and feel of any source document, including all of the fonts, formatting, colors, and graphics, regardless of the application and platform used to create it.

[Instructor Notes: The core of PDF tag support is that the logical order of the content of the document can be determined independently of the document's visual appearance and layout by traversing the tag structure hierarchy and presenting the contents of each node, which is what devices like screen readers do.](#)

Tagged PDF documents are readily available to the general public since there are more than four hundred million copies of the free Adobe Acrobat Reader in use. Over two hundred government agencies worldwide have adopted PDF. As well as providing for accessibility of information in Adobe Portable Document Format (PDF) files, Acrobat 5 also provides improved accessibility for its own operations and functions.

Acrobat assists a user who wishes to read an accessible document by providing the following features:

- Provides usability enhancements, including enhanced keyboard shortcuts, support for high-contrast viewing, and the ability to zoom in and reflow text on the screen.
- Supports screen readers: Provides direct support for screen readers via the Microsoft Active Accessibility (MSAA) application programming interface (API) for Windows.® MSAA enables Acrobat 5.0 to integrate with assistive technology products including the newest versions of screen readers from vendors such as Freedom Scientific (<http://www.freedomscientific.com>), GW Micro (<http://www.gwmicro.com>), and Dolphin Oceanic (<http://www.dolphinoceanic.com>).

In terms of the accessibility requirements for software applications, mentioned in the previous section, Adobe Reader 5 and Acrobat 5 provide the following forms of support:

- Keyboard navigation

Acrobat Reader 5 provides users with keyboard shortcuts and key combinations for access to all program functions. In Acrobat 5.05, functionality that is also found in Reader is fully accessible. However, some functions unique to Acrobat 5.05, such as the freehand drawing tools used for electronic markup, are not available via keyboard.

- Logical order

The sequential order of the PDF Tags structure determines the order in which the elements of the document, headings, paragraphs, alternate text for graphics, and so on, are processed by devices like screen readers.

At the present time, only some of the information present in the Tags structure is available to screen readers. Consequently, screen readers primarily use the sequential positioning of elements in the PDF Tags structure. In the future, one would expect such devices, in conjunction with improvements in authoring software, to make use of additional structural information about the document, such as table heading tags and lists, to provide additional context about the document to visually-impaired users.

For that reason we encourage adding formal structure to electronic documents — selecting paragraph styles rather than formatting text with tabs, spaces, and carriage returns, for example.

- Content reflow

Acrobat 5.0 provides a TouchUp Order Tool for specifying the flow order of elements of the document when it is repositioned into a different layout or format. In regard to accessibility issues, the ability to reflow and reformat content, in addition to determining the sequential order in which that content is presented, allows a user with partially impaired eyesight to magnify the text in the document sufficiently to read it, and also to have it properly formatted within the viewing window and presented in the correct logical sequence.



How PDF Forms Access Helps With Accessibility

Adobe PDF Forms Access automates the tagging process required to create accessible Adobe PDF forms from untagged PDF forms. PDF Forms Access recognizes the objects and structures in a form and links content with fillable form fields to make forms accessible. While it is possible to manually create tags for data fields in forms, PDF Forms Access significantly decreases the amount of time and effort required to do so. This is particularly true as the number of fields in the form, or the complexity of the form, increases -- consider, for example, a typical form the IRS uses for tax returns.

The tag structure of the processed form can be modified and rearranged if necessary or desired. An extremely useful feature of Forms Access is that a tag structure from another document can be imported to the current document. In effect, this makes it possible to preserve the PDF Tag structure of a form between successive updates without having to completely reconstruct the tag structure in the updated document.



Acrobat Capture® Note: Adobe PDF Forms Access can be installed and run as a stand-alone application, or as an agent (plug-in) for Adobe Acrobat Capture 3.0.x. In Acrobat Capture, the PDF Forms Access appears under *Step Templates* as *Tag Form* and under the *Workgroup Tab* as a manual step. Where appropriate, the course will use an *Acrobat Capture Note* such as this to point out minor differences between running Adobe PDF Forms Access as a stand-alone application and running it as an Acrobat Capture agent.



Brief Review of PDF Forms

It is assumed that you have some general familiarity with PDF forms. While it is probably helpful to have had experience creating a PDF form from scratch, it is not a necessary prerequisite to understanding the process of making a PDF form accessible. A full treatment of the subject of creating PDF forms is significantly beyond the scope of this course. Refer to Adobe Acrobat documentation for detailed information about types of form fields and the various actions and calculations they can trigger. For those without extensive experience with PDF forms, this section and the exercise that follows will provide a very general overview of the procedures used to create fillable fields in a PDF document.

Creating a PDF form document can be seen as a three step process:

1. Create a document using a suitable authoring tool.

The first step is to create a document containing the layout and format for the text and data fields that will comprise the form. The easiest way to do this is to use an authoring tool that is already integrated with Acrobat Distiller, such as Adobe FrameMaker®, Adobe Illustrator®, Adobe InDesign®, Adobe PageMaker®, Adobe GoLive®, Microsoft Word®, or Microsoft Excel®. Use the authoring tool to lay out the page or pages of the form, including headings, instructions, and any fields in which users will enter data. At minimum, each field should be provided with a label, although you can also draw lines or boxes next to the labels to indicate where the fillable field itself will go. Use graphics tools to draw lines and boxes -- do not use characters such as underlines and vertical bars.

2. Convert the original document to PDF.

Use whatever method the authoring tool provides to generate a PDF version of the original file. In some authoring tools, this is a Save As operation, and in others a separate tool or menu selection. It is important that you save the file as a **non-Tagged** PDF file. Do do this:

- In the PDF Setup dialog, which will appear prior to the generation of the PDF file, select the Tags tab and make sure that *Generate Tagged PDF* is *not* checked.

[Instructor Notes: make sure students understand point 2. above. Forms Access cannot tag a previously tagged file. If you try to load one, a File has already been tagged by another application error message will appear. If you re-load a file that has already been tagged \(Finished\) with Forms Access, it will load but significant information will be missing in the Structure Tree.](#)

3. Add fields to the PDF document as follows:

- Open the PDF file in Acrobat, and select the Forms Tool.
- Use the cursor to draw a rectangular box at the position of each field in the document.
- The Field Properties dialog box will appear -- if it does not, right click on the field and select Properties to bring up the Field Properties dialog.
- From the Field Properties dialog, select the desired characteristics for the field. Options include:
 - Name of the field.
 - Field Type, which can be Text, Combo Box, List Box, Radio Button, Check Box, and so on. Selection options can be provided for Combo and List Boxes.
 - An optional Short Description can be provided. The Short Description is visible during mouse over and is also accessible by screen readers. If you provide a Short Description for the field in Acrobat, it will be carried over when the form is processed by PDF Forms Access, or you can wait to create the Short description after Forms Access has initialized the form.

Save the document. Click on the hand tool to return the document to normal view. Verify that Text fields can be filled in and that Combo or List Boxes can be opened and the various options selected.



Exercise: PDF Form Field Properties

In order to do any of the exercises in this module, you must have Acrobat 5 or higher installed on your machine, and you must have copied the sample files in the archive **FormsAccessExamples.zip** to a folder on your machine.

This first exercise will consist of a tour of a simple PDF form. The PDF document was created from an original document authored with FrameMaker 7, and was created as a *non-Tagged* PDF file. Some fields have already been added to the form using Acrobat 5.05. You will add a few more fields to complete the form. In the process, you will enter information in the Field Properties dialog.

During this exercise you will use the Form Tool to examine the Field Properties of existing fields in a PDF form. You will also use the Form Tool to add a new Text field and a new Combo Box field to the form.

Opening and Inspecting the Sample Form

1. **Open** the sample file **PersonalData_unfinished.pdf**.

This is a bare bones form file. It has fillable *Text* fields for *First* and *Last* Names, *ID Number*, and *Office Number*. It also has a drop down *Combo Box* for the *Manager* field.

2. Put the **Hand** cursor on the horizontal line just to the right of the First Name field.

Notice that it changes to a vertical I-bar, indicating that text can be typed into the document at that location.

3. **Click** on area to the right of the First Name label, and type your first name there. The font will be blue.
4. **Click** on the down arrow to the right of the Manager label.

A menu of selection options will drop down. Select an option and notice that it is entered in the Manager field.

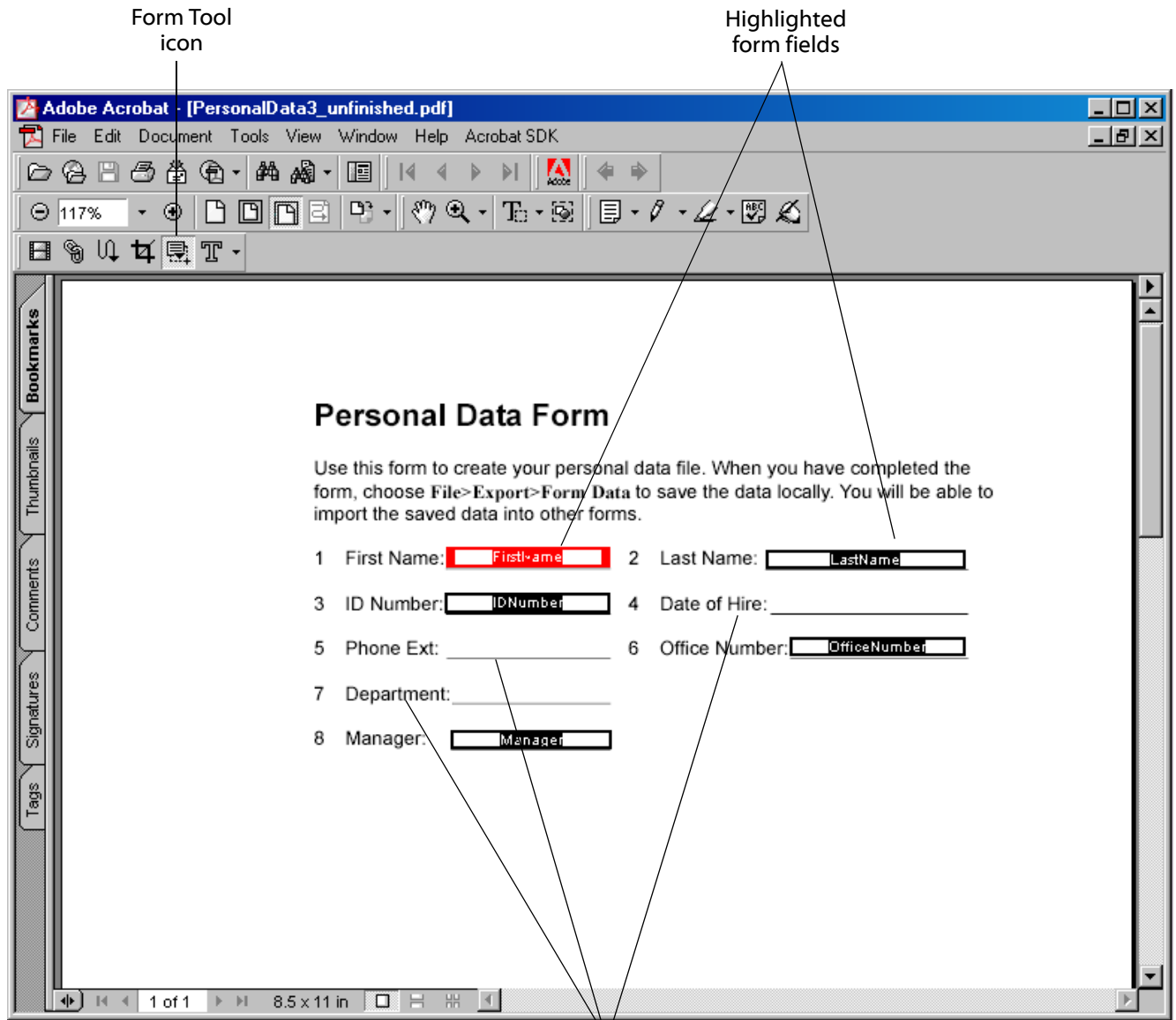
Three of the field labels, *Date of Hire*, *Phone Extension*, and *Department*, do not actually have fields associated with them.

Notice that the Hand cursor does not change to a vertical I-bar when placed to the left of the Text field labels, and that there is no drop down list associated with the *Department* field.

Examining Field Properties

1. **Click** on the **Form Tool** button  in the Toolbar.

Rectangular boxes will appear around the fields in the form. The label in the center of each box is the *Name* attribute of the field.



Field labels and place-holders. Field elements will be added here using the Form Tool.

The *Date of Hire*, *Phone Extension*, and *Department* labels do not have associated data fields and are not highlighted with boxes.

2. **Left click** on the box around the *Name* field in the document. The *Field Properties* dialog will appear. Note the following:

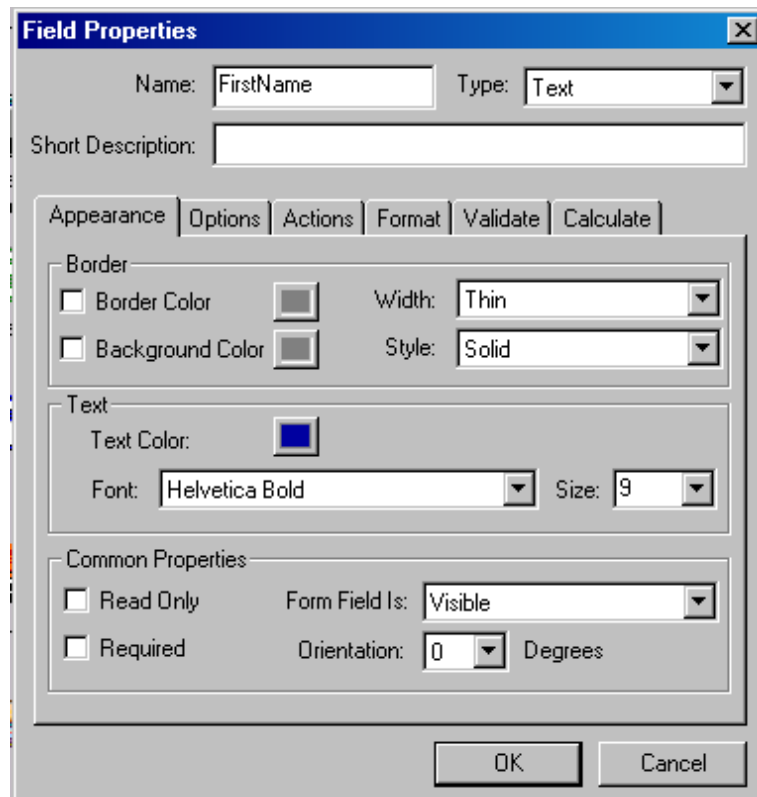
The *Type* of the field is *Text*. This specifies that the field will be a fillable, editable field in the form into which users can type data.

The *Name* field contains the label *FirstName*. This is a required field.

The *Style* is *Solid*, with no *Border* or *Background* color. This will have the effect of making the data fields themselves transparent, allowing the

underlines to the right of each field in the original document to mark the position of the data entry fields.

Any text in the *Short Description* field will appear whenever the mouse cursor is placed over the field, and it will also be read when a screen reader processes the document.



Other attributes in the Field Properties dialog can be edited and modified to alter the way the field looks or behaves in the form.


3. **Cancel** the Field Properties dialog, then **left double click** on the box around the **Manager** field and select **Properties** to bring up the Field Properties dialog again.

Note the following:

- The *Type* of the field is *Combo Box*.
- In the *Options* tab there is a list of selection options. This list specifies the options list that appears when the user clicks the arrow at the right of the Combo Box field. The item in the list in the Options tab that is highlighted will appear as the default item in the field.
- This field also has a Name and a Short Description.

Adding Fields to a Form

The form contains labels and placeholder lines for three fields that have not been created yet: *Date of Hire*, *Phone Ext*, and *Department*. You will use the Form Tool to add these fields to the form.

1. **Click** on the **Form Tool** icon .
2. Position the cross hair cursor on the underline to the right of the **Date of Hire** label.

Hold down the **left mouse button** and draw a rectangle of the same length as the underline and roughly the same height as those in the existing fields.

Release the mouse button. A rectangle will appear where you drew the field and the Field Properties dialog will appear.
3. In the **Name** field of the Field Properties dialog, type *DateOfHire* (this is an arbitrary name).
4. In the **Type** field, select *Text* if it is not already selected.
5. In the **Short Description** Field, type *Use month / day / year format*.
6. **Click** on the **Appearance** tab.

From the **Style** drop down list, select *Solid*.

Uncheck the **Border Color** and the **Background Color** check boxes.
7. **Click** the **Format** tab.

In the **Category** box, select **Date**.

In the **Date Options** box, select **1/3/81**. This will prompt for the correct format if the user enters an incorrect one despite the Short description instructions.
8. **Click OK**.
9. Repeat **Steps 2, 3, 4, 6, and 8** with the appropriate changes, for the *Phone Ext* field. Leave the Short Description field blank.
10. **Click** on the **Format** tab.

Select **Number** from the Category box.

Select **0** in the Decimal Places field.
11. **Click** on the **Options** tab.

Check the **Limit of** check box and enter **3** in the **Characters** field.
12. Click **OK**.
13. Position the cursor to the right of the **Department** field and draw a rectangular box, as in **Step 2**.
14. Select **Combo Box** from the **Type** drop down menu.

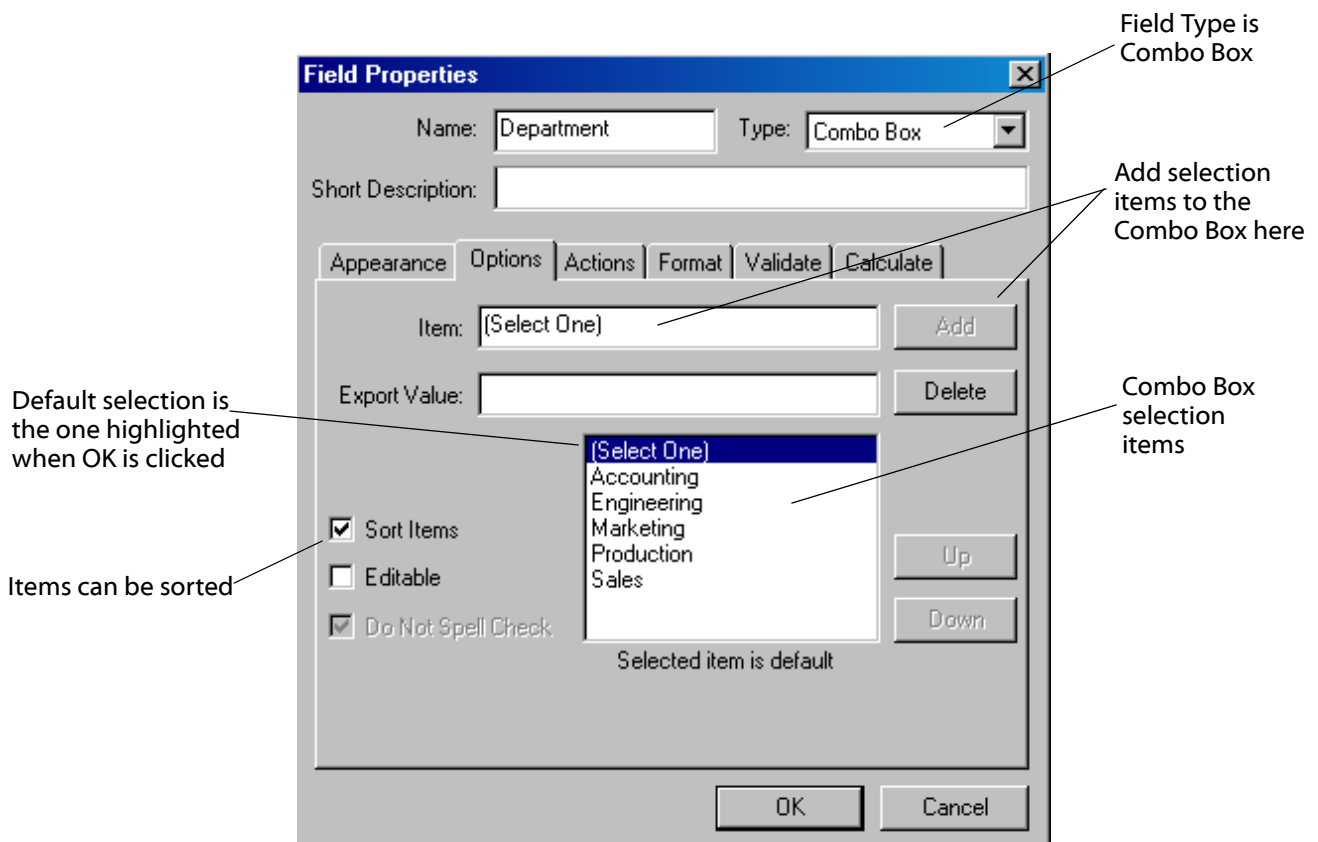
15. Type *Department* in the Field Properties dialog **Name** field.

16. Click on the **Options** tab.

Use the following procedure to add selection options to the drop down menu that will be displayed, or read by a screen reader, when the user selects the Combo Box on the form:

- Type a selection option in the **Item** field in the Options tab, and click the **Add** button. For consistency with the other Combo Box in the form, type *(Select One)* in the Item field and click the Add button. This will add your entry to the edit box in the Options tab of the Field Properties dialog.
- Repeat the above procedure for the following selection options: *Accounting, Engineering, Marketing, Production, and Sales*, clicking the Add button after each. These options are arbitrary: use them or any other options you wish.
- Make sure that *(Select One)* in the options list is highlighted. This will make it the default selection for the Combo Box in the form document.
- Check the Sort Items check box.

17. Click **OK**.



18. Last but not least, set the *tab order* for the fields. This is important because screen readers will jump from field to field in the form each time the Tab key is pressed. The cursor will move forward through the field tab order when the Tab key is pressed and backwards through the tab order when Shift-Tab is pressed. The tab order should reflect the logical reading order for the fields, typically in the order of their IDs.

Instructor Notes: Getting the field tab order right is important for accessible forms. The initial tab order is the order in which the fields were initially created. Since the fields in the sample form were not created in the logical read order, the tab order in the sample form will be wrong.

From the Acrobat Menu bar, select **Tools > Forms > Fields > Set Tab Order**.

Numbers will be displayed in the upper left corner of the borders around the fields. These numbers represent the tab order.

19. No matter what the current tab order is, do the following:

Start with the field that logically should be read first: in this case, field **1**, the *First Name* field, and **click** on it.

Now, **click** on the next field in the logical sequence, field **2**, the *Last Name* field.

Continue, clicking inside the borders of fields **3, 4, 5, 6, 7, and 8**, in that order.

Check the numbers in the boxes: they should now be in the proper order from one to eight.

20. Select **File > Save As** from the menu bar, and save the file as **PersonalData_1.pdf**.

If you were not able to complete the form, you can use the file **PersonalData.pdf**, which is included in **FormsAccessExamples.zip**. **Open PersonalData.pdf** in Acrobat and **Save it As PersonalData_1.pdf**, for use in the exercises that follow.

21. Click the **Hand Tool** to switch to normal viewing mode.
22. Test the new fields by typing something into the Text fields and by selecting an item from the Department field.



Summary

In this module, *Introduction to Accessibility Issues*, you considered some of the challenges involved in making electronic documents accessible to visually disabled readers. You learned how Adobe Tagged PDF facilitates producing accessible documents, and what Adobe PDF Forms Access does to facilitate making forms documents accessible. You reviewed editing and creating fields in Adobe Acrobat PDF form documents.

In the next section, you will tour PDF Forms Access and use it to tag a simple form.

Creating Accessible Forms with Forms Access



Introduction to PDF Forms Access

Starting with this module, you will explore the PDF Forms Access interface, become familiar with its operation, and use PDF Forms Access to process a simple form file. In the module following this one, you will examine the resulting tag structure, make necessary modifications to it, and will be introduced to methods for testing accessible forms, including screen readers. The third module will cover guidelines for forms layout, and will include examples of how Forms Access analyzes different form layouts. In the fourth module, you will use PDF Forms Access to create form tables. The last module covers importing structures from other documents, miscellaneous advanced features of PDF Forms Access, a concluding exercise, and a summary of the course. Each section will be accompanied by a hands on exercise illustrating its main points.

Learning Objectives

At the conclusion of the course, you will:

- Be familiar with the PDF Forms Access interface, and with the contents and uses of the Work Area, Structure Tree Pane, Page View Pane, and Properties Palette.
- Be familiar with how PDF Forms Access initializes a form.
- Understand what kinds of modifications can and should be made to the initial tag structure to make the form fully accessible, including:
 - How to add, move, and delete element tags in the Structure Tree.
 - How to adjust text block boundaries by splitting or combining text elements.
 - How to mark superfluous text as artifacts to eliminate it from the logical read order.
 - How to add and modify alternate text descriptions for field elements.
 - How to create element Stereotypes.
- Know how to save a form temporarily and how to finish a form.
- Be familiar with methods for testing accessible form documents.
- Be aware of how different form layouts can affect PDF Forms Access initialization.
- Know how to create and modify form tables.
- Understand how to import a structure into a form document from another form.

Module Contents

Topics	Exercises
Overview of PDF Forms Access	Initializing a Form Using PDF Forms Access
Modifying the PDF Forms Access Structure Tree	Adding Elements to the Structure Tree
Adding Accessible Text to Form Fields	Adding Accessible Text to Form Fields

File to Download for Exercises in this Section

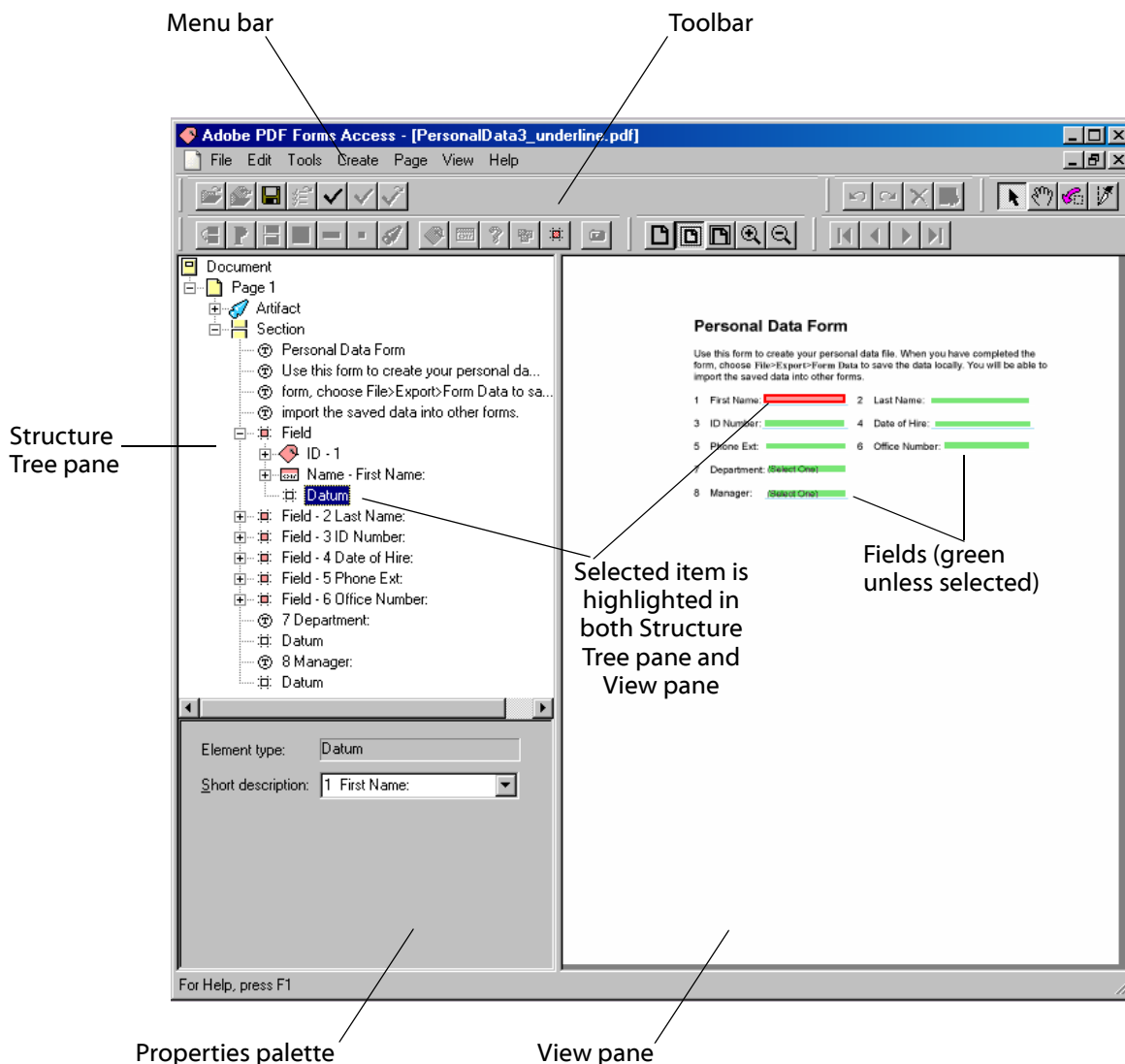
FormsAccessExamples.zip, which is normally found in the Exercises folder, contains the following files:

- **PersonalData.pdf**
- **PersonalData_underline.pdf**
- **PersonalData_unfinished.pdf**
- **ProblemForm1.pdf**
- **ProblemForm2.pdf**
- **ProblemForm2mod.pdf**
- **ProblemForm4.pdf**
- **Table1.pdf**
- **Table2.pdf**
- **Table3.pdf**
- **Table1_ext.pdf**
- **Table1_org.pdf**
- **Table1_org-new.pdf**
- **Jaws1.wav**
- **Jaws2.wav**
- **form_1040.pdf**
- **f1040_accessible.pdf**
- You should extract these files from the zip archive if they have not been extracted already.



Overview of PDF Forms Access

When you start PDF Forms Access, the work area is initially blank. When a file is loaded using File > Open, it is automatically processed and the elements in the form are given tags which appear in the Structure Tree pane at the left of the work area. The document itself appears in the View pane at the right of the work area. A Properties Palette is displayed at the lower left. The Menu bar contains menus, organized by topic, for performing various tasks. The Toolbar has buttons for common commands, enabling you to select commands quickly as you need them. Place the mouse cursor over a button's icon to see a description of its function.



The Work Area

The Work Area consists of a View pane, a Structure Tree pane, and a Properties palette for editing and validating the document's structure.

- The Structure Tree pane displays the tag hierarchy that corresponds to the elements in the document.
- The Page View pane displays the current page of a document.
 - Fields in the document are highlighted in green.
 - Selected elements in the document are highlighted in red.
 - Artifacts, which are generally non-text elements, are highlighted in blue.

When an item is selected in the View pane, its corresponding tag is highlighted in the Structure Tree pane.

- The Properties palette displays various attributes about a document element.

The Structure Tree Pane

The Structure Tree pane displays a tag hierarchy, in the form of a tree control, that represents the structure of the elements in the document. Refer to the next screen shot displaying the Structure Tree just after initial processing of the sample file `PersonalData_1.pdf`, that you created in the previous exercise. A Document node is at the root of the tree, and it contains a Page node for each page in the document. Each Page node in the tree contains other tag nodes that represent elements in that page of the document.

A node in the Structure Tree that contains one or more nested nodes is called a *parent node* -- or a *parent tag* or a *parent element* -- and the nested nodes are called the *children* (or child nodes) of the parent node. Child elements on the same level under a parent element are called *siblings*. Some attributes of a parent element, such as language, are inherited by the parent element's children. If a parent element is moved or deleted, its children are moved or deleted as well.

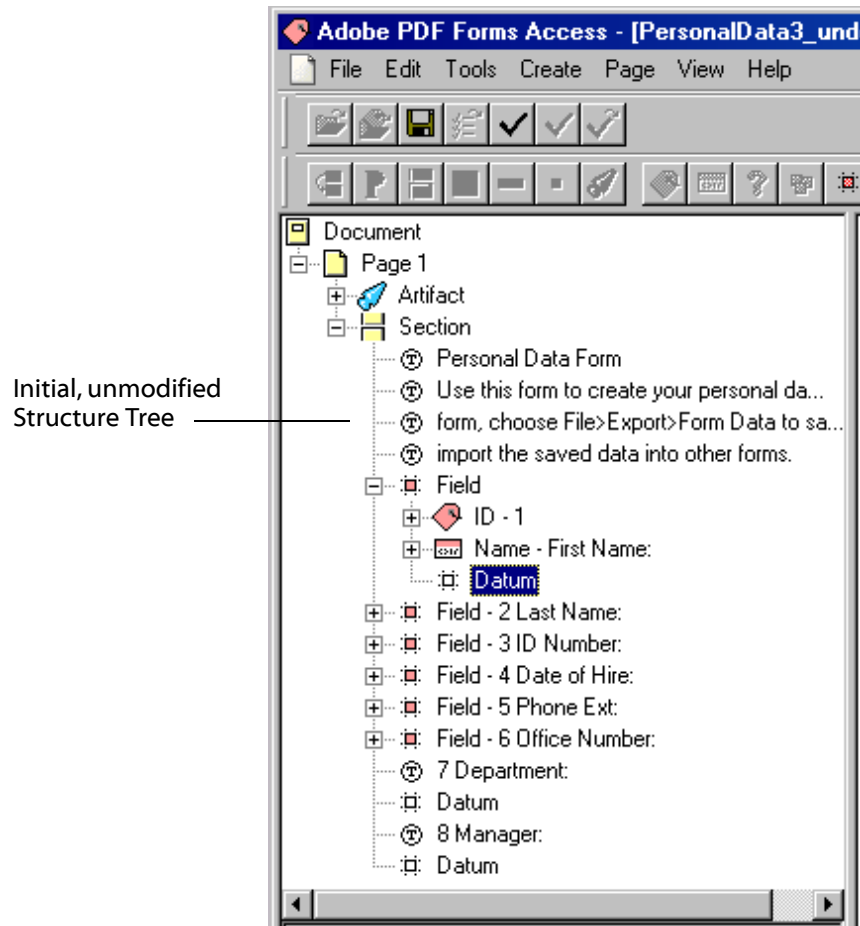
A plus sign (+) next to a node in the Structure Tree indicates that the node has children that are currently hidden. Clicking the plus sign displays all of the children on the next level. If a child node has a plus sign next to it, the child node also has children. Open all levels of the child node before changing properties or deleting a node to see which other lower-level child elements may be affected by any changes. The Page node is expandable only for the current page. When you work on one page, you cannot expand nodes for other pages.

Form File Initialization and Initial Tag Structure

When you open a non-Tagged PDF form, PDF Forms Access initially identifies elements in the document such as graphic artifacts, document sections, text blocks, and fields, and automatically defines an initial set of tags based on that analysis of the page layout. In most cases, however, you will need to make at least some further adjustments and refinements to the Structure Tree in order to make the form logically or fully accessible.



Acrobat Capture® Note: Although *Tag PDF* appears immediately after *Tag Form* under *Step Templates* in Acrobat Capture, *do not* use Tag PDF when creating accessible PDF forms. Tag PDF is for use with PDF documents, and Tag Form is for use with PDF forms.



PDF Forms Access allows you to add other types of tags to the Structure Tree, and to delete, modify, and rearrange existing tags. The objective is to create a Structure Tree hierarchy, which determines the read order of the document, that accurately and clearly reproduces the meaning and the intended logical flow of the visual document. The visual representation of a form and the Structure Tree representation of a form are separate but parallel structures. Each might contain elements not found in the other -- the visual form may contain graphical artifacts that are meaningless in the logical read order, and its Structure Tree may contain audible instructions and the like for visually impaired readers that are not found in the visual representation of the document.

In general, the relevant content, read order, and logical processing order of the form should be equivalent in both representations of the document. Having said that, there are always exceptions. For example, visual techniques might be used to make a certain piece of information in a document -- a warning, say -- stand out to the reader, so that it is noticed first almost regardless of where in the page it is placed. There would be a case for placing that piece of information first in the Structure Tree, no matter where it is placed visually in the reading sequence of the document. In this respect, the visual layout of the document and the layout of the Structure Tree are independent of each other.

Structure Tree Tags

Adobe PDF Forms Access uses the following tag types in the Structure Tree. Some of these tags are created automatically by PDF Forms Access at initialization time, and others can be added by users.

- **Document**
A container for the entire multi-page form.
- **Artifact**
Portions of a document that are not defined for accessibility. For example, graphical lines used within a table.
- **Page**
A container for a single page of a form.
- **Section**
Typically, a container for multiple higher-level elements such as Fields or Field Groups.
- **Table**
A table consisting of Table Rows and Table Cells. A Table Row consists of one or more Table Cells. A Table Cell can hold any element except another Table or Table Cell.
- **ID**
A short text identifier, such as *Part IV*, or a number. An ID node can contain only Text elements.
- **Name**
A text identifier that identifies a field on a PDF form. For example, *Farm income or (loss)* identifies a field on an IRS form. In the Structure Tree, a Name node can contain only Text elements.
- **Instruction**
Instructions pertaining to a Section, Field, or Field Group. In the Structure Tree, an Instruction node can contain only Text elements.
- **Field**
A Field can contain one ID, one Name, one or more Instructions, and one or more Datum.
- **Field Group**
A container for a group of form fields and descriptive text.
- **Paragraph**
A paragraph of text. In the Structure Tree, a Paragraph node can contain only Text elements.
- **Figure**
A figure contains text or graphics.
- **Text**
The content element of a text object.

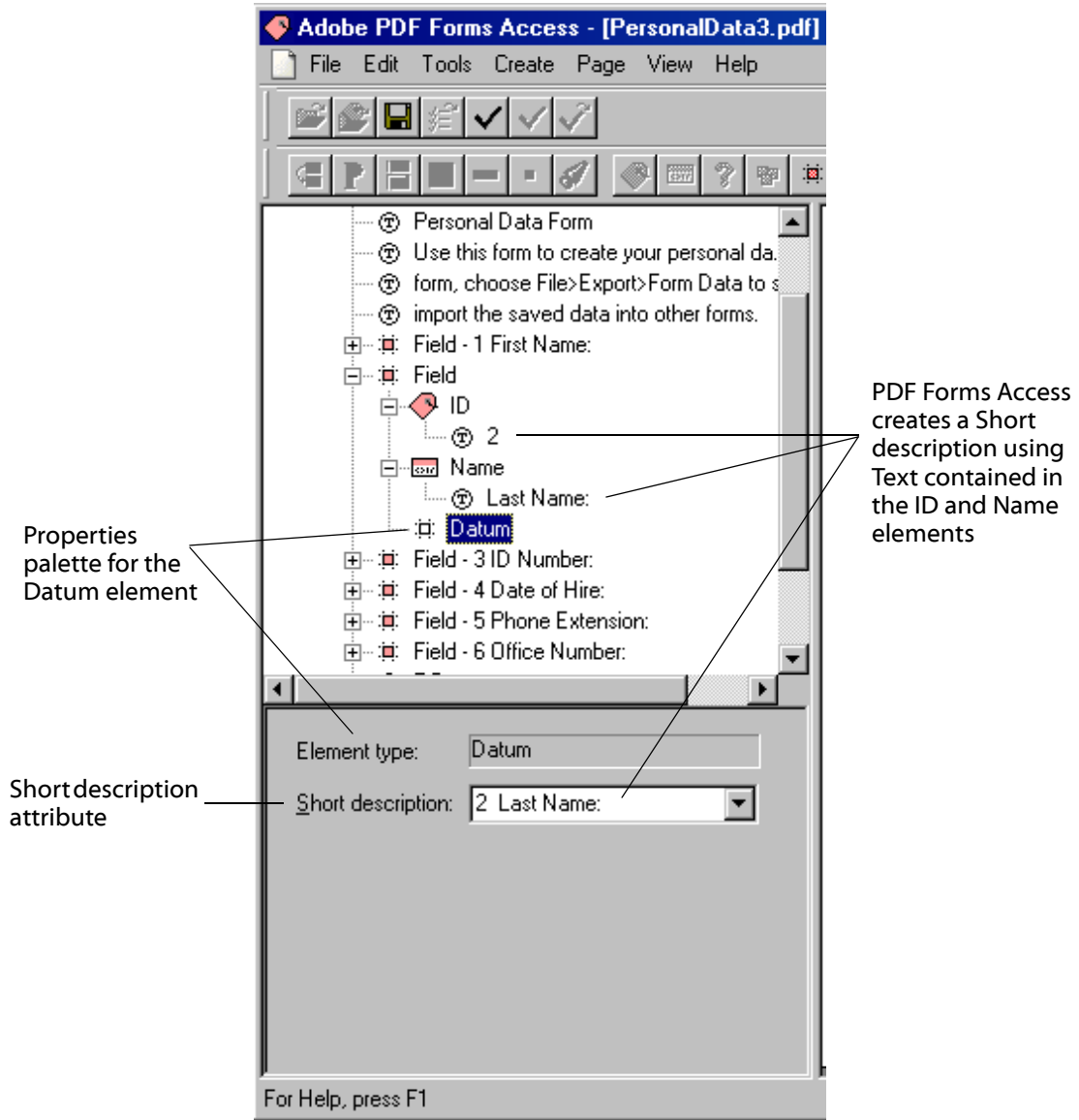
- **Datum**

The element that holds user input in a form fill-in field.

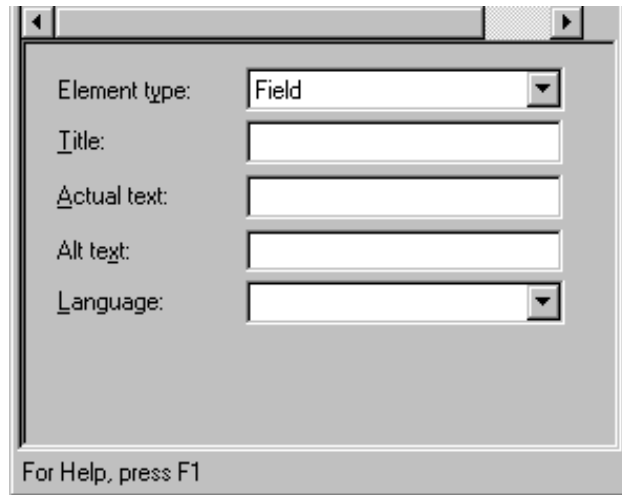
Certain tag elements, such as Table and Field, contain other child tag elements. These are referred to as template elements or **Stereotypes**. When you add a Stereotype tag to the Structure Tree, its child elements are automatically added along with it. For example, when you add a *Field* tag to the Structure Tree, *ID*, *Name*, and *Instructions* tags are also added as its child elements. When PDF Forms Access automatically creates a Field tag during initialization, however, it does not necessarily add the tag's children even though the Field element is a stereotype: Forms Access may create the Field tag with a Datum child element only, or may create a Field tag with Name and Datum child elements only, depending on layout and formatting characteristics of the form. There will be some discussion at a later point of how layout and formatting affect the way PDF Forms Access analyses elements in a form.

The Properties Palette

The Properties palette displays various attributes associated with tag elements in the Structure Tree. Tag element attribute values can be added or edited in the Properties palette. The Attributes that are available in the palette change according to what type of tag element is currently selected in the Structure Tree. A *Datum* element, for example, has only one editable field, *Short Description*, as you can see in the following screen shot. In the screen shot example, PDF Forms Access has automatically created a Short description for the Datum element by combining the text values of the *ID* and *Name* elements. This will happen automatically if PDF Forms Access successfully creates ID and Name elements for a field during initialization. You can modify the existing Short description or change it completely. If you created a Short Description for a field when you first created the form in Acrobat, that description will be transferred to the Short description attribute of the corresponding Datum element when the form is initialized by PDF Forms Access.



A different set of attributes is displayed in the Structure Tree when a Field element is selected, as illustrated in the screen shot below.



Tag element attributes have the following characteristics:

- **Element Type**
Defines the element tag type that appears in the logical structure tree.
- **Title**
Defines text that will appear next to the element tag in the Structure Tree. The Title has no direct affect on accessibility, but if you provide a descriptive value for a tag's Title attribute, it will make the Structure Tree easier to read and understand.
- **Actual Text**
Actual Text can be used for words in a document that are actually graphics rather than text characters, and are therefore not readable by a screen reader. These will generally be Figure elements. For graphics, the Actual Text should be an exact replication of the graphically rendered words or characters. Actual Text can also be used in Field elements to provide labels for fields that will be read by a screen reader. This is sometimes easier than sorting and modifying the Text tags in the Structure Tree which represent the original labels in the PDF form. The Actual Text will be spoken by a screen reader when the screen reader encounters that element in the logical flow.

[Instructor Notes: When to use Actual Text versus Alt Text is a little more ambiguous with form fields than it is in regard to graphics. Either one will be read by screen readers. Do not use both at the same time in a single element.](#)

- **Alternate Text**
Alternate Text is used to describe pictures, figures, graphs, and other images that appear in a document but whose presence is not indicated by screen readers. Alternate Text provides an audio caption for those who cannot see the image. The Alternate Text should describe the meaning that the graphic image is conveying. It can be of some length. Alternate Text is most often used with Figure elements, but can also be used to supply alternate labels or instructions for Field elements.

- **Language**
Identifies the language in which the document is written. In addition, individual elements that contain content in a language other than the main document language may be set to a different language to tell the screen reader to use an alternate pronunciation scheme or to identify alternate hyphenation schemes for various languages.
- **Short Description**
The Short Description provides a description of a Datum, or field, element. It is read by the screen reader during normal processing, and also when the user tabs to the field. PDF Forms Access creates default Short Descriptions from a combination of ID and Name attributes, although you can select different formats as well as supply a Short Description of your own. Short Descriptions that were created for fields in Acrobat will be used as the value of the Short Description attribute for the corresponding Datum elements that PDF Forms Access creates.
- **Cell Heading**
A property for a Table Cell element.
- **Span**
A property for a Table Cell where cells span rows or columns.



Exercise: Initializing a Form Using PDF Forms Access

In order to do any of the exercises in this module, you must have Acrobat 5 or higher and PDF Forms Access 1.0 or higher installed on your machine, and you must have copied the sample files in the archive **FormsAccessExamples.zip** to a folder on your machine.

In this short exercise, you will do the following:

- initialize the **PersonalData_1.pdf** form you created in the first exercise by opening it in PDF Forms Access.
- Inspect the Structure Tree tags and see the relationships between them and elements in the Page View pane.
- Note places where the initial tag structure needs adjustment, and where accessible forms of text should be added to form fields.

A discussion of Structure Tree tag characteristics will follow this exercise. You will then make the required post processing adjustments to the Structure Tree in the exercise after this one.

1. **Run** PDF Forms Access.
2. Select **File > Open**, browse to the sample file folder, and open the **PersonalData_1.pdf** file you saved in the first exercise.
*Note: if you did not finish adding all fields to the sample file during the last exercise, open the file **PersonalData.pdf** in Acrobat, then Save it As **PersonalData_1.pdf**. Repeat step 2.*
3. PDF Forms Access will structure the file, and will display the document in the **View pane** with all fields highlighted in color.
4. Click on one of the fields in the View pane. The tag hierarchy in the **Structure Tree pane** will expand, and the tag corresponding to the field you clicked will be highlighted.
5. Click on a tag in the Structure Tree. The corresponding element in the View pane will be highlighted.

Spend a few minutes exploring the tag structure by selecting various elements in the Structure Tree and View panes. As you do this, keep in mind that the top-to-bottom sequence of tags in the Structure Tree is what determines the read order of the document when processed with a screen reader.

6. Note that the **Properties** palette options change as you select different tags.
7. There are several Field parent elements in the Structure Tree. **Click** on one of them. You will see the attributes associated with the Field element in the Properties palette: *Element type*, *Title*, *Actual text*, *Alt text*, and *Language*. Actual text and Alt text can be used to provide supplemental accessible text that is spoken when a screen reader encounters a form field. You will see examples of how to do this later.

8. **Click** on the **Section** element at the top of the Structure Tree. In the View pane, a red highlighted box will enclose all the elements in the document because the Section element is a parent of all other elements on the Page except the Page itself.
9. Select **View > Outline Children**. The children elements of Section will be outlined with a red, dashed line.

Click one of the child elements of the Section tag. Select **View > Outline Parent**. A red border will enclose all the children of the Section element.

These tools will become more useful as the complexity and size of the form increases.

10. **Click** on one of the **Field** elements to expand it. Also expand all the child elements of the Field element.

As you click on the Field element and its children in the Structure Tree, notice how the corresponding items in the View pane are highlighted.

Also notice the attributes displayed in the Properties palette for each Structure tree element.

11. Select a **Datum** element from the Structure Tree and examine the Properties palette.

Note that the **Element type** cannot be changed.

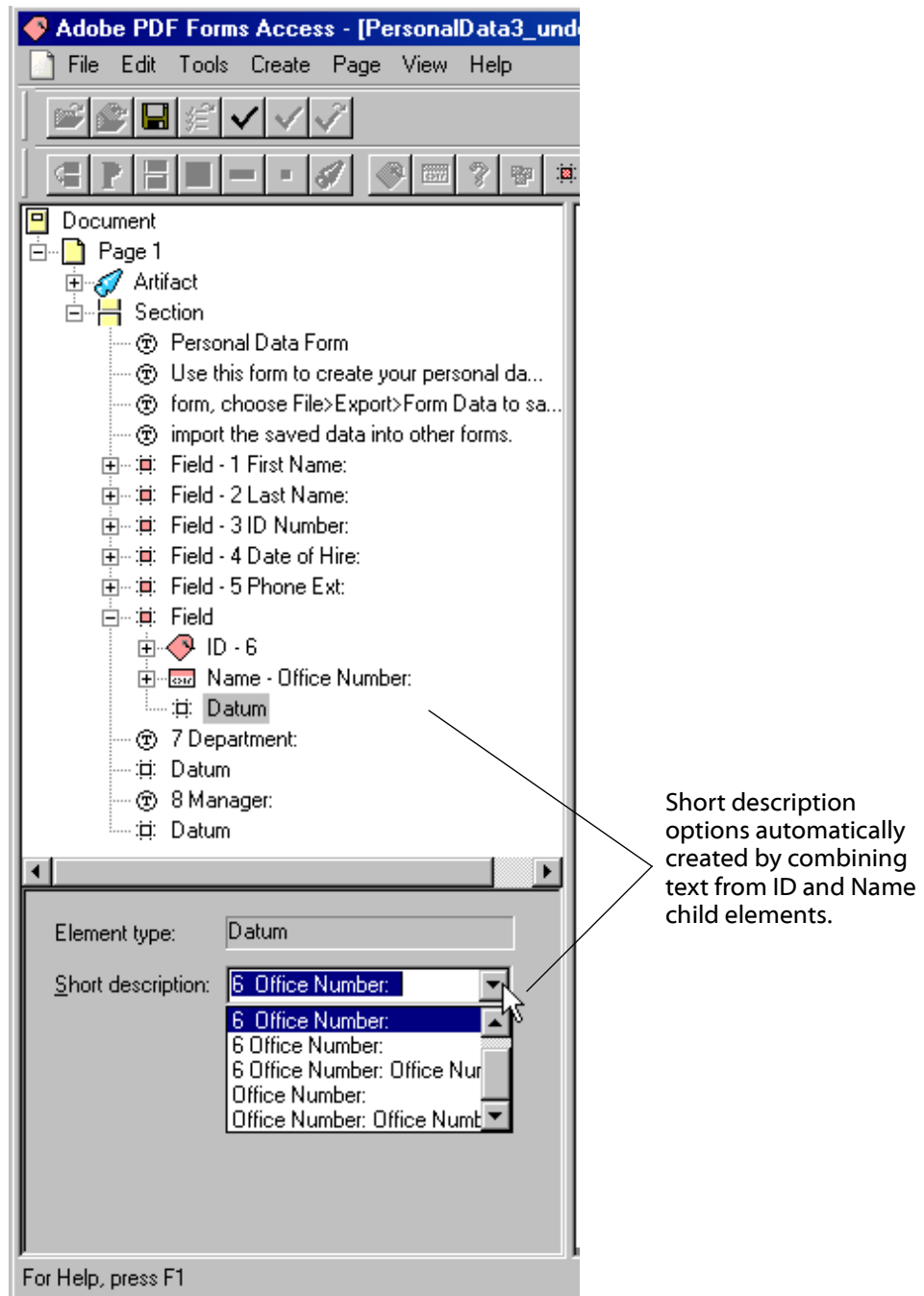
Notice that a **Short description** has been created by combining the text contained in its ID and Name sibling elements. The Short description is the most important accessible text associated with a form field. It is read by screen readers in both normal reading mode and when tabbing through fields.

PDF Forms Access will create variations of the Short description for a Datum element contained in a Field. **Click** on the Short description drop down list to select one, as in the screen shot below.

Existing Short descriptions can be modified, and missing Short descriptions can be added. If you included a **Short Description** for a field when you initially created the form in Acrobat, that description will be transferred to the **Short description** attribute of the Datum element.

12. **Click** on the **Artifact** element at the top of the Structure Tree, then expand it.

PDF Forms Access has placed all the graphic elements in the form document -- the underlines and solid rectangles associated with the data fields -- as children of the Artifact element. Artifact elements and their children will be ignored by screen readers.



- Finally, notice that the Datum elements for the last two data fields, the Combo Boxes for Department and Manager, have not been wrapped inside Field tags as the other data fields were. PDF Forms Access could not successfully analyze the structure of these fields and their labels, and so it included them verbatim in the Structure Tree. You will add Field tags for these Datum elements in a later exercise, and will also supply their missing Short descriptions.

PDF Forms Access will not automatically create Field elements and their children for fields in a form that are not *Text* type fields. The reason for this lies mostly in the historical genesis of Forms Access, which was originally developed to process IRS forms. Almost all data fields in IRS forms are of the text entry type. If you wish to create Field elements and their children for fields other than text type fields, you will have to do it manually. Fortunately, Forms Access makes this easy.

Instructor Notes: Forms Access 1.0 does have some limitations in regard to recognizing field constructs upon initialization -- that is, recognizing that an optional ID, a label, and a data field are related as a group -- depending on how the fields and their components are positioned in the form. To a great extent, this is because version 1.0 is the first release, and because its development was based on work with IRS forms. Successive versions of Forms Access will no doubt have extended pattern recognition capabilities. Try to focus on workarounds for Forms Access limitations -- all of which are, in fact, very easy -- rather than on the limitations as problems with the software.



Modifying the PDF Forms Access Structure Tree

Form documents tend to have relatively complex layouts compared to documents with a simple, single column structure like the one you are now reading. Form documents will frequently use multiple columns and multiple line text blocks to provide instructions for individual fields. Both instructions and labels can be placed in various locations relative to the fields they reference -- above, below, right, left, or across the page connected by ellipses, for example. For this reason, the initial PDF Forms Access processing of the document will generally produce a Structure Tree hierarchy that does not completely reflect the logical read order of the form.

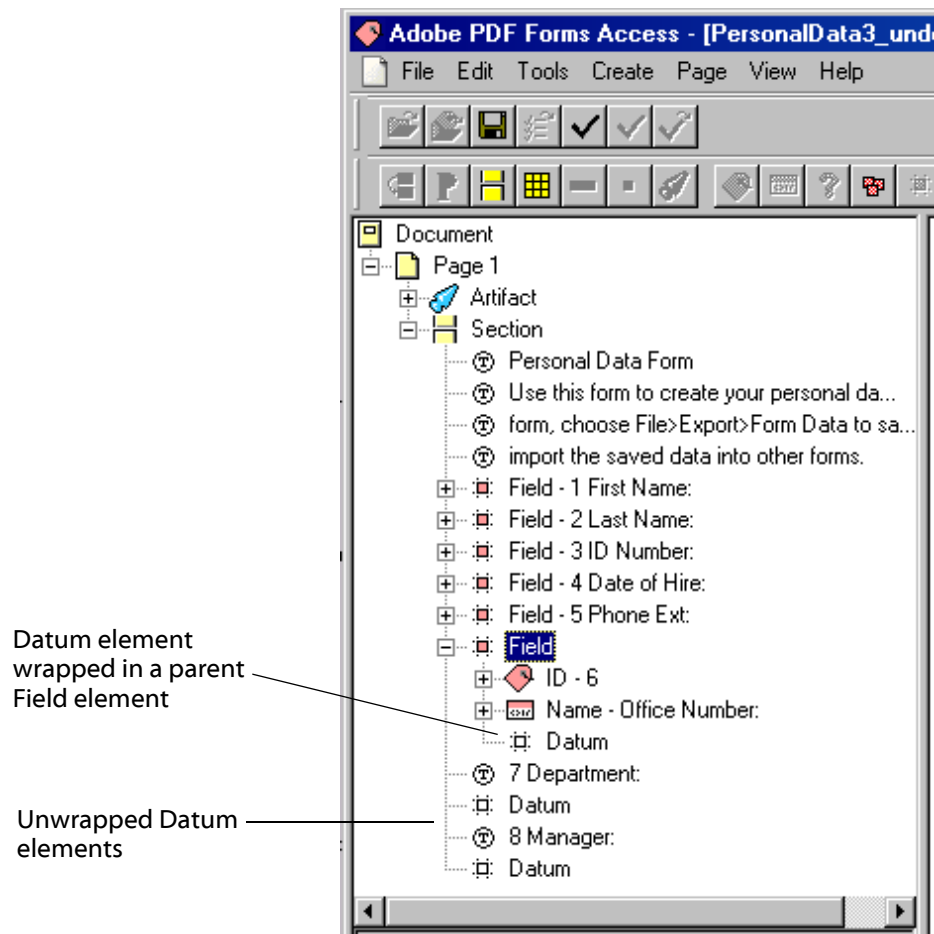
To a large extent, the success PDF Forms Access meets with in analyzing and tagging the form depends on the original formatting and layout of the form and the types of fields that are used in it. A later section will discuss some of these issues. Sometimes the layout of the parts that make up the form field in the original document -- the identifying number, the label, special instructions, and so on -- will cause problems with the way the elements in the Structure Tree are sequenced. The top-to-bottom order of document elements as they are represented in the Structure Tree determines the order in which the document elements will be read by a screen reader. At other times, PDF Forms Access will not recognize that a particular group of items in the form go together to constitute a field. Various methods are available for fixing these types of problems in the initial Structure Tree, and will be covered as the course proceeds.

Tag Sequence in the Structure Tree

The next screen shot is of the Structure Tree immediately after initializing the sample form. When the sample file was originally constructed, certain formatting procedures were followed that were known by experience to produce a correct, or nearly correct, sequence of tags in the Structure Tree. If PDF Forms Access initialization does not generate a sequence of elements in the Structure Tree that reflects the logical read order of the form, you will have to manually correct it. Fortunately, there are several relatively easy ways to do this. Even more fortunate, however, is that it is not necessary in this case. A quick inspection of the Structure Tree shows that its elements are in exactly the same order as the form would be read and processed by a sighted reader. Later, you will have a chance to correct the Structure Tree tag sequence of some not so carefully formatted documents.

Adding Elements to the Structure Tree

A quick glance at the Structure Tree also shows that PDF Forms Access treated the last two fields in the form, the Combo Boxes, differently than the other fields. PDF Forms Access recognized that fields 1 through 6 from the document were comprised of a number, a label, and a Text type data field, and subsequently created parent Field elements for each of them, with ID, Name, and Datum child elements. The Combo Box fields, on the other hand, were recognized as Datum elements, but were not recognized as being associated with a number and a label in a Field. You will fix this problem in the next exercise by manually wrapping each of the Datum elements with a parent Field element.





Exercise: Adding Elements to the Structure Tree

In order to do any of the exercises in this module, you must have Acrobat 5 or higher and PDF Forms Access 1.0 or higher installed on your machine, and you must have copied the sample files in the archive **FormsAccessExamples.zip** to a folder on your machine.

When you initialized **PersonalData_1.pdf** in the last exercise, PDF Forms Access performed the major task of creating Structure Tree tags for the form fields in the document. Your task now is to verify that the data fields and the other elements in the document, such as introductory text, field labels, instructions, and so on, are arranged in the Structure Tree to reflect the logical read order of the document, and that the steps required to complete the form are comprehensible to a visually impaired reader.

PersonalData_1.pdf consists of a heading, a text block, and eight fields with identifying numbers and short labels. As you saw earlier, the sequence of elements in the Structure Tree accurately reflects the logical read order of the form, so no adjustments need to be made to the form in that respect. However, for the sake of consistency, as well as practice, you will add Field elements to the Structure Tree and wrap the last two Datum items with them. In the process, you will add accessible text to Structure Tree elements, and will add a child element containing accessible special instructions to one of the Field elements.

At the conclusion of this exercise, you will have done the following:


- Added a parent Paragraph element to the Structure Tree and used the Move Tool to move text elements into it as children.
- Added parent Field elements to the Structure Tree for two Datum elements and created accessible text for the Fields.

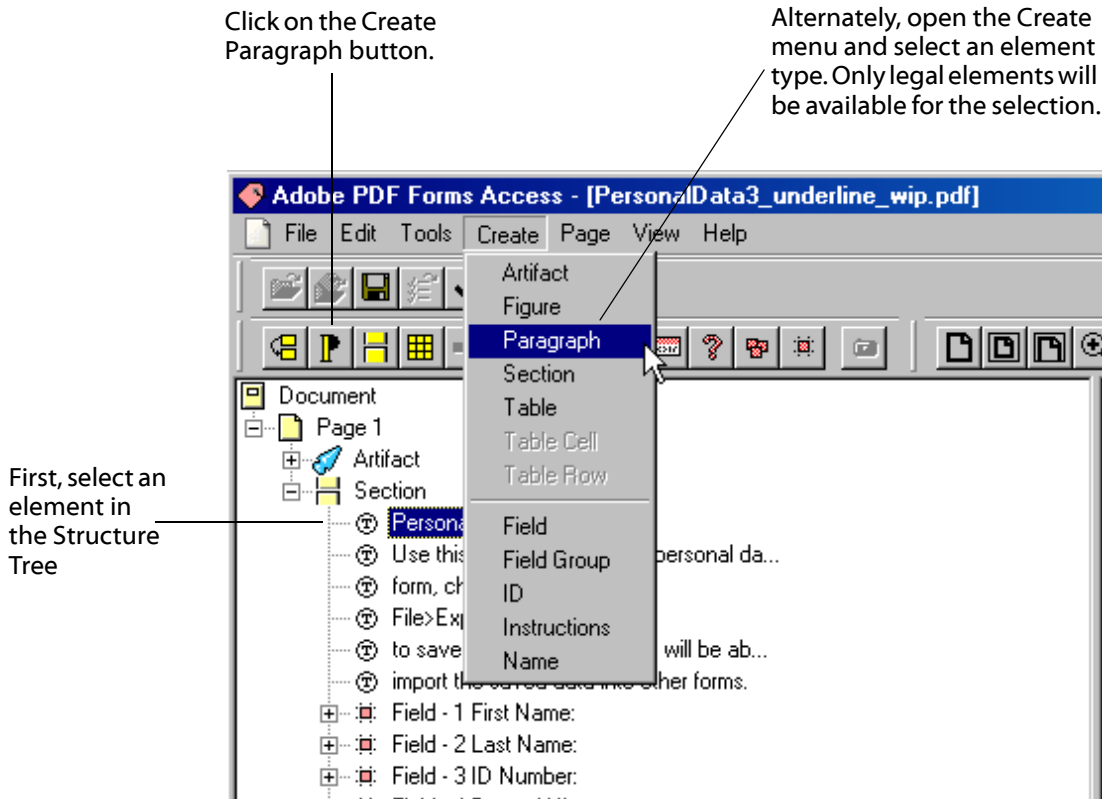
Adding Elements to the Structure Tree


1. Start PDF Forms Access if it is not already running.
2. If **PersonalData_1.pdf** is not still loaded, click on **File > Open** and select it.
3. Fully expand the tags in the Structure Tree.
4. The first task is to wrap the **Text** elements at the top of the Structure Tree with a parent **Paragraph** element.

This is not actually necessary for accessibility purposes: the individual Text elements are in the proper sequence and will therefore be correctly read by screen readers no matter what parent elements they are wrapped in. However, it will provide practice for wrapping Structure Tree elements in parent tags. It will also reduce clutter in the Structure Tree, which is particularly useful when working with complex forms, and will serve to illustrate one of the Forms Access tools, the **Move** Tool.

Click on the first **Text** element, *Personal Data Form*, in the Structure Tree.

5. Select **Create > Paragraph** from the Menu bar, or click on the **Create Paragraph** button  in the Toolbar. This will wrap the first Text item in a Paragraph element.

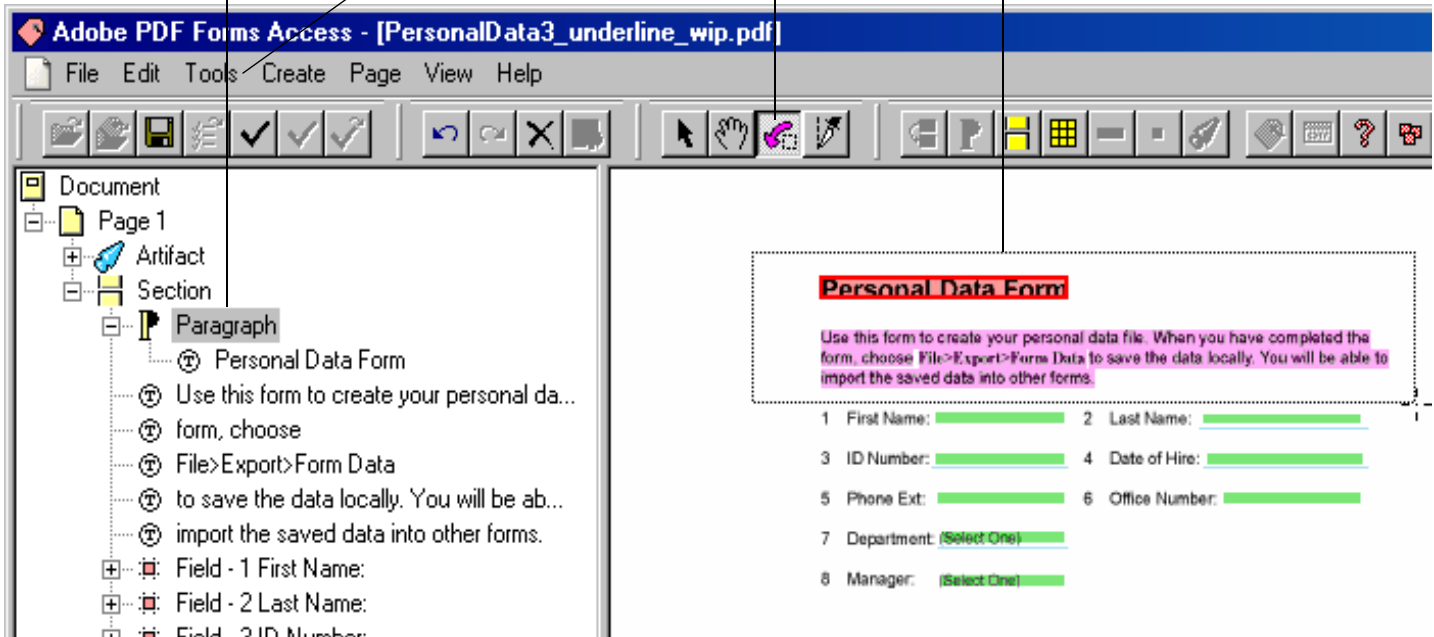


6. Select the new **Paragraph** tag in the Structure Tree.
7. Select **Tools > Move Elements** from the Menu bar, or find the Move  Tool in the Toolbar and click it.
The mouse cursor icon will change to a cross-hair.
8. Refer to the next screen shot. Hold down the **left mouse button** and draw a border around the heading and the block of text below it.
Release the mouse button.

Select the parent element in the Structure Tree.

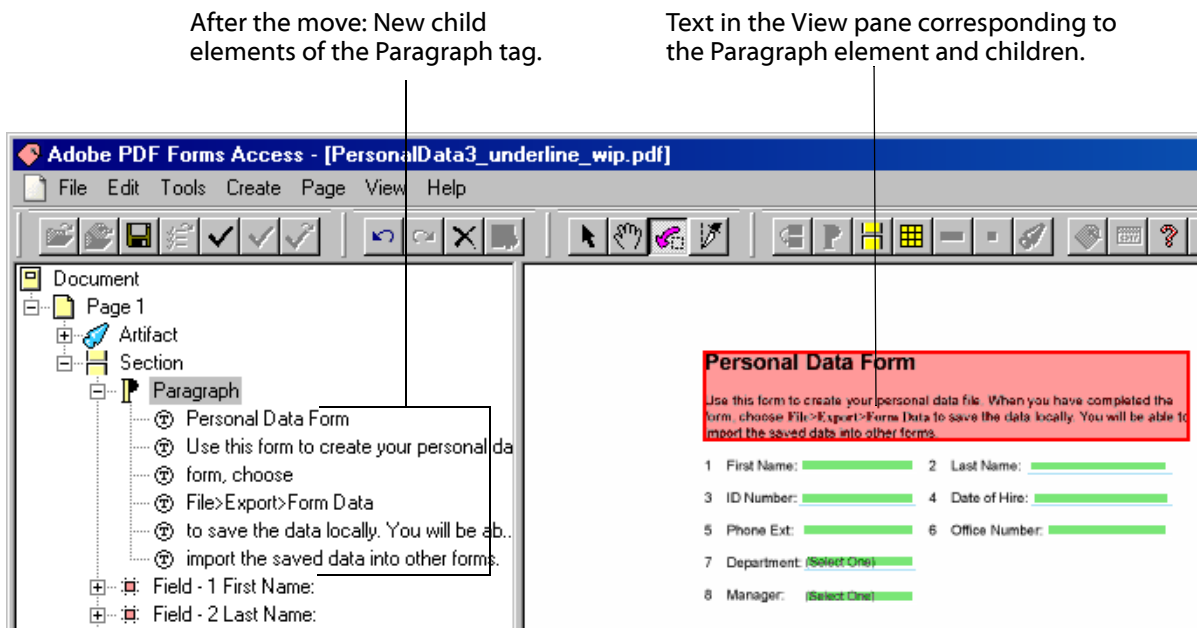
Select **Move Elements** from the **Tools** menu or Click on the **Move Tool**.

Select the document elements to move with the left mouse button, then release.




When you release the mouse button, the Text elements in the Structure Tree that correspond to the selected items in the View pane will be moved under the selected Paragraph element as its children. (The heading was moved under the Paragraph element in Step 5.) See the screen shot below. The Paragraph tag can be folded, if desired, to make the Structure Tree somewhat easier to read.

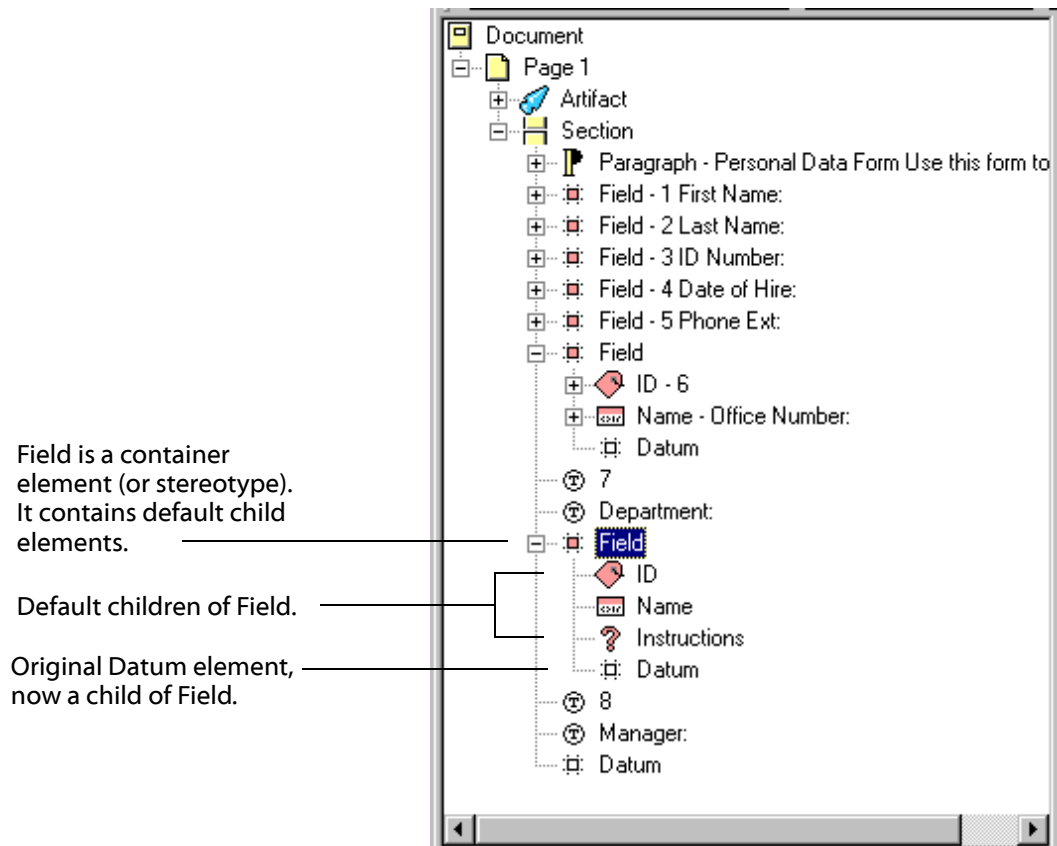
Instructor Notes: Point out that, while wrapping the Text elements for the text block in a Paragraph element will not affect the reading order in any way -- and no additional accessible text has been added -- it is good practice and makes the Structure Tree more readable and easier to comprehend. Another advantage of nested multiple items in parent tags is that they can all be moved, (or wrapped, or deleted) at the same time as a group.




9. Use this procedure to wrap the first Combo Box **Datum** item with a **Field** element.

- In the Structure Tree, **click** on the **Datum** element associated with field number 7, *Department*. to select it.
- Select **Create > Field** from the Menu bar, or click on the **Create Field**  button in the Toolbar. This will wrap the Datum element with a Field element tag.
- A **Field** element will be inserted into the Structure Tree as the parent tag of the selected Datum element. The Field element brings three children with it -- **ID**, **Name**, and **Instruction** -- which now become siblings of the Datum element you wrapped.

[Instructor Notes: Students may wonder what the Instructions element is used for. It will be covered in the next exercise.](#)



10. **Repeat Step 9** for the **Datum** element associated with field number 8, *Manager*, to create a parent Field tag for it.
11. You may have noticed that the Text  elements associated with the Datum elements you just wrapped, for example, 7, and *Department*, are not nested as children under the ID and Name elements, as they are in the Field tags that were automatically generated during initialization. Refer to the screen shot above. You can fix this by simply moving the Text elements under the ID and Name tags.

Click on the **Text** element for the number 7.

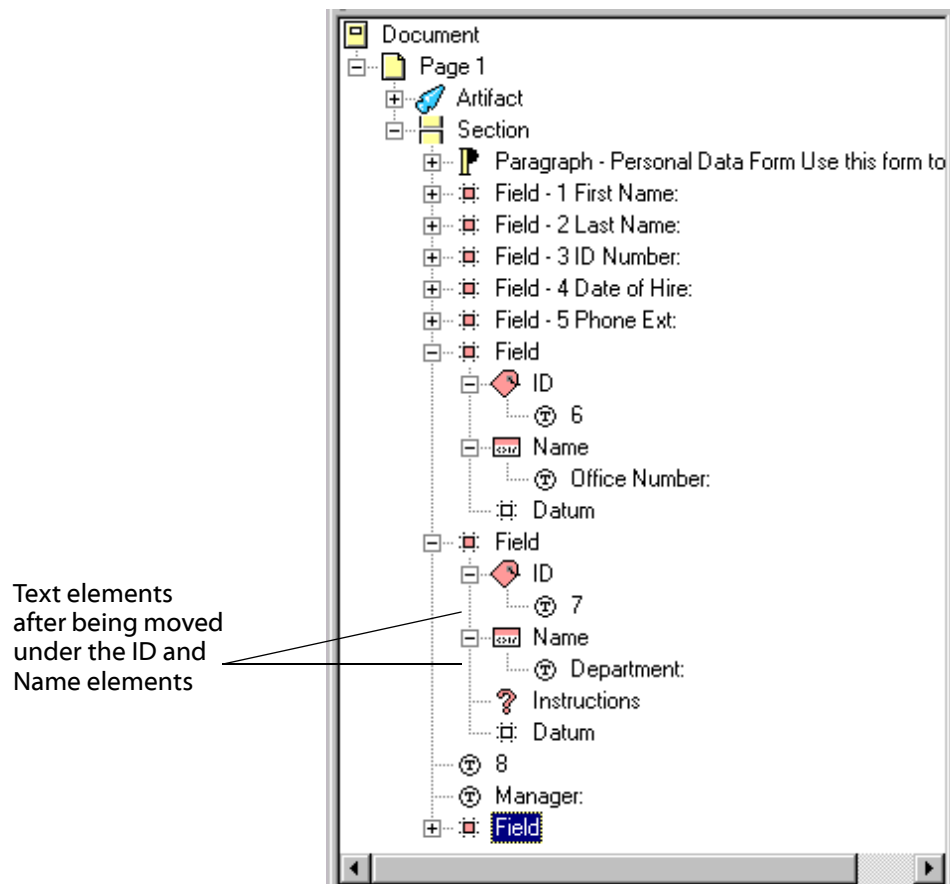
With the mouse cursor on the selected element, hold down the **left mouse button** and **drag** the Text element on top of the **ID** child element and **release** the mouse button.

[Instructor Notes: If you are moving an element to a parent element that does not yet have children, hold the element to be moved directly over the parent's icon in the Structure Tree. The parent icon will become highlighted when the moved element is in place. Release the mouse: the moved element will become a child of the parent element.](#)

Note: you will see a horizontal bar as you move the element over areas where it cannot be legally placed. When you have it correctly positioned over the ID element, the ID element will be highlighted.

Step 12 results in the Text element 7 being placed as a child of the ID element associated with the Department field in the document.

12. Select and drag the **Text** element for *Department* to the **Name** element and drop it there. The Department text is now a child of the Name element.



13. You saw earlier that PDF Forms Access automatically creates a Short description for Datum fields that it wraps in Field tags during initialization. If you manually wrap a Datum element in a Field tag, and then move Text elements into its child ID and Name tags, like you just did, PDF Forms Access will also compose Short description options for the sibling Datum element comprised of the ID and Name element text.

Click on the **Datum** element.

In the **Short description** attribute drop down box, you will see the options: 7, 7 *Department*, and *Department*. Select one.

14. Leave the last Field element, for *Manager*, as is for now. In the next exercise, you will use an alternate method to add a number ID and Name label to the Field.
15. Select **File > Save As** and save your work as **PersonalData_2.pdf**.



Note: When you use the **Save** command, PDF Forms Access does *not* save the originally loaded file. It will save a *copy* of the file, with your changes, named *(your filename)-new.pdf*. The original file remains unchanged. Later, when you **Finish** the form, PDF Forms Access also writes out the finished file with the name *(your filename)-new.pdf*. If you use the default *-new* filename, for Saving your work in progress, you run the risk of accidentally overwriting one of your Saved versions when you Finish the form. Unlike a Saved file, a Finished file cannot be successfully re-loaded into PDF Forms Access, so you may have lost any changes you made to the Structure Tree. It is safer to use the **Save As** command with a unique filename to save your work for reloading later. Then, when closing a file, or when closing PDF Forms Access, answer **No** to the **Save Changes** dialog unless you also mean to save a copy of the file as *(your filename)-new*.

[Instructor Notes: After using Save As with a different file name for the first time, use Save As with the same file name subsequently, and specify that the existing file be replaced.](#)



Adding Accessible Text to Form Fields

[Instructor Notes: The following discussion and example of creating accessible text for fields are meant to be suggestions rather than fixed guidelines. The optimal design will depend on the design of the form document to some extent, as well as on individual preferences.](#)

Many of the adjustments you will make to the Structure Tree, particularly for complex forms, will involve straightening out and ordering the labels and other instructions from the original document so that they refer to the correct fields and provide clear directions for use. If you are not visually impaired, it may be useful to imagine yourself sitting in front of a computer that is running Acrobat and has a form document loaded. However, this computer has no monitor: just a screen reader, and the tab and arrow keys with which to navigate through the form. What would you want to know, or need to know, about the document in order to complete the form without being able to see it, and with a minimal amount of confusion and frustration?

One useful piece of information would consist of some kind of sequential identification of the fields in the form, such as *Item 7*, or *Field number one*, or *21-A*, and so on. Fields in visual documents are often identified in this manner: if so, you should make sure those identifications are read by the screen reader; if not, you should consider adding field identifications to the document's Structure Tree, thereby including them in the logical read order.

Another useful piece of information would be the name or label of each field. Visual documents almost always provide these, such as *First Name*, *Last Name*, *Address*, and so on. These labels should be read by a screen reader in such a way that it is clear which label applies to which field.

A third useful piece of information might consist of instructions for entering data in a field. *Enter your full name; do not use initials or nicknames*, and, *If the amount in box 17-A is over \$3000, you must complete items 17-B through 17-E*, would be examples of such instructions.

Finally, it would be useful to have an abbreviated description for each fillable data field in the document, so that users could quickly identify each field as they navigate through the document without having to wait for the screen reader to read through body text or instructions. This short description for each field would be particularly useful for people who are familiar with a given form, for example, office workers who do regular data entry.

Putting this all together, you might design audible text to ease and clarify the task of negotiating through the document without being able to see it, along the lines of the following example (the screen reader supplies the *Edit* text):

Item 1: First name. Use your complete first name; do not use initials. Enter your first name. <Edit>

Most screen readers will jump to the next field in a form when the Tab key is pressed, providing an easy way to get to just the form fields while bypassing the rest of the document. In this case, a lengthy description of each field, like the example above or worse, somewhat defeats the purpose of quickly tabbing

through the fields. Ideally, you would like to provide an abbreviated description of the field, such as:

Enter your first name. <Edit>.

There are a number of ways to provide accessible text for data fields. Most often, the accessible text comes directly from the text in the original document itself, which gets replicated in the Structure Tree as Text elements. It is possible, however, to include accessible text in the Structure Tree that was not in the original document. The elements that are available in the Structure Tree have *attributes* that can be used to provide accessible text. Such text can be included in addition to the existing text in the document. On occasion, Structure Tree element attributes can be used to replace existing text, either verbatim or with variations on the original. You will see examples of this as the course proceeds.

The accessible text options available through elements in the Structure Tree, alone or in combination with existing text in the document, can be used to create text descriptions for fields consisting of combinations of ID, name, instructions, and short description, as in the examples above. At minimum, the text that is spoken by the screen reader when it encounters a data field should replicate the original text label, and also the ID if there is one. However, you do have the capability to augment the description of data fields beyond their original labels, if necessary, to better assist visually impaired users in processing the form. You will do this in the next exercise.

The Field element stereotype, and its child elements -- *ID*, *Name*, *Instructions*, and *Datum* -- each have attributes, some of which can be used to supply alternate or additional accessible text for the field. You can view an element's attributes in the Properties palette by clicking on the element in the Structure Tree. You can add values for attributes or edit them in the Properties palette.

The **Field**, **ID**, **Name**, and **Instructions** elements each have the following attributes which are displayed in the Properties palette:

- **Element type**
The type of the selected element, for example, *Field*. The drop down menu list contains other element types that the selected element can be legally changed to, if desired. This attribute does not contain accessible text and is not read by screen readers.
- **Title**
This attribute does not contain accessible text and will not be read by screen readers. If you give it a descriptive value, however, such as *First Name*, that term will be displayed next to the element's tag in the Structure Tree, making the tree more readable.
- **Actual text**
Any value you give to this attribute will be read by a screen reader in normal reading mode. In general, use Actual text to replicate any actual field-related text in the document that is not accessible, such as bit mapped characters or words.
- **Alt text**

Any value you give to this attribute will also be read by screen readers in normal reading mode. In general, use this attribute to supply additional accessible text for fields, such as an expanded set of instructions, or field IDs that did not exist in the original document.

- **Language**

It is good practice to select the language used by the document, but at the present time, Language is not used by accessibility tools.

The **Datum** child element of the Field element has one editable attribute:

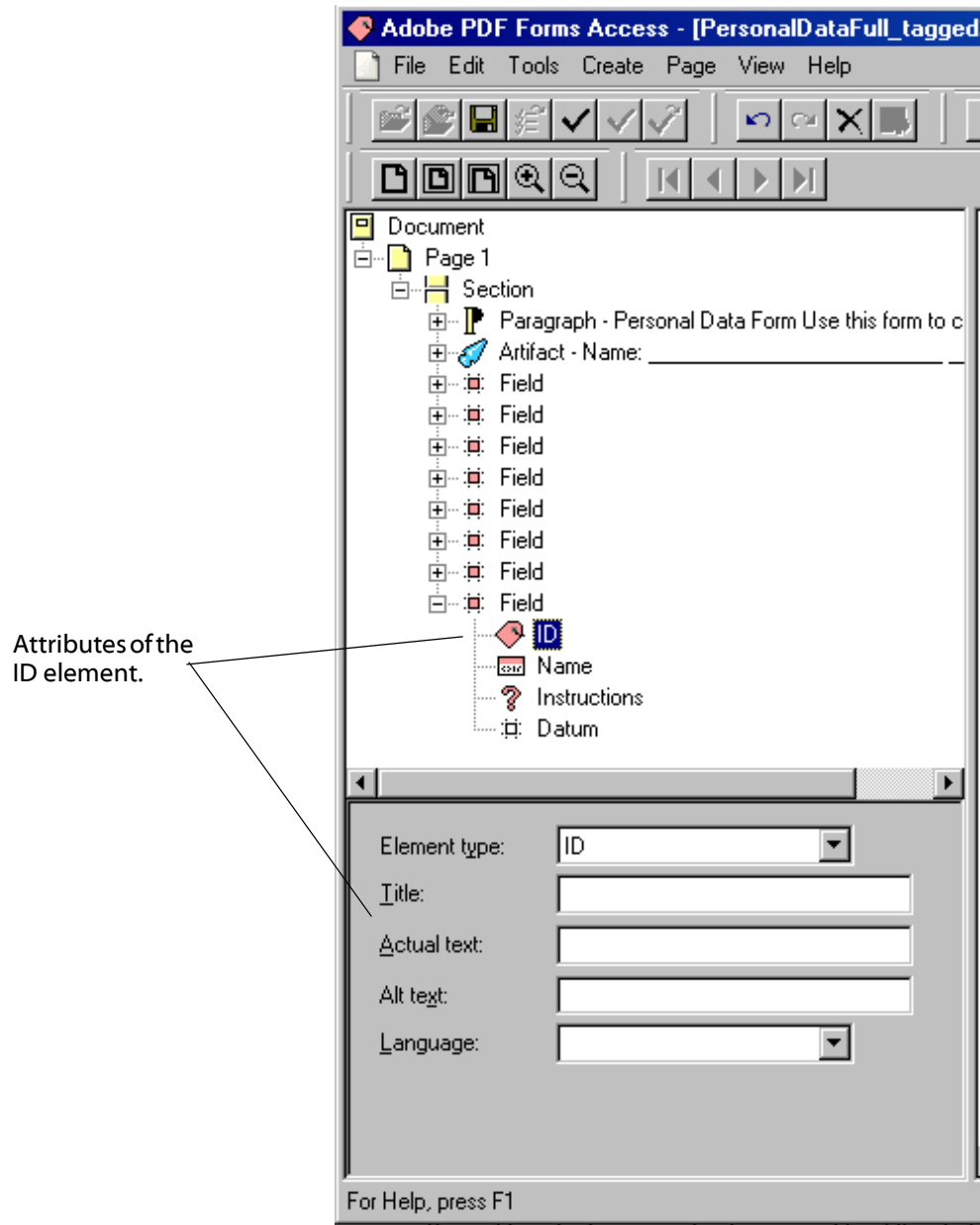
- **Short description**

As the name implies, Short description provides an abbreviated description of a field and what to do with it. PDF Forms Access will create a Short description for a Datum element automatically under the following circumstances:

- The Datum element is a child of a Field element that was either created during initialization or added manually.
- The ID and Name child elements of the Field element (which are siblings of the Datum element) themselves have children Text elements. These Text element children can be created by PDF Forms Access at initialization time, or moved manually from other places in the Structure Tree to be children of the ID and Name elements.

If PDF Forms Access has not created a Short description, you will need to supply one manually. If Forms Access has created a Short description, you can modify it if you wish. Although its composition is somewhat arbitrary, a Short description for the field *First Name* might be something like, *Enter your first name*, or simply, *1: First Name*.

The Short description is always read by screen readers in both normal reading mode and when tabbing between data fields. If the Short description is missing the screen reader will report an error on the order of *MSAA data not available ...*



When a screen reader in normal reading mode encounters a Field element and its children in the tag structure of a form document, it will read any text supplied as values for the *Actual text* or *Alt text* attributes of the Field or its child elements, and it will read any text supplied for the *Short description* attribute of the Datum element. When the user is tabbing sequentially through the fields in a form, the screen reader will read only the *Short description* text of the Datum element as the cursor jumps to each field.

The sequence of what is read by the screen reader when it encounters a field is determined by the sequence of the elements that are children of the Field element, because the screen reader reads sequentially through the Structure Tree

top to bottom. The Field element and its children provide a template of sorts for providing accessible information about a field in a logical manner. Earlier, you considered some of the information that might facilitate the process of negotiating a form without the benefit of a visual representation of it. A suggestion was made that something along these lines might be useful:

Item 1: First name. Use your complete first name; do not use initials. Enter your first name. (Edit).

The sequence of child elements under the Field tag provides the mechanism to construct such a message. The pattern is, in effect:

ID, Name, Instructions, Datum (Short description)

which are provided in sequence by one of the accessible text attributes of each of the Field element's children. In this manner, the Field element template also provides a convenient way to use a standard pattern for accessible text across fields.

For the *ID*, *Name*, and *Instructions* elements, accessible text can come from two sources:

- Moving Text elements from the original Structure Tree to be children under the ID, Name, and Instructions elements.
- Supplying values for the *Actual text* or *Alt text* attributes of the ID, Name, and Instructions elements.

For the *Datum* element, an accessible text value is supplied for the only editable attribute, *Short description*.

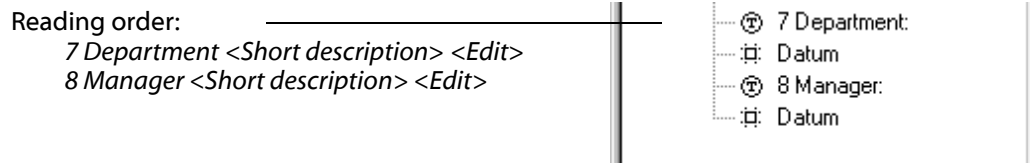


Whether or not you supply values for the Actual text or Alt text attributes of the ID, Name, and Instruction elements is somewhat optional, and somewhat dependent on the complexity of the document. *Short description*, on the other hand, should always be given a text value. A missing Short description will sometimes not make much difference in normal read mode, but when the user tabs to a field without a Short description, the screen reader will say something to the effect of *MSAA data is not available*. In addition, if the Acrobat Accessibility Checker is run on a document processed by PDF Forms Access, it will flag as errors any missing Short descriptions for fields.

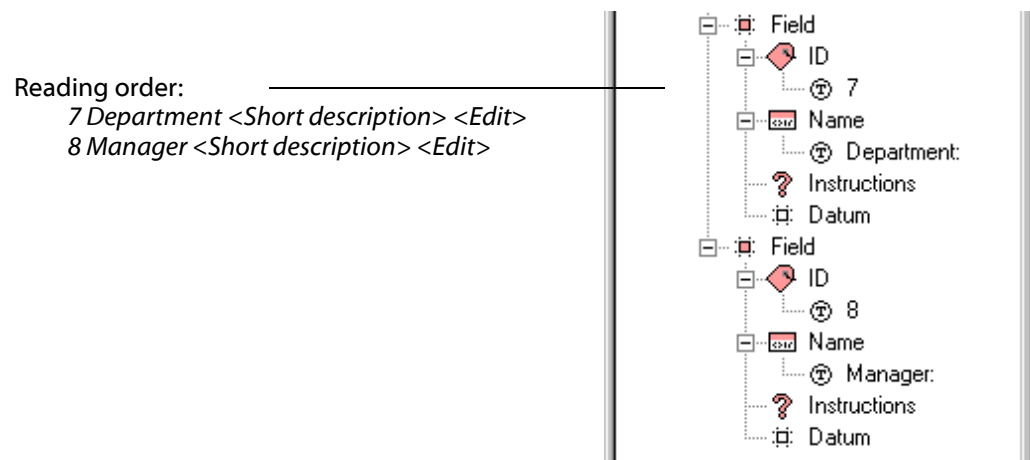
Accessible Text for Data Fields Summary

Keep in mind that the initial Structure Tree created by PDF Access may not need significant modification. The primary issue to consider is the read order of the form as determined by the elements in the Structure Tree. If the initial Structure Tree represents the elements from the document in their correct reading order, no matter how the tags in the tree are structured, then the document is properly accessible. There may be reasons, however, to structure the tags in the Structure Tree more formally than in the initial configuration even if the initial reading order is correct.

The following two example sections from two different Structure Trees would be read in exactly the same way by a screen reader:



The *Edit* instruction is supplied by the screen reader for each data field.



The only difference between the two samples is that the bottom one, with the Datum element as a child of a Field element, will have an automatically generated Short description, whereas you would have to supply one manually for the example at the top. If the Short descriptions were the same, however, the two samples would be read in the same way by a screen reader.

The advantages of more formally organizing the Structure Tree include the following:

- The Structure Tree is less cluttered and easier to read because the parent tags can be collapsed. This is particularly helpful for complex forms.
- The use of parent Field containers helps standardize the way IDs and field labels are used throughout the document.
- The use of parent Field elements provides separate, distinguishable containers for the different elements that support a data field: its ID and Name label, and, if desired, a special set of instructions, making it easier to identify the individual parts associated with a field.
- Datum elements that are wrapped in Field elements will have Short descriptions automatically generated from the text value contained in the ID and Name child elements of the Field.
- Related groups of elements that are wrapped in a parent element can be moved, deleted, or wrapped all at once using the parent tag.



Exercise: Adding Accessible Text to Form Fields

In order to do any of the exercises in this module, you must have Acrobat 5 or higher and PDF Forms Access 1.0 or higher installed on your machine, and you must have copied the sample files in the archive **FormsAccessExamples.zip** to a folder on your machine.

In this exercise, you will explore an alternate method for providing accessible text for a data field using Structure Tree element attributes. It should be emphasized that this is an alternate or supplemental method, but not necessarily a recommended one, depending on the nature of the form. Some forms are more suited to one method or the other, and you will see some examples of this later on. You will also make use of the Instructions element for one of the data fields, and learn how to Finish the processing of a form.

Using Element Attributes for Accessible Text

You will begin by working on the **Field** element associated with *Manager*. As you recall, and can see in the Structure Tree, you did not move the Text elements **8** and *Manager* to be children of the ID and Name elements as you did for the *Department* Field. For instructional purposes, you will take a different approach this time, reproducing that text in the *Actual text* attributes of the ID and Name elements, and then converting the original text to Artifacts.

1. **Expand** the **Field** tag for the *Manager*, if it is not already expanded, until you see its child elements **ID**, **Name**, **Instructions**, and **Datum**.
2. **Click** on the **ID** tag under the *Manager* Field element.

Notice the attributes of the ID element as listed in the Properties palette: *Element type*, *Title*, *Actual text*, *Alt text*, and *Language*. The Element type is *ID*, logically enough, and the other attributes have no values as yet.

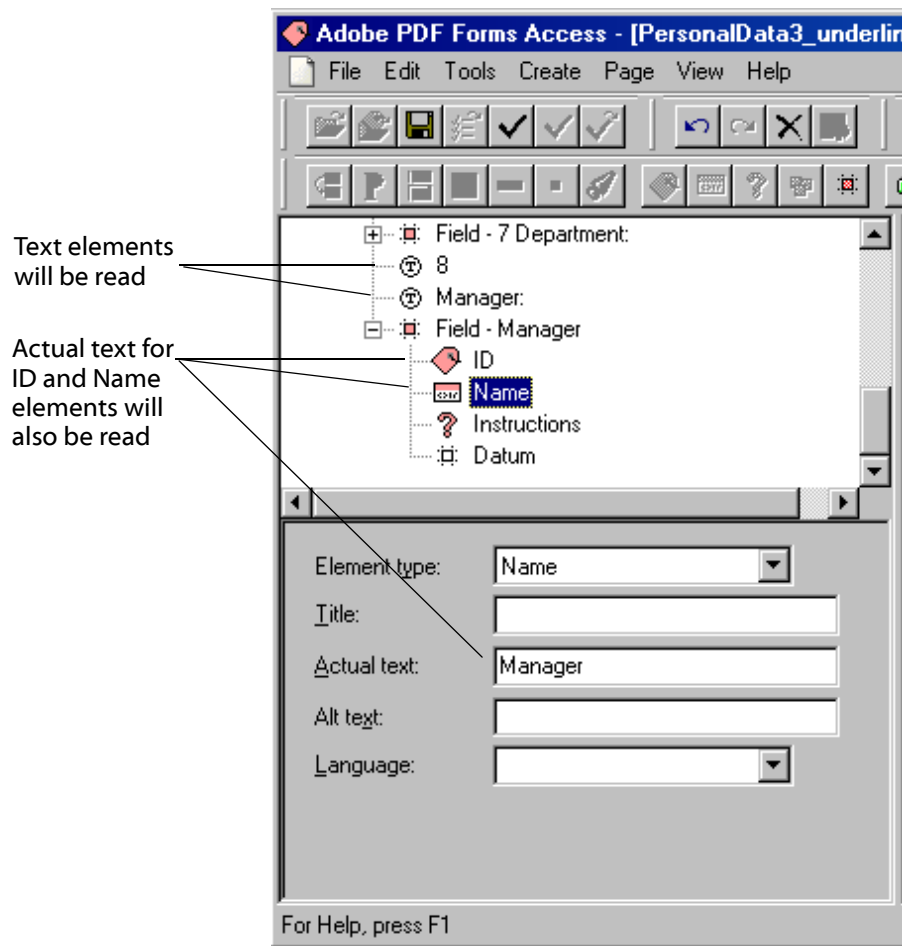
3. You will add text to the *Actual text* attribute of the ID element that will identify the positional sequence of this field in the form. Naturally, it makes sense to follow the format of the other field IDs in the form and replicate the existing ID, which is the numeral **8**.

Type **8** into the **Actual text** attribute field.

4. **Click** on the **Name** element. The Name element has the same attributes in its Properties palette as does the ID element.
5. Type the actual text value of the field label, *Manager*, into the **Actual text** attribute field of the **Name** element.

The current situation with the Manager Field, as reflected in the Structure Tree screen shot below, is that the Actual text values for the ID and Name elements will be read, in that order. However, the original Text elements, **8** and *Manager*, are also still in the Structure Tree and will also be read by screen readers, in that order. You need to get rid of them somehow to avoid the repetition. The

way to remove elements from the Structure Tree and therefore from the read order, is to wrap them in Artifact elements.



6. **Click** on the **Text** element for the number **8**.

Select **Create > Artifact** from the Menu bar or click **Create Artifact** in the Toolbar.

This will wrap the Text element in an Artifact tag. The text will not be included in the logical read order, and the Artifact tag will not even appear in the Acrobat PDF Tag structure after the form has been fully processed in PDF Forms Access.

7. An alternate way to do this would be to simply move the Text element into the existing Artifact tag at the top of the Structure.

To do this, **click** on the **Text** element for *Manager* and hold down the left mouse button to drag it on top of the Artifact tag.

8. Because the accessible text for ID and Name has been created through element attributes, rather than by placing Text elements as children under ID and Name tags, you have some unfinished business. The **Datum** element **Short description** has not been generated automatically, so you will have to add it manually.

Click on the **Datum** element.

In the **Short description** field, enter a description that matches the existing format, which is `<number> <label>`. Type:

8 Manager:

As mentioned, the preceding steps describe an alternate, but not necessarily recommended, method of supplying accessible text for fields. When or why would you want to use it? One example might be adding accessible text for field IDs that were not present in the original document. Another use might be in expanding label abbreviations, like the one for field number 5, *Phone Ext*. You might want to give it an Actual text or Alt text value of *Phone Extension*, and then wrap the original *Phone Ext* label in an Artifact.


Using the Instructions Element

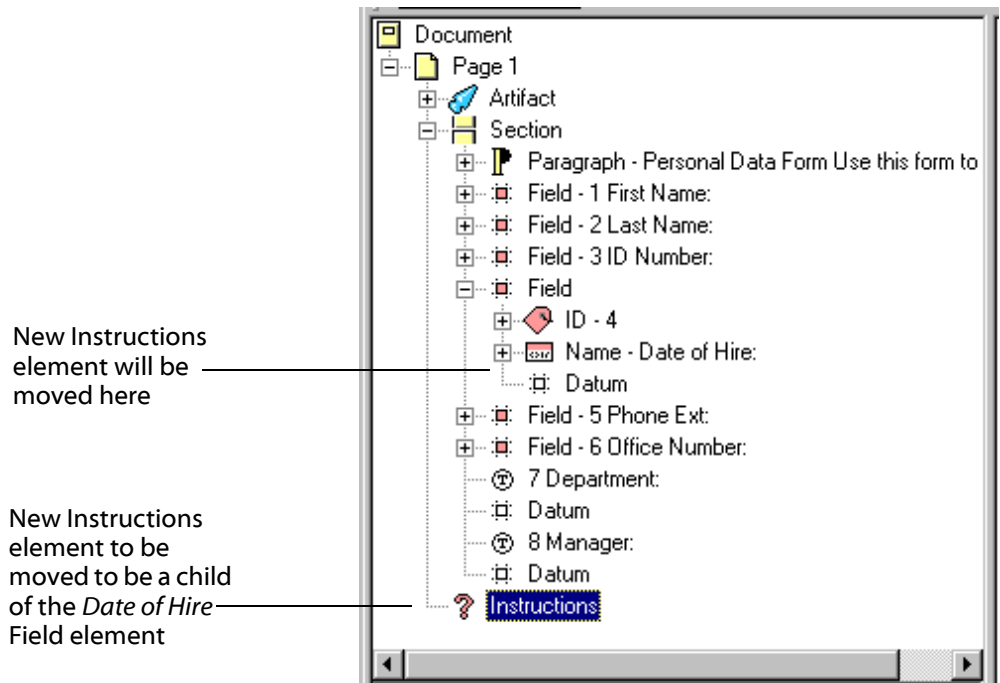
You may have been wondering about the *Instructions* child element that was created when you manually added a Field element to the Structure Tree. The Instructions element provides a container for any special instructions that may be associated with a Field. Think of the IRS 1040 form as an example: there are instructions at various places on the form, running down the side margins, and at various orientations to the fields to which they refer. The Instructions element would allow these text blocks to be gathered and placed in the Structure Tree in specific relation to their fields. Or, the Instructions element could be used to supply additional accessible instructions useful for visually disabled users.

Accessible text is added to the Instructions element in much the same way you added text to the ID and Name elements, except you will use the *Alt text* attribute rather than the Actual text attribute since you will be adding text to the logical read order that does not actually exist in the document. In reality, you could use either Actual text or Alt text. You will add accessible instructions text to the *Date Of Hire* field, explaining the format to use for the date.

1. **Expand** the **Field** tag for the *Date of Hire* field.

You need to add an *Instructions* element to the Structure Tree as a child of the Field element. However, if you click on the Field element or its ID and Name children and then open the Create menu, you will discover that you cannot add an Instruction element to the Date of Hire Field.

2. **Click** on the **Page 1** element at the top of the Structure Tree.
3. Select **Create > Instructions**, or click on the **Create Instructions**  button on the menu bar. This will place a new Instructions element at the end of the Structure Tree.



- Now, **drag** the new **Instructions** element into place as a child element of the *Date of Hire* field. Place it below the Name element and above the Datum element.
- Type an instruction into the **Alt text** attribute field of the **Instructions** element. It might read as follows:

Use the form month / day / year.

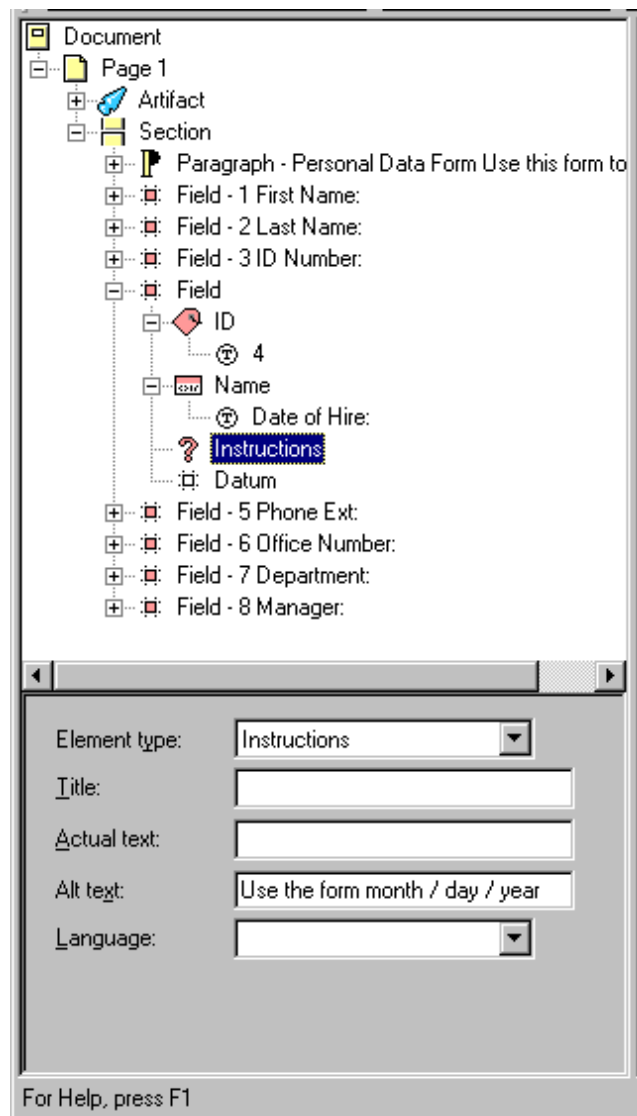
In normal reading mode, a screen reader will read the following sequence when it encounters the Date of Hire field:

*4 Date of Hire <colon> ... Use the form month <slash> day <slash> year ...
4 Date of Hire <Edit>*

If you wished to do so, you could edit the Short description to remove the redundant number 4, and perhaps say something like:

Enter the date on which you were hired.

The screen shot below shows the completed operation:



Short Description Summary

Here are some things to keep in mind about the Short description:

- Because the Datum element is last in the sequence of the Field's child elements, its accessible text will be read last when the screen reader encounters the field, after any existing accessible text for ID, Name, and Instructions.
- When the user is tabbing from field to field, *only* the Short description will be read.
- When you plan accessible text sequences for fields, keep the above points in mind. The accessible text you add to attributes for each field should make logical and grammatical sense when strung together by a screen

reader. The Short description should make sense on its own, but also when read as the last element in a sequence of accessible text for ID, Name, and Instructions. If accessible text exists for the Instructions element, for example, it should not overlap with the Short description text because the screen reader will read both in normal reading mode.

- You can get by with providing text for just the Datum element Short description and ignore all the other child elements of Field. The Short description will be read in all screen reader modes. This is probably more appropriate for very simple forms. You will have to balance brevity with detail if you use only the Short description for accessible text for forms.
- If you provided a Short Description for a field when you originally created a form in Acrobat, that Short Description will be carried over to the Datum element's Short description attribute. You can edit it, or delete it, in PDF Forms Access if desired.

Finishing the Form

1. Click on **File > Save As**, and select `PersonalData_2.pdf` from the browser list. Select **Yes** at the dialog asking whether you want to replace the existing file.

As you have seen, you can use the PDF Forms Access Save As command at various points to save the modifications you have made to the Structure Tree. When you are satisfied with the modifications, you must use the **Finish** command to complete the processing of the file into an accessible PDF form



Note: The Finished PDF Form *cannot* be loaded back into PDF Forms Access. Therefore, it is advisable that you use Save As to save your final Tree Structure configuration just prior to issuing the Finish command. The Finished PDF Form will actually load back into PDF Forms Access, but it will lose significant information in the process.

2. Select **File > Finish** to complete PDF Forms Access processing of the form.
The form file will be closed when processing is finished.
3. You can now **Exit** PDF Forms Access.

The next steps in the development process are to open the Finished file in Acrobat and test its accessibility, which will be covered in the next Section.



Acrobat Capture® Note: When you run Adobe PDF Forms Access with Acrobat Capture, you can save forms as temporary files that remain in the *Form Tag* state until you commit the temporary files to the next step in the workflow. Committing a PDF form marks the form as Finished and moves it to the next step in the workflow, which is typically a *Store Step*. The file uses typical Acrobat Capture workflow naming conventions, rather than appending *-new* to the filename.

Testing Accessible Forms in Adobe Acrobat



Introduction to Accessibility Testing

In this module, you will use Adobe Acrobat to open the form you tagged in the previous exercise. You will examine the resulting tag structure and make necessary modifications to it, and you will be introduced to methods for testing accessible forms, including the use of screen readers.

Module Contents

Topics	Exercises
Opening a Tagged Form File in Adobe Acrobat	Examining the PDF Tags Palette
Accessibility Testing	Testing Form Accessibility

File to Download for Exercises in this Section

FormsAccessExamples.zip, which is normally found in the Exercises folder, contains the following files:

- `PersonalData.pdf`
 - `PersonalData_underline.pdf`
 - `PersonalData_unfinished.pdf`
 - `ProblemForm1.pdf`
 - `ProblemForm2.pdf`
 - `ProblemForm2mod.pdf`
 - `ProblemForm4.pdf`
 - `Table1.pdf`
 - `Table2.pdf`
 - `Table3.pdf`
 - `Table1_ext.pdf`
 - `Table1_org.pdf`
 - `Table1_org-new.pdf`
 - `Jaws1.wav`
 - `Jaws2.wav`
 - `form_1040.pdf`
 - `f1040_accessible.pdf`
- You should extract these files from the zip archive if they have not been extracted already.



Opening a Tagged Form File in Adobe Acrobat

When you used the Finish command to complete processing of the sample form file, the Structure Tree tag hierarchy was converted to another tag structure recognized by Adobe Acrobat, sometimes referred to as the PDF Tag structure. The PDF Tag structure can be viewed in Acrobat by clicking on the Tags tab in the palette pane at the left of the workspace window. The elements, or Tags, in the PDF Tag structure determine what is actually read by screen readers. The sequence of the Tags in the PDF Tag hierarchy reflects the sequence of the tags you set up in the Structure Tree in PDF Forms Access, and the sequence of the PDF Tags in the tag hierarchy controls the read order of the document.

Instructor Notes: The use of the term *tags* in both Acrobat and Forms Access (Adobe InDesign uses it too) can be confusing. They all mean and do basically the same thing -- they provide a separate structure from the visual document whose elements identify the various parts of the visual document. The elements can be arranged independently from the visual document, and they can contain additional information related to the document parts, such as what kind of part it is and whether it has any attached accessible text. Although Acrobat tags and Forms Access tags function similarly, they use completely different mechanisms. The end product -- what is read by a screen reader -- is always an Acrobat Tagged PDF document. Forms Access tags are translated into Acrobat PDF tags with the Finish command. Because the tag mechanisms are not the same, the translated structure will not always be identical to the original, although its resulting logical read order should be. Certain attributes of Forms Access tags, notably Short description, do not get translated to the PDF Tags Palette and will not be visible there even though they are in the logical read order and will be read by screen readers.

When you open the Finished PDF form in Acrobat, many, but not all, of the same elements from the Forms Access Structure Tree will be found in the PDF Tag hierarchy in the Acrobat Tags Palette. Some elements, like Artifacts, are not transferred to the PDF Tag structure. Some elements are given other names in the PDF Tag structure: *Datum* becomes *Form*, for example. The Short description from the Datum element will *not* be found in the PDF Tag hierarchy, but it is in the logical read order and will be read by screen readers if declared correctly in PDF Forms Access. Most of the tag labels from the Structure Tree will be transferred verbatim to the PDF Tags palette, however.

The PDF tag hierarchy starts with a Root node and expands into intermediate parent and child nodes in the same way the Structure Tree does in PDF Forms Access. At the lowest level or end node of each structure tag is the actual content of the document itself, which could be text, a graphic reference, a cross reference, or other content element in the document.

Acrobat provides various tools for modifying the PDF Tag structure, and tag elements themselves or their placement in the PDF Tag hierarchy can be modified in much the same way you can alter the Structure Tree in PDF Forms Access. This section will introduce the PDF Tag structure and how it relates to accessible text in a document, but it will not cover modifying the PDF Tag structure in Acrobat, primarily because that subject has been covered elsewhere (see, for

example, *Authoring for Accessibility*, at <http://partners.adobe.com/access/acroaccess.html>), but also because it is always preferable, if possible, to control the PDF Tag structure from the original authoring tool rather than having to make post processing modifications to the PDF Tag structure once it has been generated. Modifications to the PDF Tag structure made in Acrobat cannot be saved and reapplied: in the event the document is updated, they must be done over again manually. If the tag structure is adequately controlled from the original authoring tool, however, which in our case is PDF Forms Access, then the required PDF Tag structure will be regenerated along with the updated document.



Exercise: Examining the PDF Tags Palette

In order to do any of the exercises in this module, you must have Acrobat 5 or higher and PDF Forms Access 1.0 or higher installed on your machine, and you must have copied the sample files in the archive **FormsAccessExamples.zip** to a folder on your machine.

In this exercise, you will open the Finished form in Acrobat and examine the resulting PDF Tag structure.

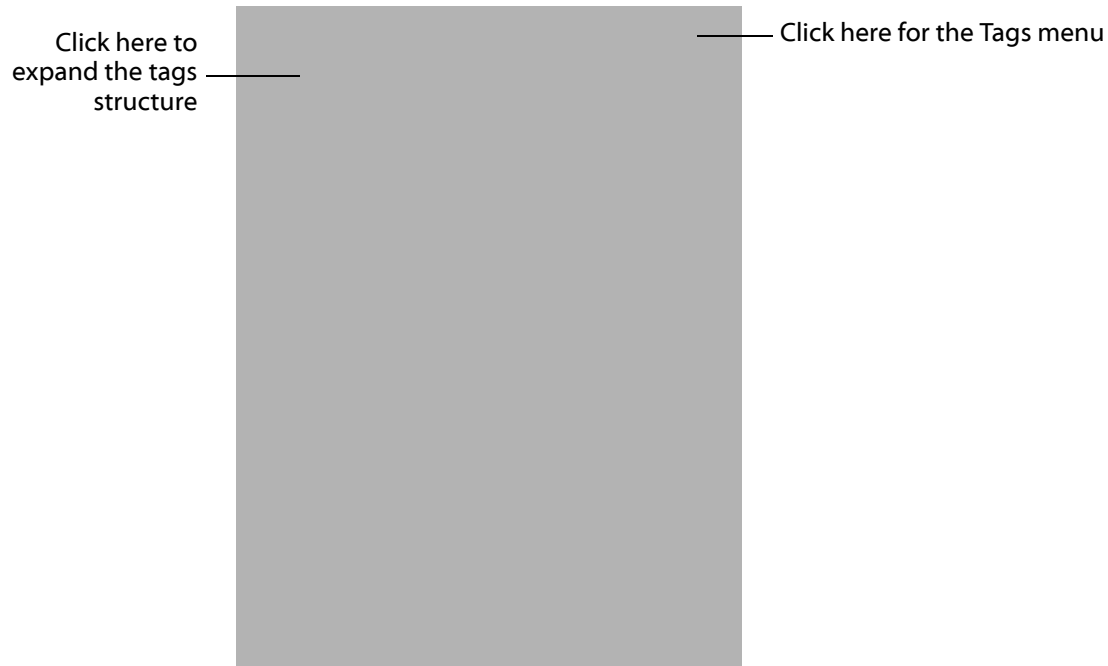
Viewing the PDF Tags Palette

1. Open the PDF file, **PersonalData_2.pdf**, that you generated in the last exercise using one of these methods:
 - Use **Start > Programs > Adobe Acrobat 5.x** to start Acrobat, then use **File > Open** from Acrobat and select **PersonalData_2.pdf**.
 - Navigate to the folder containing **PersonalData_2.pdf**, then double-click the filename to start Acrobat.
2. In Acrobat, select **Window > Tags** to open the Acrobat Tags Palette.



If this is the first time you have opened the Tags Palette, it may be floating detached from the main Acrobat window. If so, put the mouse cursor on the Tags tab on the Tags Palette, hold down the left mouse button, and move it into an empty space at the far left of the palette area in the main window in order to anchor the Tags tab there. You can then click the Tags tab to view the Tags Palette.

When you first open the Tags Palette for a tagged PDF document, you will see only the Tags Root node of the collapsed structure tree, as shown in the following figure.



3. Click on the **+** sign to expand all the elements in the tree.

As you expand the PDF Tag hierarchy, you will see that the sequential order of the tag structure, for all practical purposes, corresponds to that of the Structure Tree in the document you finished with PDF Forms Access.

The Paragraph tag, **<P>**, contains the heading and text from the original document.

The **Field** tags contains the ID, Name, possibly Instructor, and Datum (now named Form) child elements from the Forms Access Structure Tree.

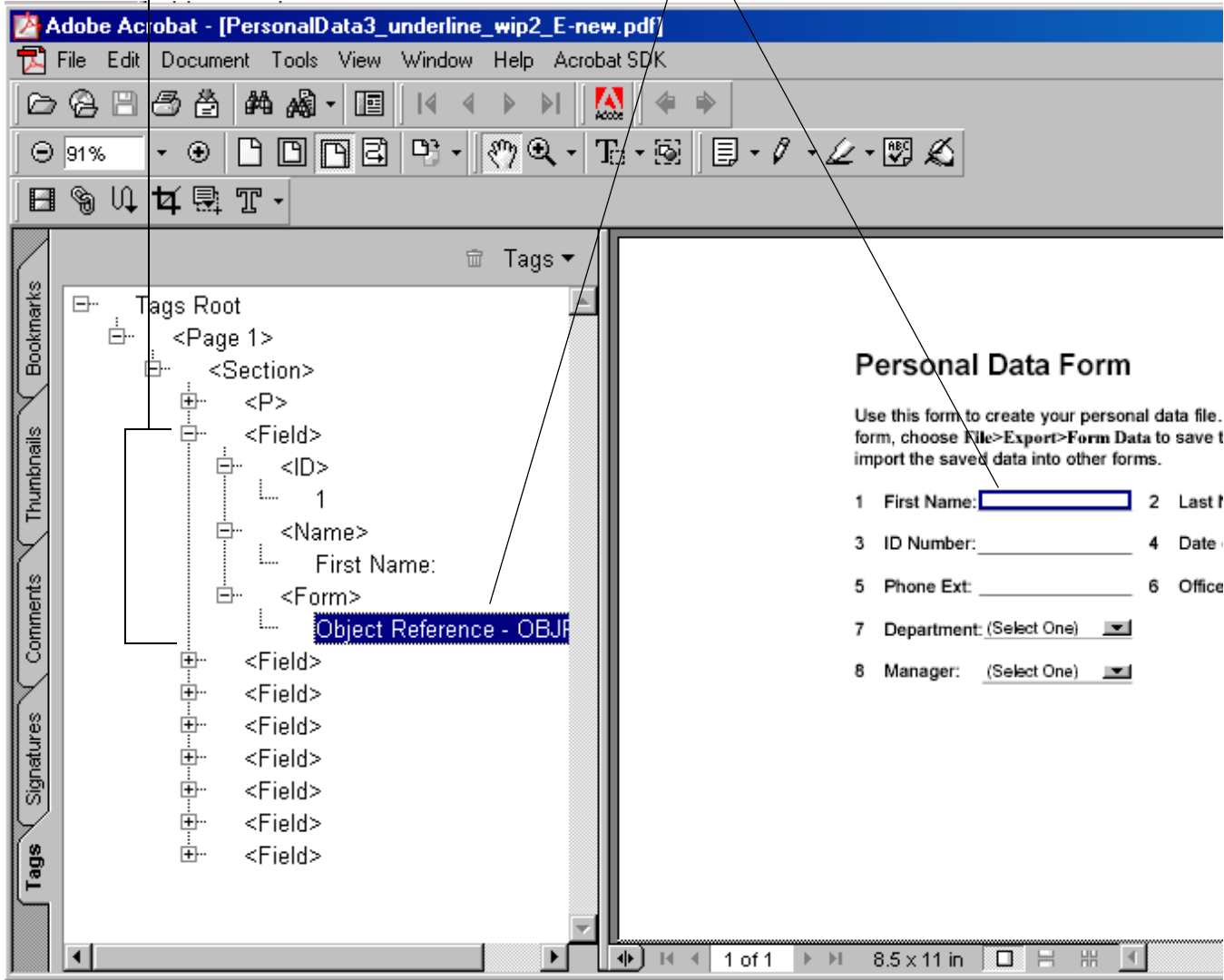
4. Right-click the **Tags Root** element in the Tags Palette and select **Turn On Associated Content Highlighting** from the menu.

This causes associated elements in the document View pane to be highlighted when a corresponding PDF Tag is selected in the Tags palette.

5. Select the **<P>** Paragraph element. The corresponding text area at the top of the document will be highlighted with a surrounding border.
6. Select the **Object Reference** element under the **Form** tag child of the first Field tag. The data entry area of the First Name field will be highlighted.
7. Click on the **ID** and **Name** tags, and note their corresponding elements in the document.
8. Select other Field tags and their children. Verify that the top-down order of tags in the Tags Palette matches the intended logical read order of the form.

Field Tag and children

Using *Turn On Associated Content Highlighting* will highlight associated items in the Tag Structure and the View pane.



9. **Expand** the PDF Tag for the forth field, *Date of Hire*.

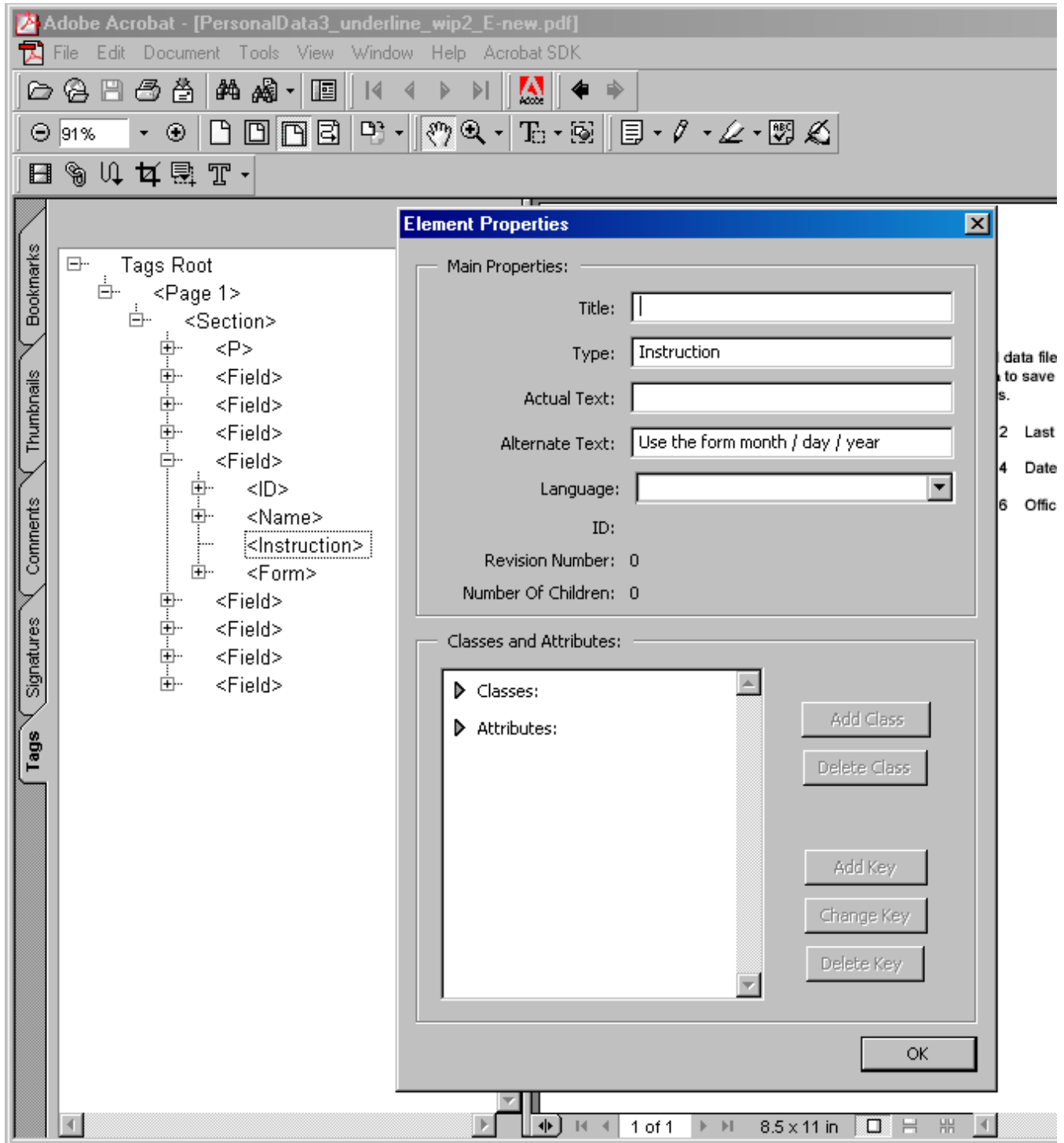
Where is the Alt text you entered for the Instructions element?

Right click on the **Instructions** tag and select **Turn Off Associated Content Highlighting**, then select **Element Properties**.

[Instructor Notes: Turning on Associated Content Highlighting disables all other options in the Tags Palette right-mouse button menu. To access the other options, students must turn Associated Content Highlighting off.](#)

In the Element Properties dialog, you will see an attribute field for **Alternate Text** with the value you supplied in PDF Forms Access.

If any Actual text or Alt text exists in the ID, Name, or Instructions tags, it will be read by screen readers in the sequence in which the tags appear in the structure tree.



10. **Right click** on the **Form** tag and select **Element Properties**.

Except for Type, the attributes have no values. What happened to **Short description**? The short answer is that it is there, in the logical flow, but you cannot see it in the Tags Palette because the Short description is stored in a different format than are PDF Tags. The Short description can be viewed in Acrobat via the Form Tool. You can be confident that, if a Short description was present in the form file before it was Finished in PDF Forms Access, it will be read by screen readers in the same order it appears in the tag hierarchy. The only way you can be absolutely sure, however, is to use a screen reader.



Accessibility Testing

[Instructor Notes: This is an introduction to testing methods. Emphasize that thorough testing requires the use of multiple testing methods. Point out that a screen reader is a useful method for checking logical flow, but that it will not provide information about untagged elements in the document. The Accessibility Checker will do the latter, but is not useful for testing logical flow.](#)

There are three primary methods, each with strengths and shortcomings, for verifying and testing the accessibility of PFD Forms. The best approach when testing for accessibility compliance is to use more than one method.

Stepping through the Tags Palette

In the previous exercise, [Examining the PDF Tags Palette](#), you used one method of testing logical flow, that of verifying the reading order by clicking down the Acrobat Tags Palette with Associated Content Highlighting turned on, and checking the corresponding areas of the document. This is the easiest and most immediate way to verify the reading order, and you should use it regularly when developing the form and modifying the Structure Tree. This method will not directly test the accessibility of the document, however.

Accessibility Checker

Acrobat provides a tool called the Accessibility Checker, which reports on possible accessibility problems in a document and marks the areas of the document where those problems occur. The Accessibility Checker does not look for problems with the logical read order. It can, however, look for the following types of problems:

- No alternate explanations for graphics

Visuals that convey important information for a document should have an alternate text explanation for visually-impaired people. There are some cases where you may choose not to provide alternate text because the graphic does not contribute any significant content, such as a company logo or border graphics for a page.

- Document components, like text, graphics, and so on, that have no corresponding tag element

There may be occasions when the conversion process is unable to convert text, symbols, or visuals into a corresponding tag element. You should investigate these errors since any content without an associated tag element will be invisible to assistive technologies such as screen readers.

- Form fields that have no descriptions

The Accessibility Checker will report errors for any data field that does not have a Short description. This is the only error related to data fields that the Checker will flag.

- No language specification

In the future, screen readers may be able to change languages from document to document. Specifying the language in which the document is written will provide the necessary information. Currently, assisting technologies do not utilize this feature, but the Accessibility Checker still reports missing language specifications as errors.

- Characters that do not have a corresponding Unicode encoding

Unicode provides a means to avoid problems caused by the existence of conflicting character codes found in the hundreds of separate character coding systems used throughout the world. The Accessibility Checker flags those characters because the text that corresponds to the character cannot be reproduced when the encoding is missing, and a therefore a screen reader cannot read it.

While the Accessibility Checker can be a helpful tool, you need to be the final judge on whether a problem warrants fixing.

Screen Readers

Instructor Notes: The use of a screen reader is almost essential for complete and accurate testing of document accessibility. We strongly recommend installing at least one for use by the class. Ideally, each student workstation would have a screen reader installation along with a headset to reduce noise. Alternately, the instructor can have a single copy of the screen reader installed on a workstation and can demonstrate its use to the entire class at appropriate times. In the latter case, the instructor should call for students' attention toward the end of testing exercises, and use the screen reader to demonstrate the results of document modifications made during the exercise.

Screen readers can be difficult and confusing to use. We recommend that you spend some time practicing with one before delivering the course. Learn the basic navigation commands needed to use a screen reader for the kinds of testing described in these exercises, such as start at the top, start over, read next line, read last line, jump to hyperlinks, and so on. Free demo downloads of popular screen readers, with documentation, are available from the URLs listed below.

Using a screen reader to read a tagged PDF file is the benchmark of accessibility. Obviously, if you do not hear the content that you expect to hear, something is wrong. You will probably want to use the screen reader somewhat selectively since listening to a screen reader process a document can be a time-consuming and tedious task. You will need to refer to the screen reader's documentation to understand how to use it. Expect to invest some practice time learning whichever screen reader you select.

Some major vendors of screen readers, GW Micro, Freedom Scientific, and Dolphin Oceanic among them, offer free downloadable demonstration versions of their products. You can use the demonstration version for 30 minutes, then you need to reboot your machine to use it again. You can try them out at

- GW Micro - <http://www.gwmicro.com/demo/>
- Freedom Scientific - http://www.freedomscientific.com/fs_downloads/jaws.asp
- Dolphin Oceanic - <http://www.dolphinusa.com/download/demos.htm>



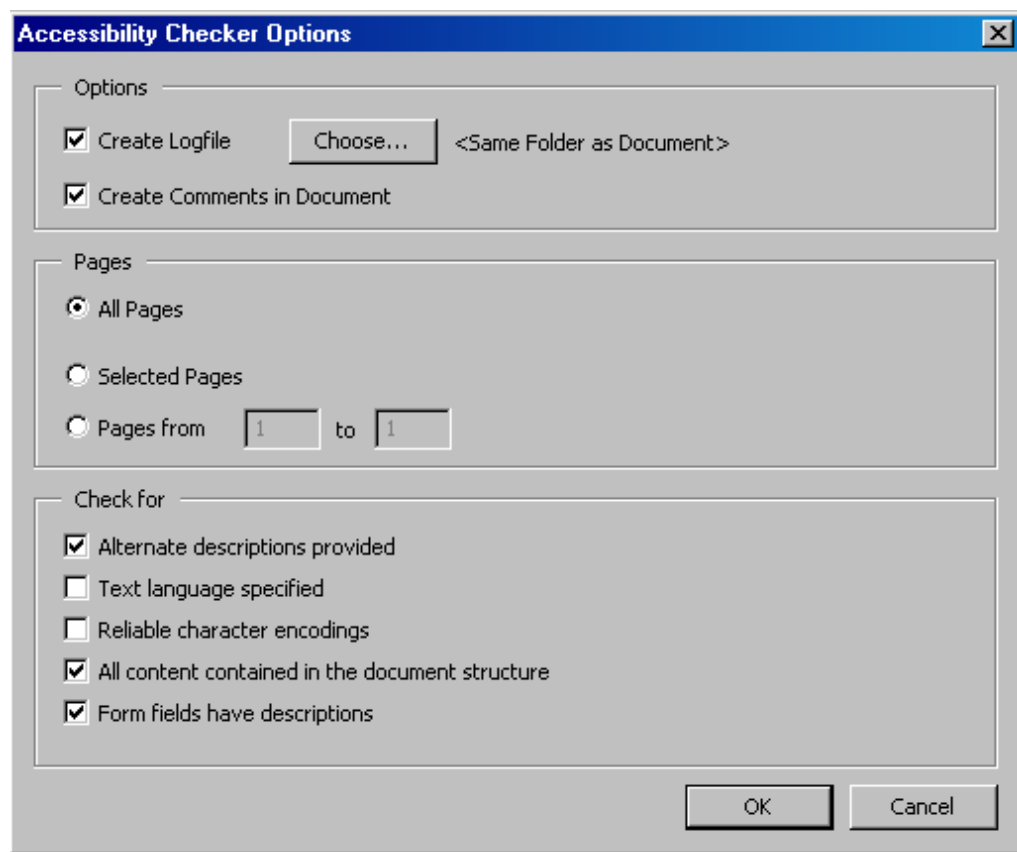
Exercise: Testing Form Accessibility

In order to do any of the exercises in this module, you must have Acrobat 5 or higher and PDF Forms Access 1.0 or higher installed on your machine, and you must have copied the sample files in the archive **FormsAccessExamples.zip** to a folder on your machine.

You have already used the most basic form of testing for accessibility, which is to verify that the top-down order of tags in the PDF Tags Palette matches the intended logical reading order of the form. In this exercise, you will use the Accessibility Checker to verify your form document, and you will listen to how a screen reader reads the form.

Using the Accessibility Checker

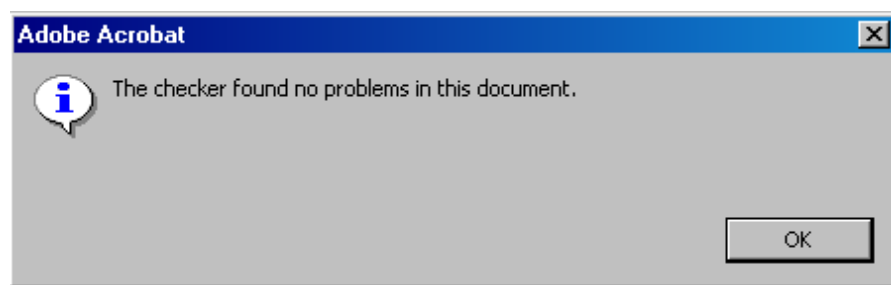
1. In Acrobat, select **Tools > Accessibility Checker**.
2. In the **Accessibility Checker Options** dialog, select the following options:



- Select **Create Logfile** and **Create Comments in Document**. Create Comments in Document is the most useful of the two because Acrobat will mark the exact location of accessibility errors with a comment box.
- Check selected pages or all pages

- Uncheck the *Text language* and *Reliable character encodings* boxes. This is optional: the former does not affect accessibility and the latter is not likely to be a problem with a simple English language document such as the sample form.
 - You do want to check for alternate text for graphics, and that all the content of the document is contained within the PDF Tag structure.
 - You do want to verify that form fields have descriptions. This refers to the *Short description* for the Datum elements. Since you cannot see the Short descriptions in the PDF Tags Palette, this is a very good way to check whether any are missing. Use the Form Tool to verify the Short description.
3. When you have selected the Accessibility Checker options, **click OK**.

You probably will see a not particularly dramatic message like this:



If by chance you neglected to create or select a Short Description for a Datum field, you will see a message indicating the number of errors found, and you will see a Comment placed in the document at the location of the error, much like this:

The screenshot shows the Adobe Acrobat interface with a PDF form titled "Personal Data Form". The form contains seven fields for personal information. A warning dialog box is open over field 7, "Department: (Select One)", indicating that the field lacks a description for accessibility. The dialog box text reads: "X Loretta Guarino Fields need a description so they can be identified during editing by assistive technology like screen readers." The Acrobat toolbar and status bar are also visible.

Personal Data Form

Use this form to create your personal data file. When you have a form, choose **File>Export>Form Data** to save the data locally. You import the saved data into other forms.

1 First Name: _____ 2 Last Name: _____

3 ID Number: _____ 4 Date of Hire: _____

5 Phone Ext: _____ 6 Office Number: _____

7 Department: (Select One) ▾

X Loretta Guarino
Fields need a description so they can be identified during editing by assistive technology like screen readers.

1 of 1 8.5 x 11 in

Screen readers will also complain about form fields that have no Short description. If the Accessibility Checker flags a missing description, you will have to go back to PDF Forms Access and use the following procedure to fix the problem:

- Reload the final version of the form file -- *not* the Finished <filename>-new file that Forms Access created.
- Find the errant Datum element and add or select a Short Description for it.
- Finish the file.

Using a Screen Reader

Screen readers provide a host of commands for navigating documents and for extracting information from and about them. Learning to use a screen reader effectively requires time, practice, and the software manual. Screen reader products will vary in the keystroke shortcuts used to invoke commands.

Teaching you to use a screen reader is well beyond the scope of this course. However, because screen readers are such an essential component of accessibility, and because forms are difficult to fully test without one, some very basic examples of screen reader use will be presented here. How you approach this exercise depends on how your computer, or your classroom, is configured. If you have a screen reader installed on your computer, use it directly (ideally with a

set of headphones). If only the instructor has an installed screen reader, she or he will demonstrate it to you. If a screen reader is not available, there are some .wav files, listed below, containing screen reader recordings. You can play these if you have playback software and a sound card.

Instructor Notes: Depending on the situation and configuration of the room, you will have to exercise some organizational control over the screen reader exercise or end up with what sounds like a room of quacking ducks. Headphones for all is the best solution, otherwise select a few in the class to demo their forms or present the demonstration yourself.

The following instructions are specific for Jaws® from Freedom Scientific.

1. Make sure the **Finished** form is loaded in Acrobat.
2. Run the screen reader if you have one installed, or listen to the instructor demonstrate it.

Alternately, you can open the file `jaws1.wav`, which is a recording of the Jaws screen reader reading the form in normal reading mode.

3. First, test the form in normal reading mode.

With the Jaws screen reader, for example, you would select the Acrobat document and use **Insert + Down Arrow Key** to begin reading it from the top of the page.

The screen reader will read the heading and instructional text block. It will then read over each field, using the ID and Name text elements, the Instructions element if it exists, and the Datum Short description from each parent Field element.

As the screen reader reads the fields, judge for yourself whether the descriptions are clear and logical. Can you easily distinguish individual fields as the screen reader processes them? Do the special instructions get read? Are all the Short descriptions in place? Does everything fit together grammatically?

You may not like the fact that the screen reader says *colon* when it encounters a colon in the field label, for example, or that the ID and Name text is essentially repeated in the Short descriptions. Some screen reader behavior, such as verbosity, reading speed, and even voice, can be modified in the screen reader's configuration settings. You could also modify the Short description, however, to remove the colon or to prevent redundancy with the ID and Name. For example, a Short description of *Enter your first name <Edit>* might read more smoothly and with a little more clarity than simply repeating the ID and Name text.

4. If you want to examine the accessible text in more detail, you can use the **Up** and **Down Arrow** keys (in Jaws), to move through the document one line at a time. When you reach a Field, the screen reader will read one component of the text for the Field -- ID, Name, Instructions if they exist, and Short description, each time you press the Down Arrow key. Using the Up Arrow key will move backwards through the document.

5. Next, tab through just the data fields by pressing the **Tab** key. Shift-tab will move the screen reader backwards through the fields. Each time you tab or shift-tab to a field, the screen reader will read the Short description. Judge whether the existing Short descriptions adequately convey enough information in the context of tabbing through the fields.

If you do not have access to a screen reader, open the file `jaws2.wav`. This is a recording of a user tabbing through the fields of the form.

6. If you are using Jaws, pressing the **Enter** key after tabbing to a field will place the screen reader in data entry mode. If you type into a Text data field, the screen reader will read each keystroke. If you press Enter at a Combo Box then use the arrow keys to navigate the drop down list, the screen reader will read each selection.

Accessibility Testing Summary

The first order of testing form accessibility is to carefully check the elements in the Forms Access Structure Tree. Make sure the top-down order of the elements in the Structure Tree accurately reflects the order in which the parts of the form document should be read. Make sure that any required alternate text for fields is in place as a value for either the Actual Text or Alt Text attribute of that Field. Make sure that a Short description exists for each Datum element, and that it makes logical and grammatical sense when read in conjunction with the other text for the Field. Verify that the tab order accurately reflects the order in which the data fields should be encountered by the user.

After you Finish the form in PDF Forms Access and open it in Acrobat, run the Accessibility Checker to verify that everything in the document that should be accessible really is. In regard to forms, the Accessibility Checker will flag any missing Short descriptions. These should be fixed back in PDF Forms Access.

The ultimate testing method is to verify how the document actually sounds when read by a screen reader. Using a screen reader can be tedious and slow, especially for large or complex forms, so you may want to use it somewhat selectively. Learning to use a screen reader effectively will take some time and practice.

PDF Form Structure and Layout



Form Structure and Layout Guidelines

In this module, you will see some examples of how Forms Access analyzes different form layouts, some better than others, and you will be presented with a set of suggested guidelines for use in designing forms for accessibility.

Module Contents

Topics	Exercises
Form Structure and Layout	Form Layout Guidelines

File to Download for Exercises in this Section

FormsAccessExamples.zip, which is normally found in the Exercises folder, contains the following files:

- **PersonalData.pdf**
 - **PersonalData_underline.pdf**
 - **PersonalData_unfinished.pdf**
 - **ProblemForm1.pdf**
 - **ProblemForm2.pdf**
 - **ProblemForm2mod.pdf**
 - **ProblemForm4.pdf**
 - **Table1.pdf**
 - **Table2.pdf**
 - **Table3.pdf**
 - **Table1_ext.pdf**
 - **Table1_org.pdf**
 - **Table1_org-new.pdf**
 - **Jaws1.wav**
 - **Jaws2.wav**
 - **form_1040.pdf**
 - **f1040_accessible.pdf**
- You should extract these files from the zip archive if they have not been extracted already.



Form Structure and Layout

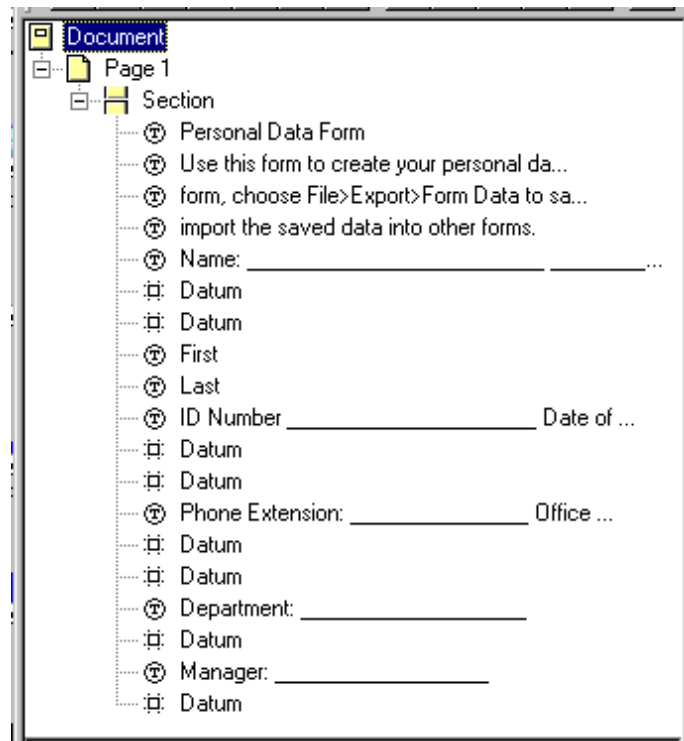
Now that you know a bit more about how PDF Forms Access processes forms, it would be useful to backtrack somewhat to an earlier subject, that of form design and layout. The layout of the original form document can influence the way PDF Forms Access generates the elements in the Structure Tree. Depending, for example, on how the numbering is done, how the boxes or lines for data entry fields are created, what type of fields are used, and how the labels are placed in the original document, PDF Forms Access will not always be able to analyze the components of a field accurately enough to generate a standard Field element for it in the Structure Tree, with its child ID, Name, and Datum elements.

PDF Forms Access was initially developed to create accessible Internal Revenue Service forms, such as Form 1040. In general, the closer a form is to the structure and layout of IRS forms, the more completely and accurately will PDF Forms Access be able to initialize it when you first open the form. Of course, this is not always possible or desirable, but fortunately, as you have seen, Forms Access provides a number of ways to modify and correct the Structure Tree after initialization.

If you do not have access to the original form created with the original authoring tool -- the one that was used in the first place to create the Acrobat file you initialized -- then you can no longer change the structure or layout of the form in anything but minor ways. You may have control over the design of the layout if you are creating the form from scratch, however. This section of the course will present some examples of form and field layouts that can confuse Forms Access during initialization, and will outline solutions to these problems. In the process, you will be introduced to another PDF Forms Access tool, the Slice Element Tool.

The exercises that follow will be somewhat more abbreviated than previous exercises. In the interest of time, you will step through the procedures but not Finish or test the sample forms.

1. Open the file **ProblemForm1.pdf** in PDF Forms Access.
2. Examine the Structure Tree.



Some inconsistencies with the Structure Tree for the original sample file, **PersonalData_2.pdf**, should be immediately apparent:

- No Field element tags have been created for any of the Datum items. Subsequently, no ID, Name, or Instructions child elements have been created. This is not necessarily a problem if the reading order of the elements in the Structure Tree is correct, and if you manually create Short descriptions for each Datum element in the tree.
- To a sighted reader, it would be obvious that the "Name" label in the original document refers to two separate fields, "First Name" and "Last Name", because the labels "First" and "Last" are positioned directly under each field. PDF Forms Access, however, is not aware of the conceptual connection between the labels "Name," and "First" and "Last". Consequently, the reading order of the form, as determined by the order in the Structure Tree, would be:

Name ... <Edit data field> ... <Edit data field> ... First ... Last

- The two field labels from each line in the document have been combined into a single Text element in the Structure Tree.
- There is a further problem caused by the form layout that you would not be aware of unless you tested the form with a screen reader. The horizontal line after each field in the form was created with underscore characters. If

the screen reader is configured to read all punctuation, it will dutifully say *underline* for each underscore character in the horizontal line. The user can configure the screen reader to be less verbose, but this is not the kind of burden the developer should put on the user if it can be avoided. Had the horizontal lines been created with a line tool, rather than as characters, they would have been tagged as *Artifacts* and would not be included in the logical read order. Rather than return to the original document to fix this, you will take care of it by altering the Structure Tree tags instead.

There are at least two ways to approach the situation:

- Wrap the Datum elements in Field tags and move the existing Text elements under the Name child element of each Field element.
- Alternately, wrap the Datum elements in Field tags, then wrap all existing Text elements in Artifact elements and recreate the accessible text as attributes of the Name children of the Field elements.

Optionally, after the Datum fields have been wrapped with Field elements, use attributes of the ID element to create an accessible text identification, which did not exist in the original document, for each field.

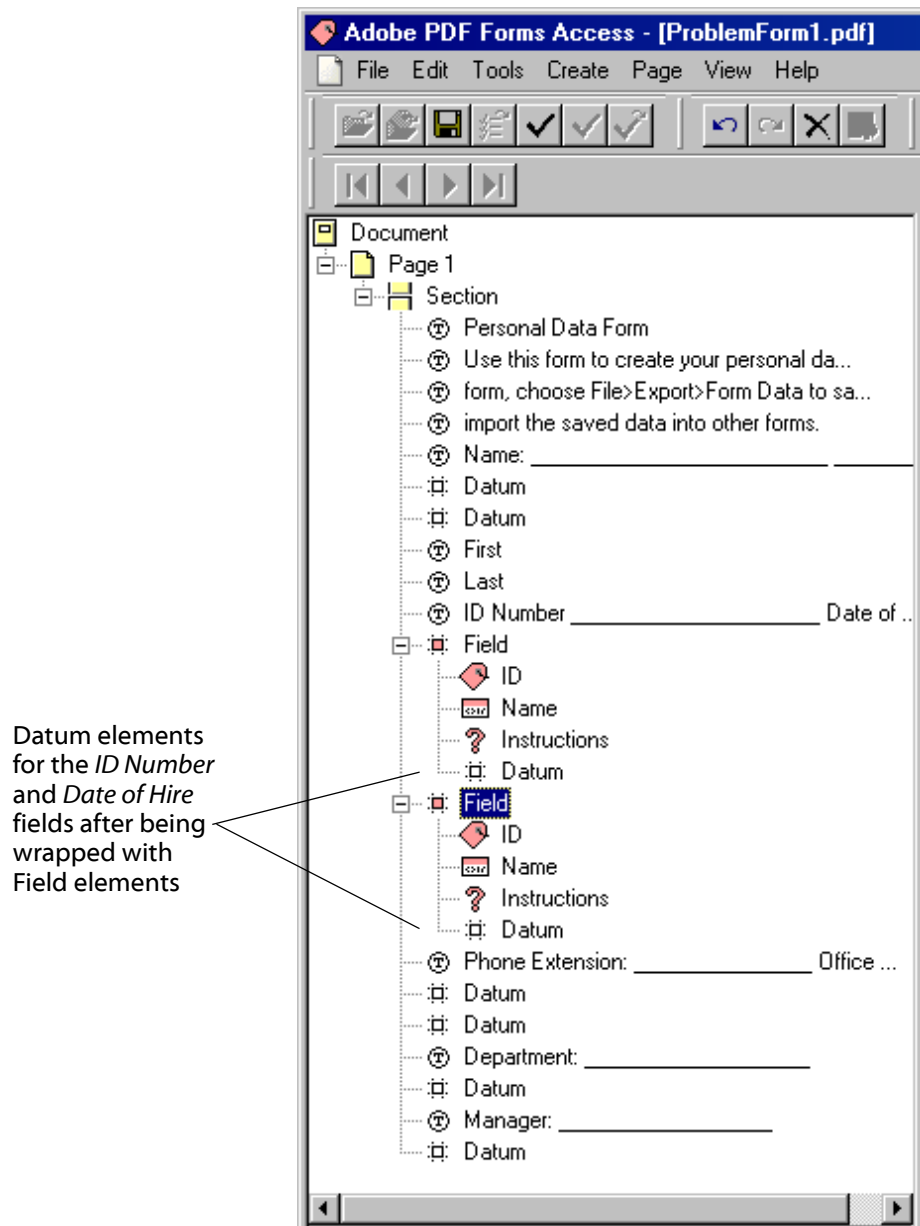
The choice of which of these methods to use depends to a great extent on the complexity of the form and the degree to which PDF Forms Access is able to successfully analyze the form. With very simple forms, such as the sample file you have been working with, if Forms Access is not particularly successful in analyzing the form it is sometimes easier to, in effect, throw the existing text away and recreate it using attributes in the Structure Tree. For complex forms containing complex labels or instructions, such as the IRS 1040, you will probably find it easier to move and restructure the existing text rather than recreate it.


Form Layout Example One, Option One: Using Existing Text

1. Start with the fields for *ID Number* and *Date of Hire*. Wrap each of the Datum elements for these fields in a Field element, as follows:

Click on the **Datum** element in the Structure Tree.

Select **Create > Field** from the Menu bar or click the **Create Field** button in the Toolbar.



2. You now want to move the existing text labels for the fields into the Name elements of each Field element. The immediate problem, of course, is that the two labels have been joined into one Text element, and, also, that you do not want the underlines to be spoken as part of the field's description. The solution is the use the Slice Element Tool  .

Click on the following **Text** element in the Structure Tree:

ID Number _____ Date of ...

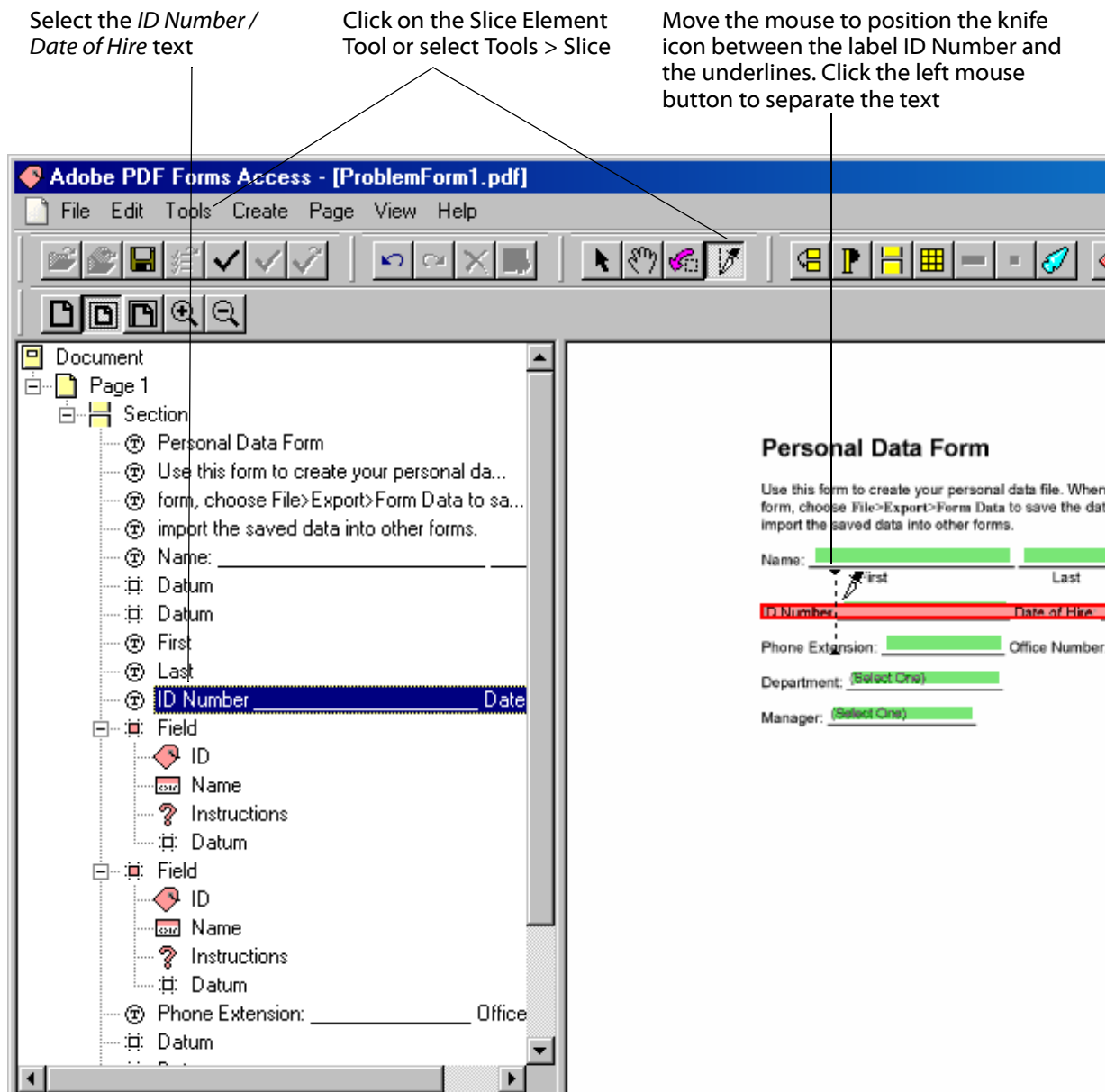
That line will be highlighted in the document in the View pane.

3. Select **Tools > Slice** from the Menu bar or click on the **Slice Element Tool** in the Toolbar.

A *knife* icon will appear, as in the following screen shot.

Without pressing any mouse buttons, use the mouse cursor to position the knife icon along the highlighted line of text in the View pane. The knife icon will only stop at places where it is legal to separate a line of text into two elements.

Place the knife icon between the text *ID Number* and the first underline and click the left mouse button.



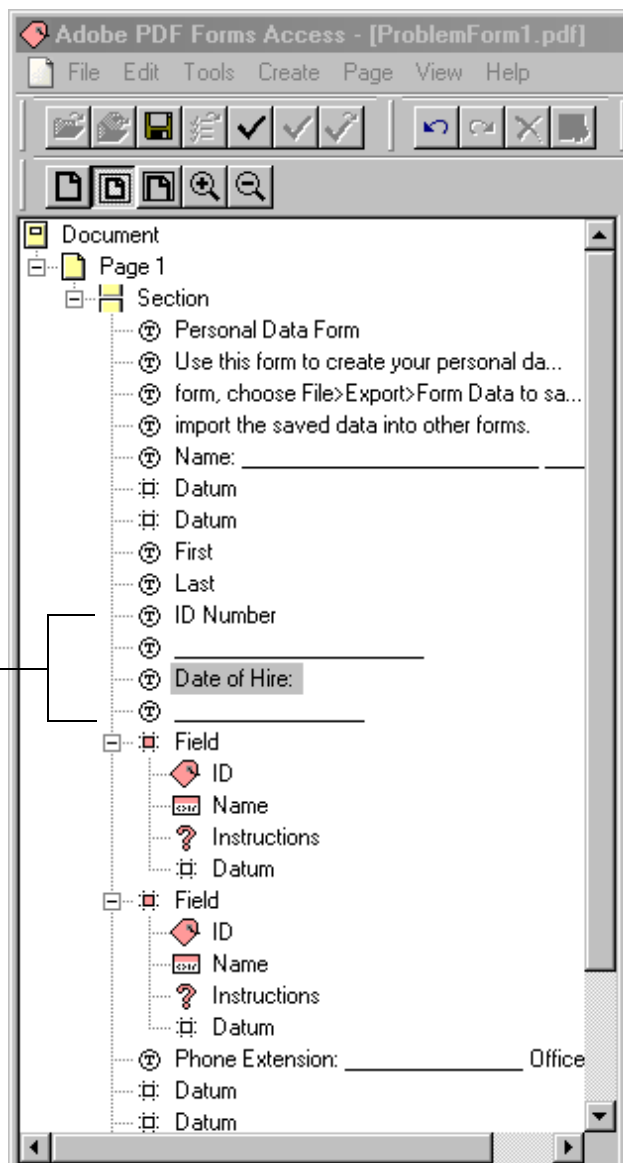
You will see that the text *ID Number* and the remaining text on that line in the document have been split into two Text elements in the Structure Tree.

4. Repeat **Step 3** as follows:

- Click on the Text element that contains the remaining text: *underlines, Date of Hire, underlines*.
- Click on the Slice Tool and separate the first group of underlines from the rest of the text.
- Click on the Text element that contains the remaining text: *Date of Hire, underlines*.
- Click on the Slice Tool and separate the text *Date of Hire* from the group of underlines.

This will result in four separate Text elements in the Structure Tree, as in the following screen shot:

Four separate
Text elements
after the Slice
Elements
operation



- Next, you will move the labels *ID Number* and *Date of Hire* into the Name child elements of their respective Fields.

Click on the Text element for *ID Number*, hold down the **left** mouse button, **drag** the Text element to the first Name element, and **drop** it there.

Click on the Text element for *Date of Hire*, hold down the **left** mouse button, **drag** the Text element to the next Name element, and **drop** it there.

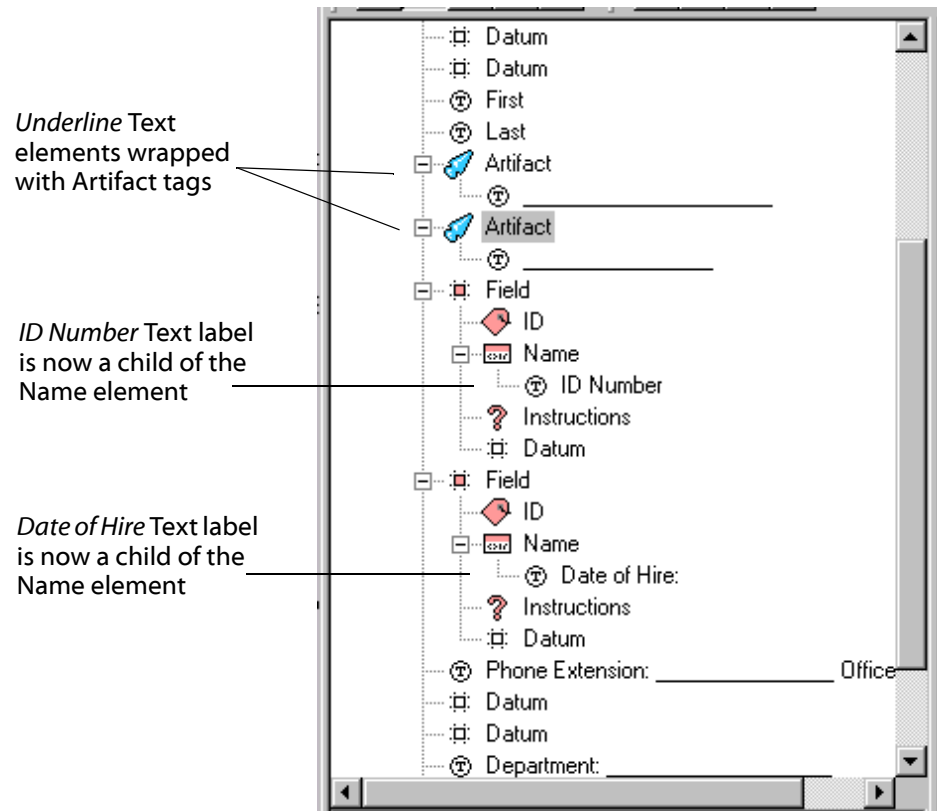
- Now, remove the underlines from the logical read order by wrapping them in Artifact elements.

Click on one of **Text** elements containing the *underlines*.

Select **Create > Artifact** from the Menu bar, or click on the **Create Artifact** button in the Toolbar.

Do the same for the other underline Text element.

Here is how the Structure Tree will look when you are finished with these two Fields:



- When you use this method -- creating Field elements and then using their Name element children to hold the Text labels for the Fields -- the Text for the Name element is automatically made available as one of the Short description options for the sibling Datum element.

Click on the *ID Number* Field's **Datum** element.

In the **Properties** palette, open the **Short description** drop down menu and select *ID Number*. You can modify the text if desired.

8. If you were to finish the form, which you will not do in this exercise, you would proceed with the other Fields according to the Steps above.

An option you might consider, although there are no field IDs in the original form, is to provide accessible text for the Field ID elements. To do this, you would provide a value for the ID element's *Alt text* (or *Actual text*) attribute. Screen readers will read the ID element's Alt text followed by the Text child element of the Name element.

Form Layout Example One, Option Two: Using Alternate Text

This approach is similar to the previous one in that you will wrap all Datum elements with Field elements. However, with this approach, you will discard all existing text elements by wrapping them in Artifact elements, and recreate the text using attributes of the ID and Name elements for each Field. As in the previous exercise, you will not complete the entire form.

1. **Close** the file `ProblemForm1.pdf`. Do *not* Save it. Then **Open** it again in PDF Forms Access.

In the Structure Tree, find the **Datum** elements for *First Name* and *Last Name*.

Click on the Datum element for First Name, then select **Create > Field** from the Menu bar or click the **Create Field** button in the Toolbar.

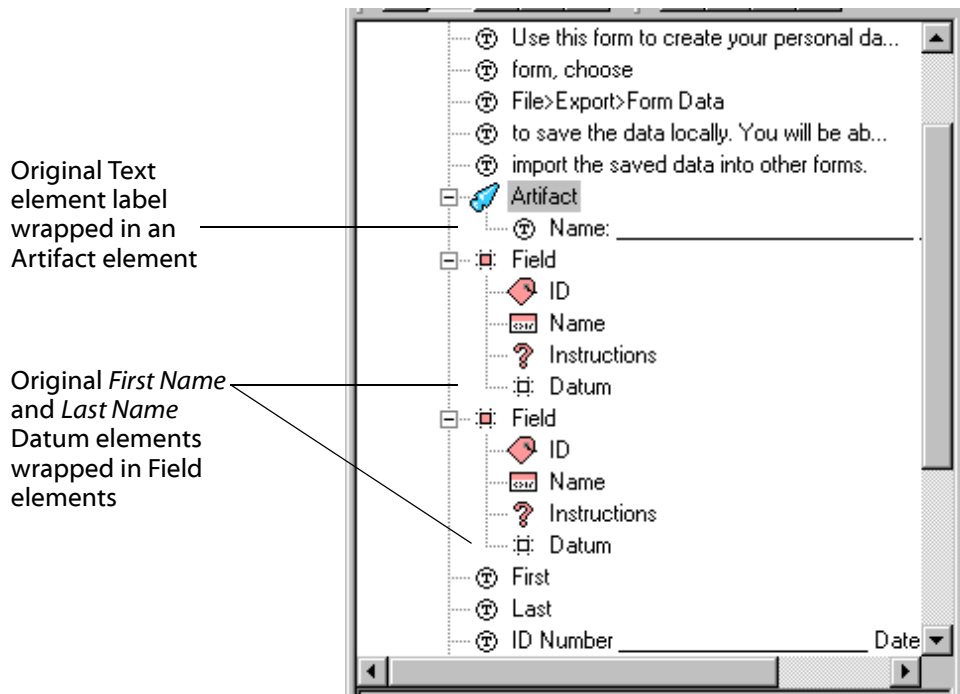
Repeat the step above for the *Last Name Datum* element.

2. Click on the Text element containing the *Name:* label.

Select **Create > Artifact** from the Menu bar or click the **Create Artifact** button in the Toolbar.

You have now removed all the original text pertaining to the *First Name* and *Last Name* fields. The next steps involve recreating the text as attributes of the Field elements' children. This will very similar to the steps you followed to add alternate text to the Manager field in an earlier exercise, [Using Element Attributes for Accessible Text](#).

Your Structure Tree should look like the screen shot below.

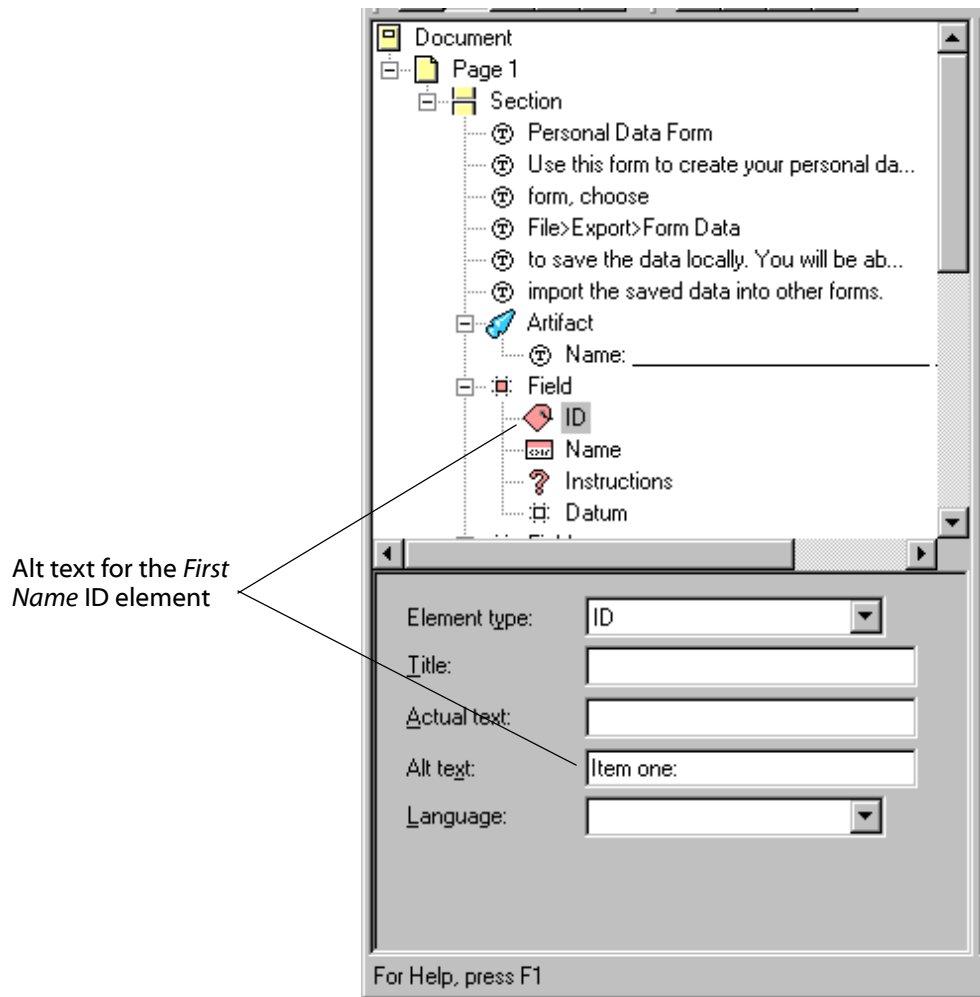


3. **Click** on the **ID** element under the *First Name* Field element. The **Properties** palette for the *ID* element will be displayed.
4. Although none exist in the original form, assume you wish to add an accessible text ID to each field.

In the **Alt text** attribute field for the *ID* element, type:

Item one:

or, any other form of ID you wish. Note that using the *Actual text* attribute would have the same effect when read by a screen reader.



5. Click on the **Name** element under the *First Name* Field element. The **Properties** palette for the *Name* element will be displayed.

In the **Actual text** attribute field, type:

First Name

6. Click on the **Datum** element under the *First Name* Field element. The **Properties** palette for the *Datum* element will be displayed.

In the Short description attribute field, type:

Enter your first name.

When a screen reader encounters the First Name Field, it will not read the original text that you converted to Artifacts, but instead will read the alternate text from the Field's child elements as follows:

Item one <colon> First Name Enter your first name <Edit>

7. To complete the form, which you will not do in this exercise, you would repeat Steps 2 through 6 for each Text and Datum element in the Structure Tree.

Form Layout Example One Summary

The structural and layout characteristics of a form document -- in particular, the layout of fields and their associated labels and IDs, if any -- will affect the thoroughness with which PDF Forms Access can process the form. Forms Access processes the form one line at a time. In the case of the sample file **ProblemForm1.pdf**, Forms Access was not able to discern any structures within any of the lines in the form with the following basic pattern:

<optional ID> <label> <field>

Why not? At first glance, the form looks as if at least some of the fields adhere to that pattern. The root of the problem lies back in the original document where the data entry lines for each field were created with underline characters rather than with graphic artifacts. Data fields for the form were then created in Acrobat and placed on top of the underline characters. PDF Forms Access interpreted the underline characters as part of the label. Since both labels and their underlines together occupy the entire line, there was, in effect, no room left on the line for the actual data fields. Forms Access could not determine a connection between the Text labels and the two data fields hovering above the underlines, so it created individual Text and Datum elements for them and placed them in the Structure Tree as direct children of the Section element, rather than creating parent Field elements for them.

If you are the developer of the original form, you may wish to avoid using characters such as underlines for creating lines and boxes for data entry. Use the authoring software's graphic tools instead, if such exist. These will be tagged as *Artifacts* when PDF Forms Access initializes the form and, consequently, they will not interfere with processing the fields.

If the original authoring tool does not have graphics capabilities, leave the data entry areas blank. When you create the data fields in Acrobat, you can specify a border style for the field, such as *Underline*, which will be visible in the form document.

If you cannot change the original document, use the methods described above to make adjustments to the Structure Tree if PDF Forms Access is not able to interpret the field structure of the form.

Form Layout Examples Two and Three

Next, you will initialize two variations of the sample form and briefly look at the resulting Structure Tree. These two files use graphic lines rather than underline characters for the data entry lines. The graphic lines are converted to *Artifacts* during initialization, thereby avoiding one of the problems in the previous form. However, Forms Access will not be able to find a match for the pattern *<optional ID> <label> <field>* in the first of the sample files. A slight adjustment to the field layouts in the original PDF form will solve the problem, as you will see in the second sample file.

Just observe the Structure Trees after you initialize the files. The methods you would use to fix the trees are similar to those have already used, so you will not do any repair work on these examples.

1. **Open** the file **ProblemForm2.pdf** in Forms Access. In this form's layout, the IDs and labels of the first six fields are placed on the line above the data field.

Observe the Structure Tree. Notice that the data entry lines for the fields have been converted to Artifacts.

The field IDs and the labels for each field were not recognized as being separate items. None of the Datum elements were wrapped in Field elements because PDF Forms Access could not recognize any patterns consisting of an optional ID, a label, and a field.

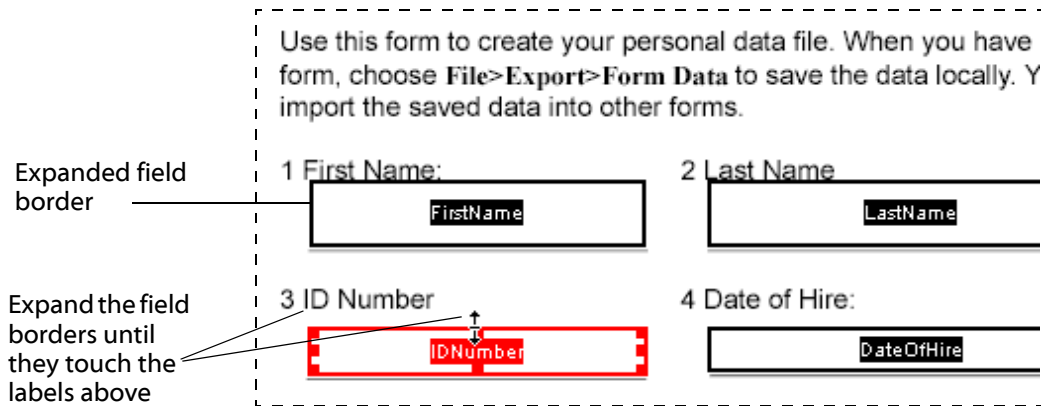
The screenshot displays the Adobe PDF Forms Access interface. The title bar reads "Adobe PDF Forms Access - [ProblemForm2.pdf]". The menu bar includes "File", "Edit", "Tools", "Create", "Page", "View", and "Help". The toolbar contains various icons for file operations, navigation, and editing. The Structure Tree on the left shows a hierarchy: Document > Page 1 > Artifact > Section > Personal Data Form. Below this, a list of elements is shown, including labels like "1 First Name:", "2 Last Name", "3 ID Number", "4 Date of Hire", "5 Phone Extension:", "6 Office Number:", "7 Department:", and "8 Manager:", each followed by a "Datum" icon. The main window displays the "Personal Data Form" with the following text: "Use this form to create your personal data file. When you have completed form, choose File>Export>Form Data to save the data locally. You will import the saved data into other forms." Below this are eight input fields: 1 First Name, 2 Last Name, 3 ID Number, 4 Date of Hire, 5 Phone Extension, 6 Office Number, 7 Department (with a "(Select One)" dropdown), and 8 Manager (with a "(Select One)" dropdown).

You could use the different methods presented so far to fix various problems with this form in the Structure Tree:

- Create parent Field elements for each Datum element.
- Move the existing text labels into the Name child element of each Field.

- Optionally, use the Slice Tool to split the text label into an ID and a Name section and place each under their respective elements.
- Alternately, create accessible text for Fields as attributes of the ID and Name child elements, and wrap existing Text elements in Artifact tags.
- Create or select a Short description for each Datum element.

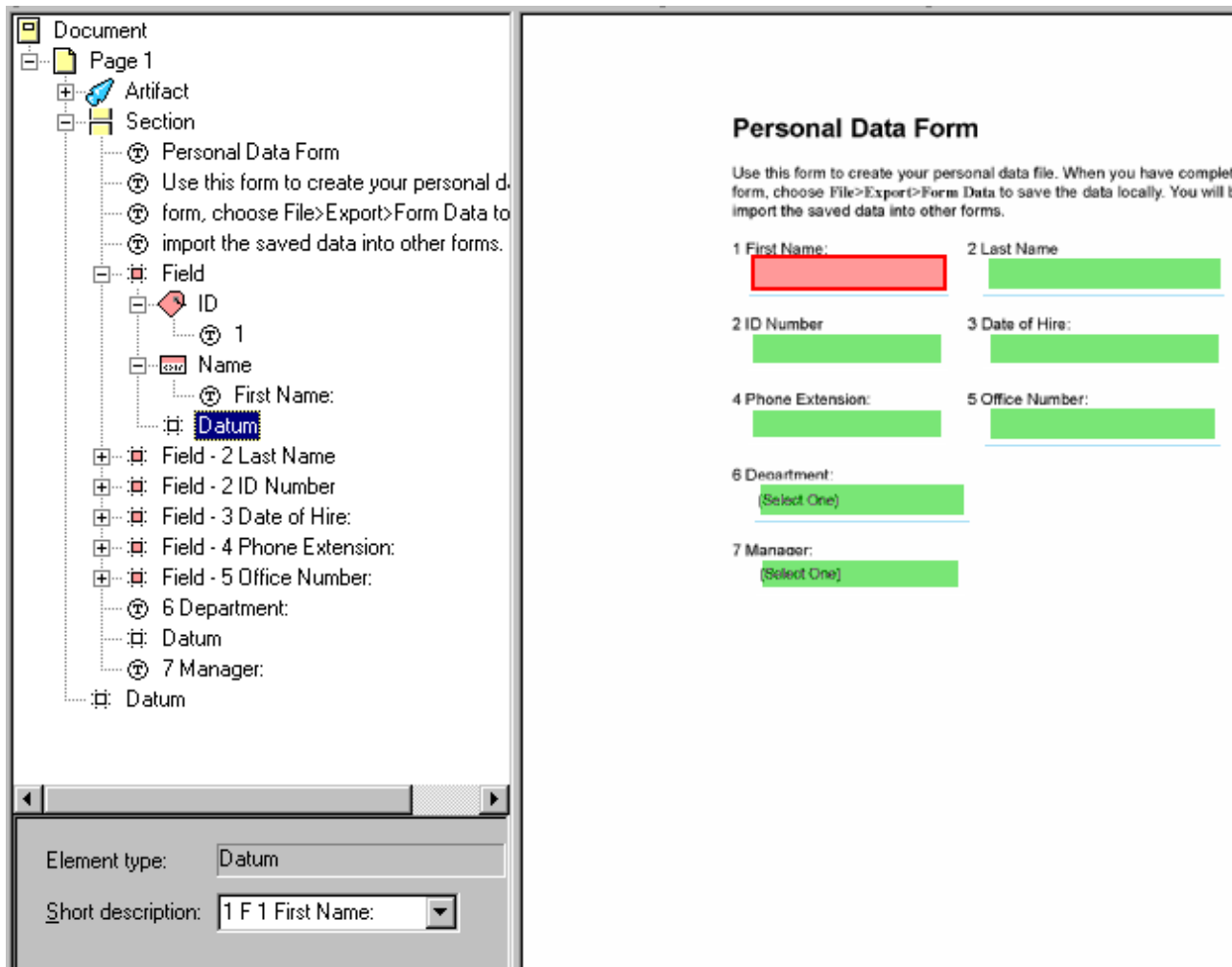
However, the real fix is to go back to the original Acrobat document where the fields were created, and slightly enlarge the field borders so that they touch or border the IDs and labels above them, as in the screen shot below, then re-initialize the form with PDF Forms Access:



2. Close the file `ProblemForm2.pdf`.

Open `ProblemForm2mod.pdf` in Forms Access. This is a version of `ProblemForm2.pdf` that has had the field borders adjusted as described above.

With the usual exception of the Combo Boxes, as you can see in the screen shot below, adjusting the field borders allowed Forms Access to recognize each field's ID, label and data field as a related group.



Form Layout Examples Two and Three Summary

What is the probable reason for this odd behavior? Stretching the data field's boundary up to touch the line above allowed it to be noticed by Forms Access, which processed lines one at a time. Note that the data field's fill is transparent, so the exaggerated boundaries will not be particularly apparent when viewed in Acrobat, but it would have been helpful if the underlines for the data fields had been placed closer to the labels in the original document.

Form Layout Example Four

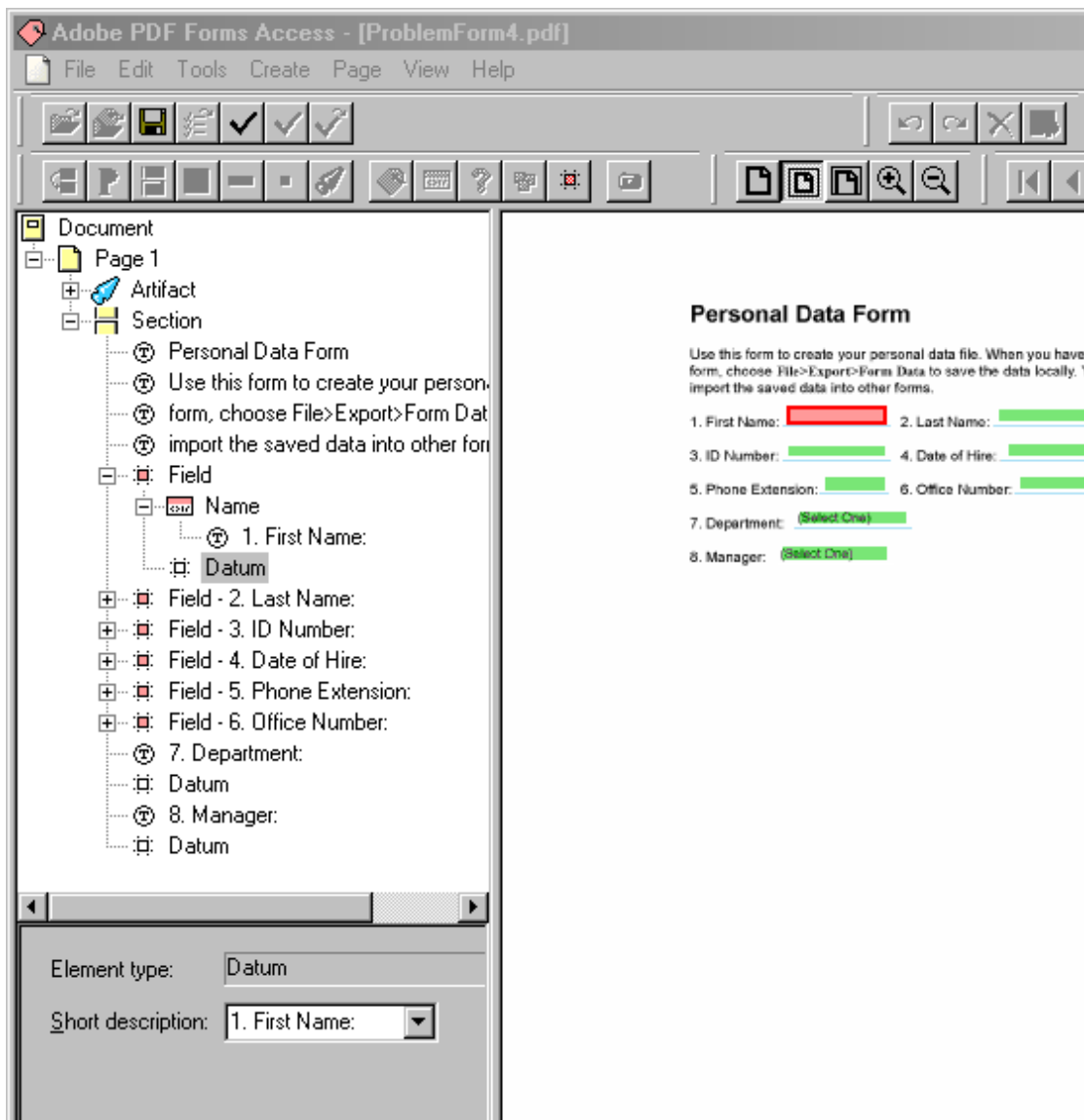
The fourth example file is quite similar to the `PersonalData_1.pdf` form. Each data field is on the same line as its associated ID and label, arranged in the

pattern `<ID> <label> <field>`. A graphic tool was used to create the data entry lines.

1. **Close** `ProblemForm2mod.pdf` without saving it.

Open the file `ProblemForm4.pdf` in PDF Forms Access.

Examine the Structure Tree:



The Structure Tree is very similar to that of the form `PersonalData_2.pdf`:

- PDF Forms Access has correctly analyzed the first six fields along with their IDs and labels, and has created Field elements for them.

- Each Field element contains a Name child element which in turn contains the Text label.
- The Short description for each of the Datum elements that were wrapped in a Field element has been generated automatically.

You may notice something a bit curious, however: PDF Forms Access did not create separate ID elements for the field numbers as it did with **PersonalData_2.pdf**.

The *only* difference between the numerical IDs in this form and those in **PersonalData_2.pdf** is that these IDs have periods after them. And, indeed, PDF Forms Access will not recognize an ID as an ID element if it is followed by a punctuation mark. The reason lies back in the genesis of PDF Forms Access: it was originally created to process IRS forms, and the ID numbers in IRS forms are separated from their accompanying labels by a space or tab only.

Form Layout Example Four Summary

When PDF Forms Access initializes a form, it will automatically create Field elements with nested ID and Name element children for the data fields in a form, under the following conditions:

- In the original document, the field identification, the label, and the data field are all on the same line and in that order, or, the data field is touching part of the label on the line above it.
- The data fields themselves, which are created in Acrobat, are Text entry type fields.
- The field identification text is separated from the accompanying label text by a space or a tab, not a punctuation character.

Creating Accessible Form Tables



Introduction to Form Tables

In this module, you will be introduced to form tables, which are tables whose cells contain fillable data fields, and will be presented with examples and guidelines for making form tables accessible using PDF Forms Access.

Module Contents

Topics	Exercises
Form Tables	Creating Accessible Form Tables

File to Download for Exercises in this Section

FormsAccessExamples.zip, which is normally found in the Exercises folder, contains the following files:

- **PersonalData.pdf**
 - **PersonalData_underline.pdf**
 - **PersonalData_unfinished.pdf**
 - **ProblemForm1.pdf**
 - **ProblemForm2.pdf**
 - **ProblemForm2mod.pdf**
 - **ProblemForm4.pdf**
 - **Table1.pdf**
 - **Table2.pdf**
 - **Table3.pdf**
 - **Table1_ext.pdf**
 - **Table1_org.pdf**
 - **Table1_org-new.pdf**
 - **Jaws1.wav**
 - **Jaws2.wav**
 - **form_1040.pdf**
 - **f1040_accessible.pdf**
- You should extract these files from the zip archive if they have not been extracted already.



Form Tables

[Instructor Notes: The next two sections, Form Tables and Importing Structures, represent a transition to somewhat more advanced topics, building on the fundamentals that have been discussed to this point.](#)




Creating accessible tables presents some special challenges. Fully sighted persons are able to recognize and orient themselves very quickly to even very large tables. A glance across the column headings and down the rows will, in most cases, allow sighted persons to quickly grasp the general meaning and purpose of a table, although fully comprehending or processing its contents will take longer. A sighted person might be able to immediately recognize, for example, that only a few cells in the entire table are of personal interest or relevance. Sequential devices like screen readers, however, must process table structures sequentially, one cell at a time and one row at a time. One of the challenges, therefore, in making tables accessible is to provide the user with an overall description of the table and what it contains, as well as a sense of orientation within the table.

Creating accessible form tables -- that is, tables that contain data entry fields -- adds to the general challenge of creating accessible tables. You might imagine filling out fields in a table in an electronic form without the benefit of a monitor, just using a screen reader. What would you need to know -- and what would you want to know -- about the table and about each cell in the table in order to successfully complete the form?

- First of all, it would be nice to know that the screen reader was about to read a table.
- In terms of orientation, it would be helpful to know how many columns and rows the table contained.
- In terms of conveying the overall meaning of the table, it would be helpful to have the screen reader read the column headings.
- As the screen reader encounters each cell, it would be helpful to know the column and the row number of the cell.
- If the table contains data fields, you would need to know the name of the data field and would want to be aware of any special instructions for entering data that might be associated with it.

PDF Forms Access provides the tools to make this kind of information about form tables accessible.

You saw examples earlier of how the layout and structure of a form document can affect the degree to which Forms Access recognizes Field element patterns in the form. The same thing holds for form tables: the way they are structured and the characteristics of the fields within them can affect the way PDF Forms Access analyzes the table and the fields within it. Fortunately, it is relatively simple to reprocess form tables that Forms Access could not initialize in quite the right way, and Forms Access provides tools for this purpose.

Upon initialization, PDF Forms Access will always recognize the data fields within a table and assign *Datum* elements to them in the Structure Tree. Forms Access will not necessarily recognize the table structure as a table, however. When Forms Access is able to recognize a table pattern, it creates a **Table** element  in the Structure Tree, with Table Row  and Table Cell  child elements, for the heading and data fields within the table. When it cannot recognize a table structure, Forms Access will usually create **Section** elements for each row with the row's Datum elements as children. In that case, you will Create a Table element to wrap the Datum elements from the original table.

In the exercise to follow, you will initialize three simple form tables with slightly different layouts, each of which will produce a different result in the Structure Tree. You will then use Forms Access table tools to make adjustments to the Structure Tree.



Exercise: Creating Accessible Form Tables

In order to do any of the exercises in this module, you must have Acrobat 5 or higher and PDF Forms Access 1.0 or higher installed on your machine, and you must have copied the sample files in the archive **FormsAccessExamples.zip** to a folder on your machine.

In this exercise, you will initialize three sample form tables with PDF Forms Access. The first two form tables differ in only one respect: the row heading of one table is left justified and the row heading of the other is center justified. Nevertheless, initializing the two form tables will result in Structure Trees with somewhat different characteristics, each of which can have its advantages. You will modify one of the Structure Trees using Forms Access table tools. The third form table differs from the first one only in the way the data fields were originally created in Acrobat, but it will also result in a Structure Tree with different characteristics than the first. This will lead in to a discussion of practices to use and to avoid when creating data fields inside tables.

Sample Form Table One: Left Justified Column Headings

The first sample form table has left-justified headings for the columns *First Name*, *Last Name*, and *Phone*.

1. Open the sample file **Table1.pdf** in PDF Forms Access to initialize it.

As you can see, Forms Access did a reasonably good initial job of interpreting the form table:

- It created a *Table* element for the entire table, including the headings. You can click on the Table element to see the highlighted table in the View pane.
- It created *Table Row* elements for each row in the table.
- It created *one* parent *Table Cell* element within each Table Row for *three* child *Datum* elements.

This is probably the major difference between the first sample form table Structure Tree and the one that will result from processing the second form table, which you will have to do in part manually.

PDF Forms Access has, in effect, interpreted each row of the table as being composed of one cell which in turn is composed of multiple related data fields.

Adobe PDF Forms Access - [Table1test.pdf]

File Edit Tools Create Page View Help

Document

- Page 1
 - Artifact
 - Section - Sample Table Table 1:
 - Table
 - Table Row
 - Table Cell
 - First Name
 - Last Name
 - Phone
 - Table Row
 - Table Cell
 - Datum
 - Datum
 - Datum
 - Table Row
 - Table Cell
 - Datum
 - Datum
 - Datum
 - Table Row
 - Table Cell
 - Datum
 - Datum
 - Datum

Element type: Datum

Short description: row 1; First Name Last Name Phone

Sample Table

Table 1:

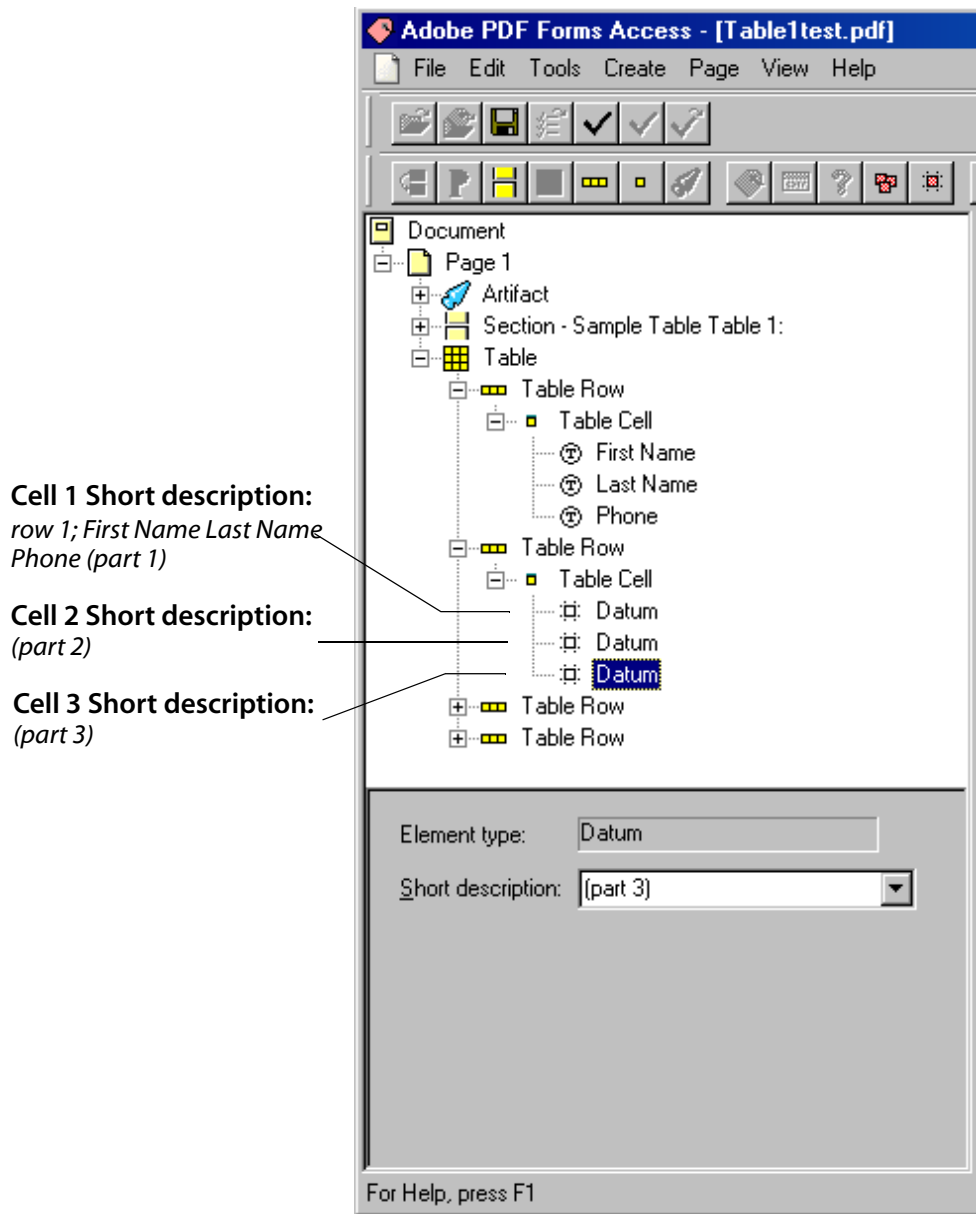
First Name	Last Name	Phone

2. Click on the first **Datum** element in the table and observe the **Short description** in the Properties palette.

You can see that it includes a row identifier, *row 1*; followed by all three column headings, *First Name Last Name Phone*, followed by a part number (*part 1*) identifying this as the first data field of a related group.

3. Click on the second **Datum** element for that row. Its Short description reads simply (*part 2*).

4. **Click** on the third **Datum** element for that row. As you might have guessed, its Short description reads simply (*part 3*).



Depending on the circumstances, this may be exactly what you want. In situations in which the user is very familiar with the form and enters data into the form on a regular basis, you might want to abbreviate field descriptions to allow faster interaction with the form. Or, you might use this method for the entry of data whose pattern is familiar to most people, such as area code and phone number or Social Security number, for example:


row 1: Social Security Number (part 1) <Edit> (part 2) <Edit> (part 3) <Edit>

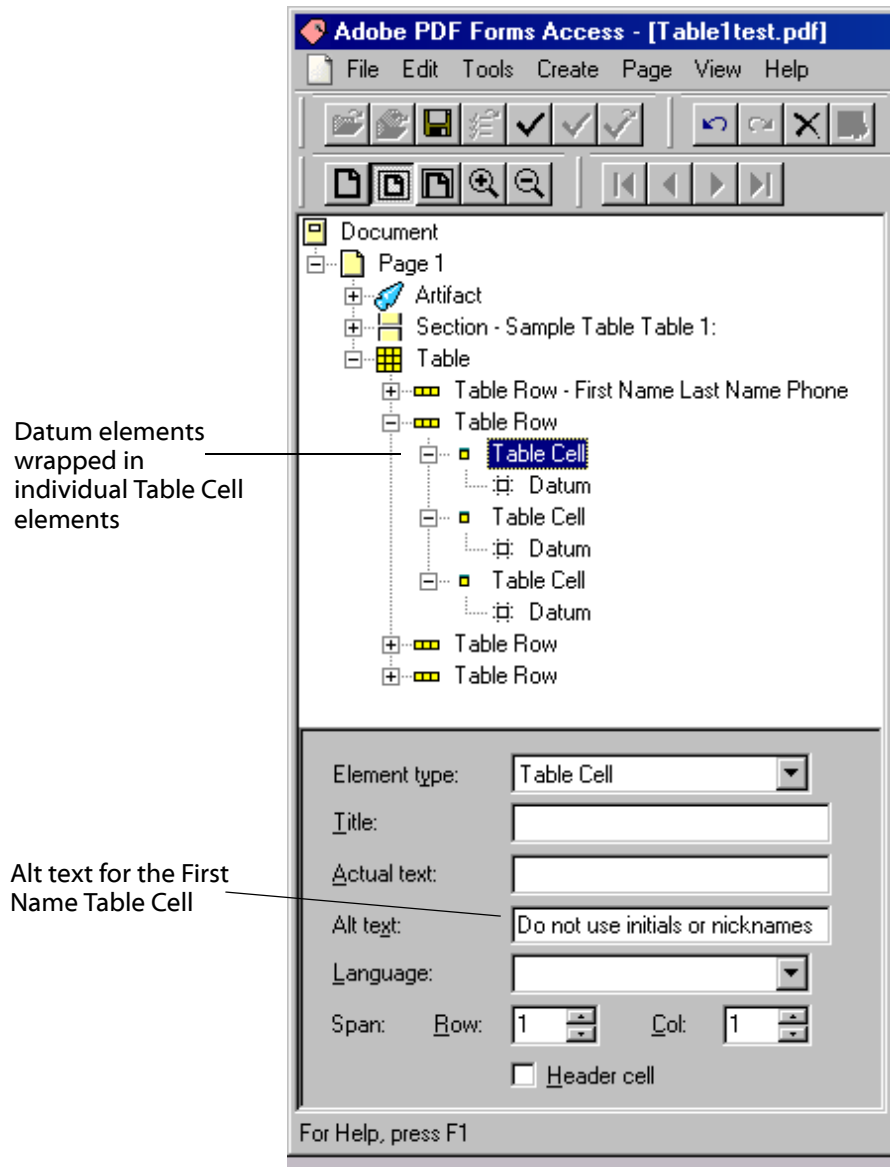
In the above example, it would be helpful to the user if the three data entry fields were restricted to three, two, and four digits, respectively.

5. Suppose you do not want all the data fields in a row to be treated as a group, or suppose that even if all the data fields are part of a group, you want to give them specific Short descriptions rather than part numbers?

The immediate solution in PDF Forms Access is to modify the **Short descriptions** of the **Datum** elements.

You might want to give each Datum element a Short description consisting of its column label, with or without the row number. It might be sufficient to include the row number for just the Short description that is read for the first field in each row. Again, your approach will depend on the circumstances; above all, strive for clarity.

Another approach that might be useful at times is to wrap the Datum elements in Table Cell elements  and then modify the Datum elements' Short descriptions, as above, or add accessible text to the Table Cell elements' attributes, or both. This would allow you, for example, to add special instructions selectively to one or more of the data fields in a row by adding text to the Alt text or Actual text attributes of the Table Cell element. That text would be read by screen readers in normal read mode, but not tabbing mode, just before the Short description for the field was read. See the screen shot below for an example.



Sample Form Table Two: Center Justified Column Headings

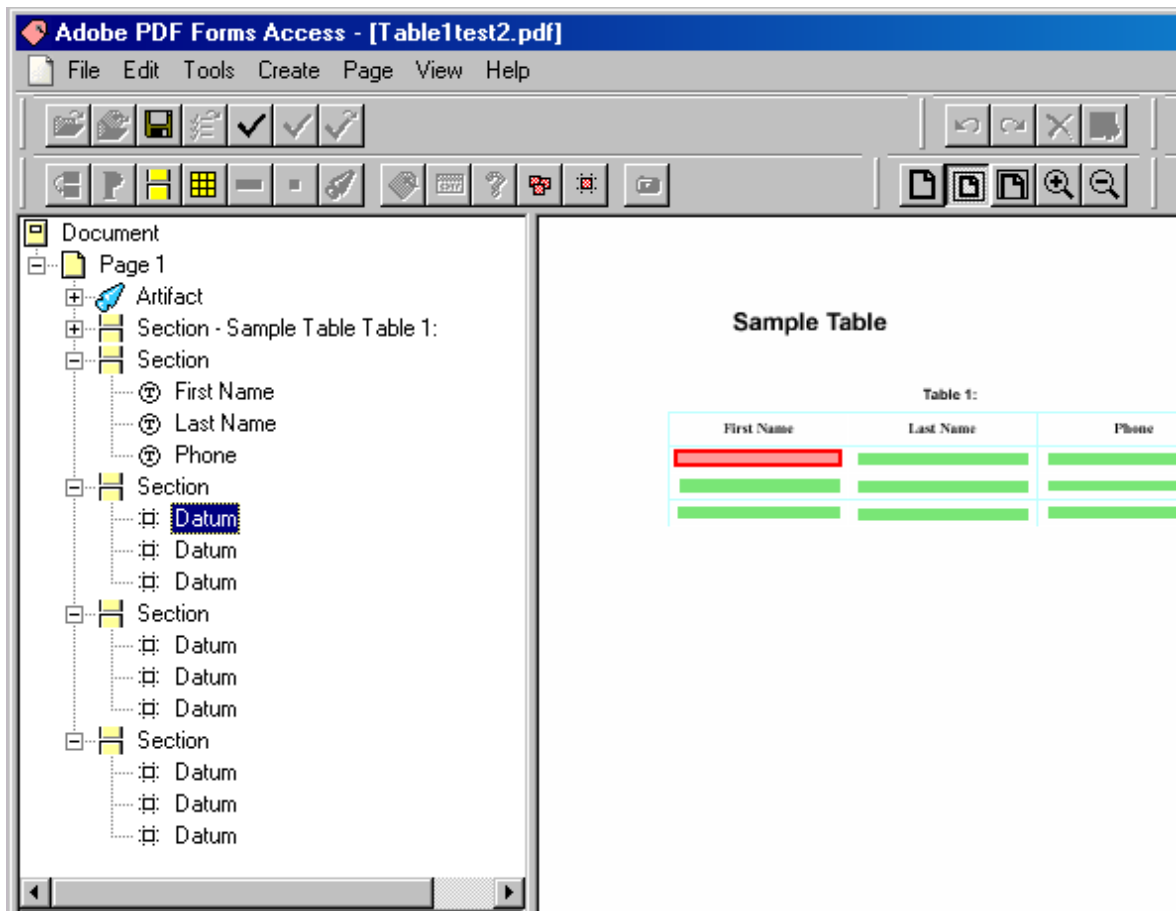
Sample form table two has center-justified headings for the columns *First Name*, *Last Name*, and *Phone*. This is the only way in which it differs from the first sample form table.

1. Open the sample file `Table2.pdf` in PDF Forms Access to initialize it.


You will notice immediately that Forms Access did not create Table elements for the table, but instead created *Section* elements for each row in the table.

2. **Click** on one of the **Datum** elements.

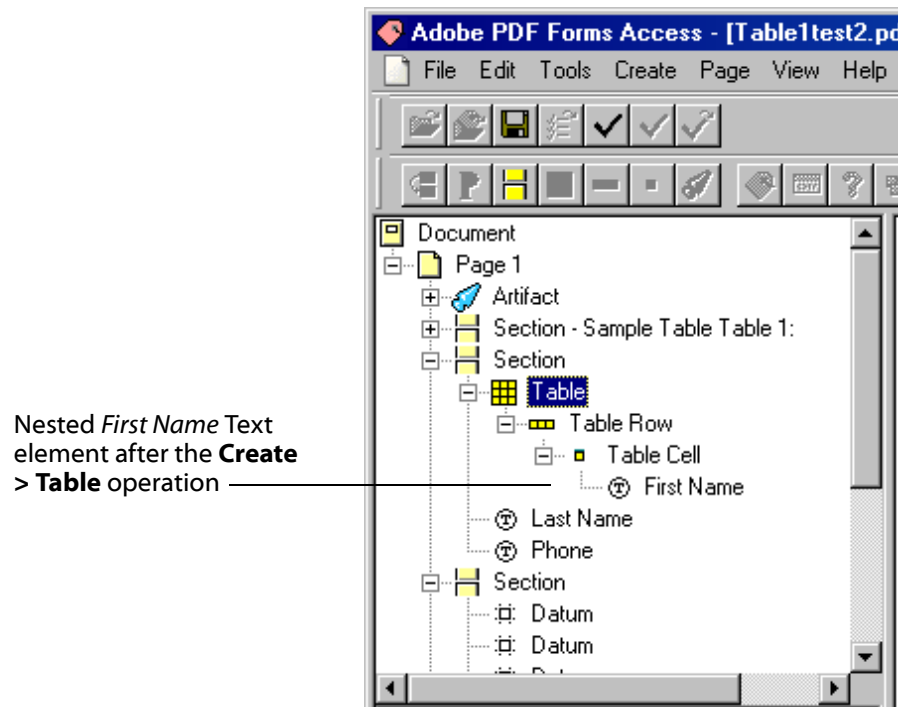
Notice in the Properties palette that there is no Short description for this or any other Datum element.




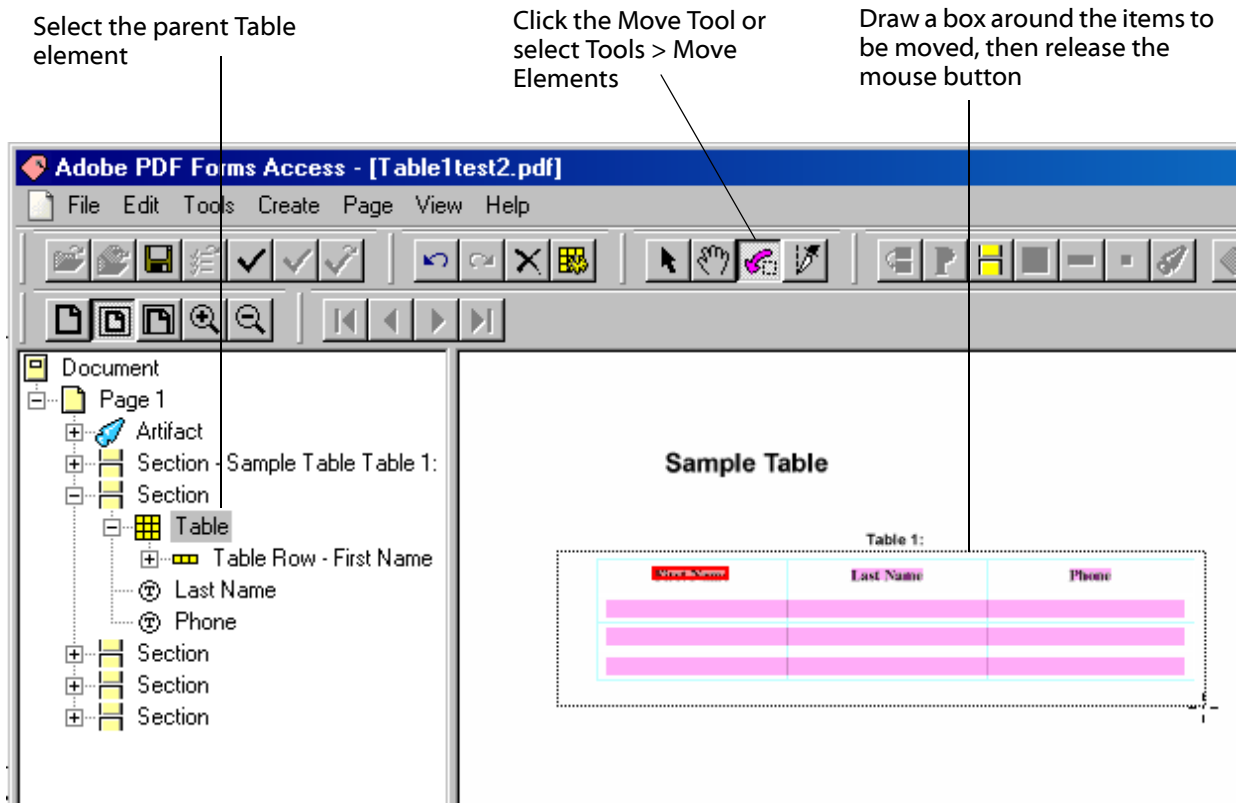
This will need some fixing, and all because of a little variation in the way the column headings are justified. Luckily, the fix is quite easy, and will even produce some unexpected benefits.

3. **Select** the first item in the table from either the Structure Tree or the View pane. In this case, that would be the first column heading cell, **First Name**.
4. Select **Create > Table** from the Menu bar or click the **Create Table**  button in the Toolbar.

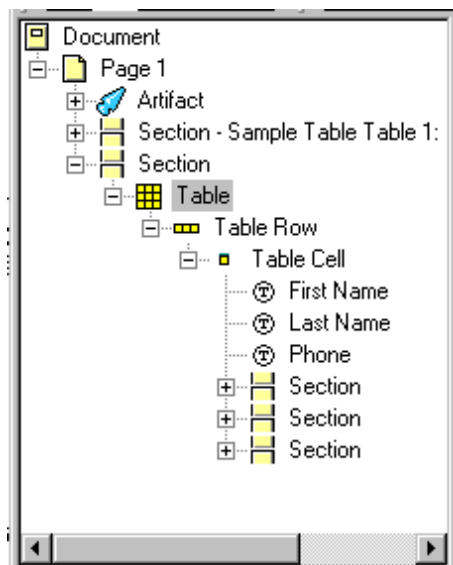
This will wrap the *First Name* Text element in a **Table Cell** element, which in turn is wrapped in a **Table Row** element, which in turn is wrapped in a **Table Cell** element, as in the screen shot below:




5. Make sure the new **Table** element is highlighted, then **click** on the **Move Tool** button, , in the Toolbar, or select **Tools > Move Elements** from the Menu bar.
6. Hold down the **left mouse button** and **draw** a box around all cells in the table, including the column headings. **Release** the mouse button.



The remaining column headings and all the Section elements with their Datum element children will be moved under the Table element as children.



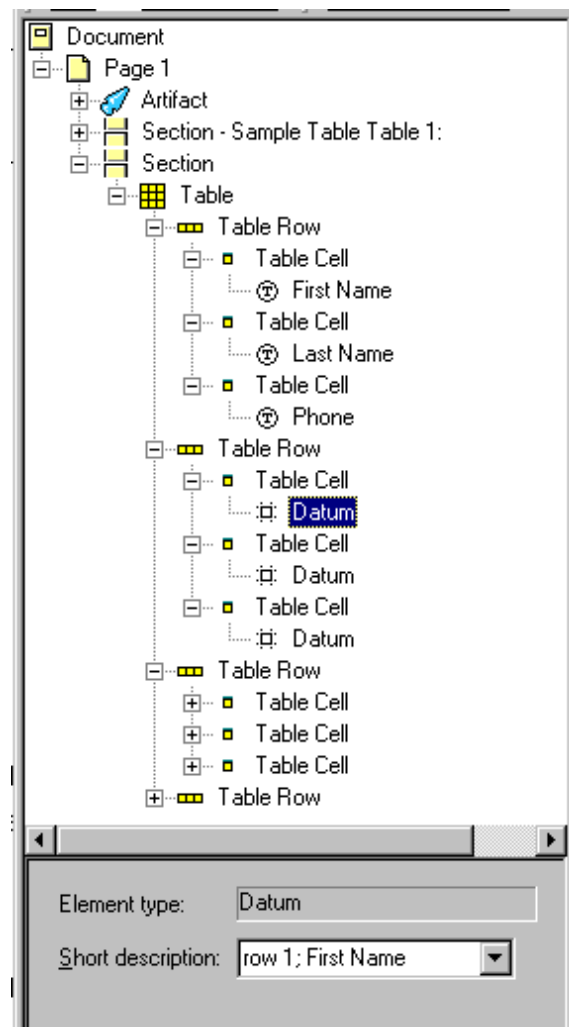
7. Finally, with the **Table** element still highlighted, select **Edit > Recognize Table** in the Menu bar, or click the **Recognize Table**  button in the Toolbar.

Note the following:

- The Section elements have been converted to *Table Row* elements.
- The contents of each individual cell, including the column headings, have been wrapped in *Table Cell* elements.
- Short descriptions have been automatically generated for the *Datum* elements. These consist of the row number plus the heading text for that column, as such:

row 1; First Name ... row 1; Last Name ... row 1; Phone ...

row 2; First Name ... row 2; Last Name ... row 3; Phone ...



You have seen an example of how different form table layouts can result in different Structure Tree arrangements upon initialization. Keep in mind that the Structure Trees of both sample form tables could have been modified with relative ease after initialization to resemble the other.

Sample Form Table Three: Methods for Creating Fields Inside Tables

Instructor Notes: This exercise is mostly observational -- students will not do any work on sample form three. The intention of the exercise is to bridge into a discussion of recommended methods for creating fields in tables.

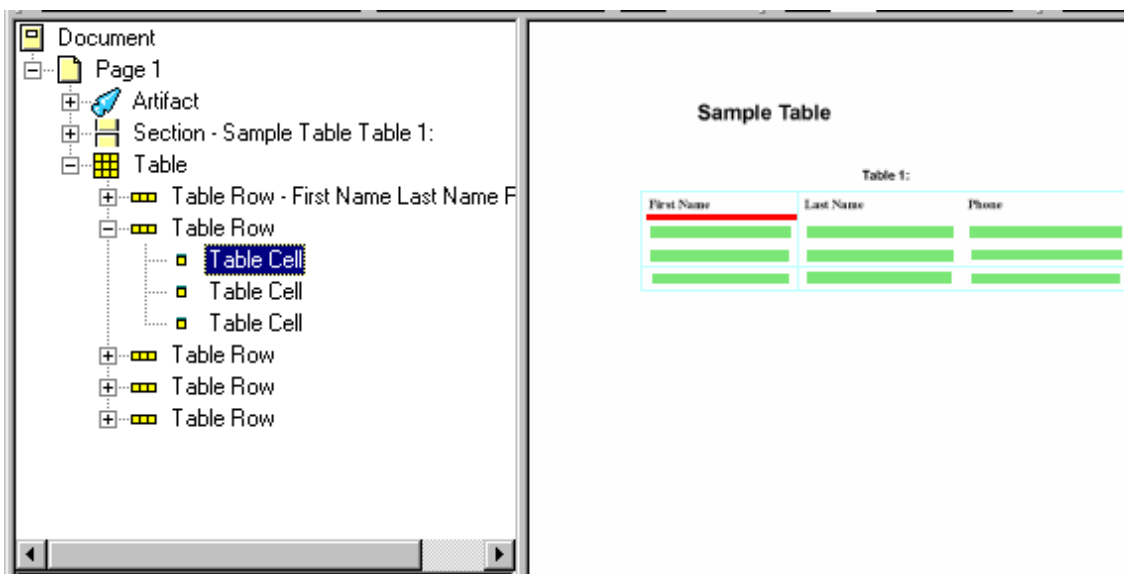
The third sample form table resembles the first sample form table in that the column headings are left-justified. It differs from the first two in the way the data fields were originally created for the table in Acrobat. The data fields for the third sample were created individually, one at a time, and given unique names. The data fields for the first two samples were created by replicating, as opposed to copying, one row of data fields into the rest of the table, and then modifying the names that Acrobat generates automatically for the replicated fields.

In this exercise you will first see an example of how, under certain circumstances, different methods of creating fields in tables can produce unexpected results in the Structure Tree. This will lead into a general discussion of recommended practices for creating data fields in tables.

1. Open the sample file `Table3.pdf` in PDF Forms Access to initialize it.

At first glance, the resulting Structure Tree looks very much like the first sample form table. PDF Forms Access has created a Table element, and Table Row elements for the column headings and the three data field rows.

But, wait a minute -- there are five Table Row elements rather than four as there should be. If you click on the second Table Row element, you will see in the View pane that it is a narrow row in between the column heading row and the first data field row. If you expand the tag, you will see that it contains three empty Table Cell elements.



Furthermore, if you expand the *Table Row* tag for the first data field row, then expand the first *Table Cell* child element, then click on the *Datum* element, you will see that its Short description reads:

row 2; First Name

In other words, Forms Access is counting the ghost row with the empty cells as the first row in the table. When a screen reader encounters the table, it will announce:

Table with three columns and five rows

However, the screen reader will not read the empty Table Cells, so the first thing it will say after reading the headings is, *Row 2; First Name*, leaving users to wonder, *What happened to row one?*

Your only choice, in this case, is to Delete the empty Table Row element and then edit the Short descriptions to use the correct row number. If you delete the empty row and then use Edit > Recognize Table, Forms Access will put the ghost row back into the Structure Tree again.

By the way, the extra empty row behavior described above does not happen if the table's data fields are created one at a time in Acrobat, but the column headings are center-justified. In that case, the initial Structure Tree will be exactly like the second sample form table: it will have the correct number of rows for the table, but you will still have to add a parent Table element to the Structure Tree.

Recommendations for Creating Fields in Tables

Creating data fields one at a time for a multiple-column, multiple-row table is not the best way to go about the task anyway, getting proportionally more tedious with the size of the table. Each individual data field must be drawn and a unique Name entered for it in the Field Properties dialog.

Do not use Copy and Paste to create data fields in the table in Acrobat unless you intend to have an exact duplicate of the copied field in some other location in the table. You cannot create unique Names for copied fields: if you change the Name of one, the Name of the other will also change. Data entered into one field will be reflected in its copy, and vice versa.

The way to go is to use the procedure for creating multiple table fields that is described in the *Creating Form Tables* section of the Acrobat on-line Help documentation, which will be paraphrased below. The procedure is to, first, individually create one complete row of fields in the table. Those fields are then selected by using *Shift-drag*, and then replicated by using *Ctrl-drag* to expand the first row to the rest of the table.

Unfortunately, you will still have to edit the replicated fields one at a time, for reasons to be explained, but this amounts to considerably less work than creating fields individually.

When you replicate the first row of data fields, Acrobat appends a unique row identifier to the end of each field's Name. This, in effect, creates unique names for

each field in the table. The way Acrobat does this is to separate the field's original Name with a period followed by the numerical value of the row. For example,

FirstName.0 ... LastName.0 ... Phone.0 ...

FirstName.1 ... LastName.1 ... Phone.1 ...

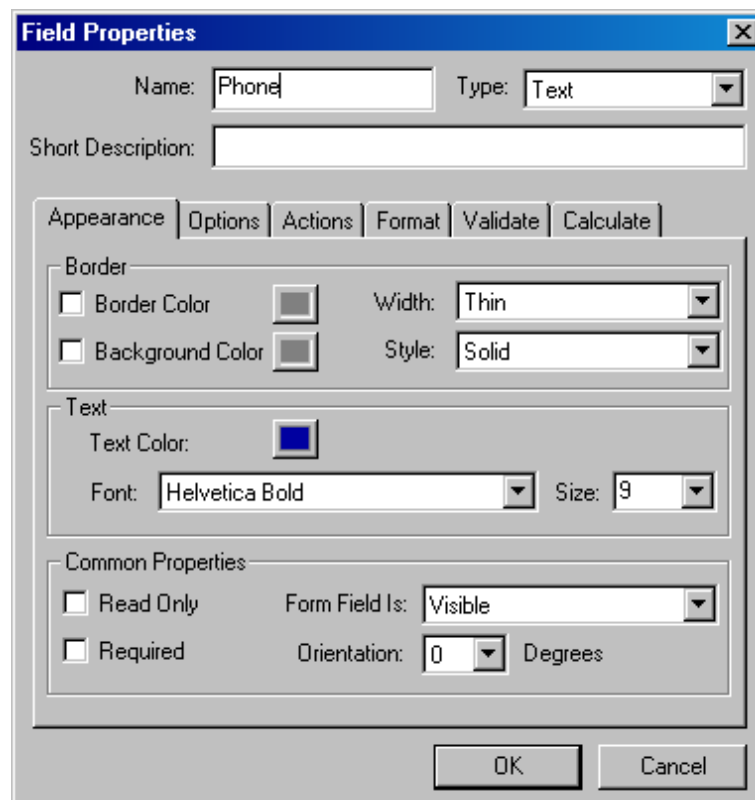
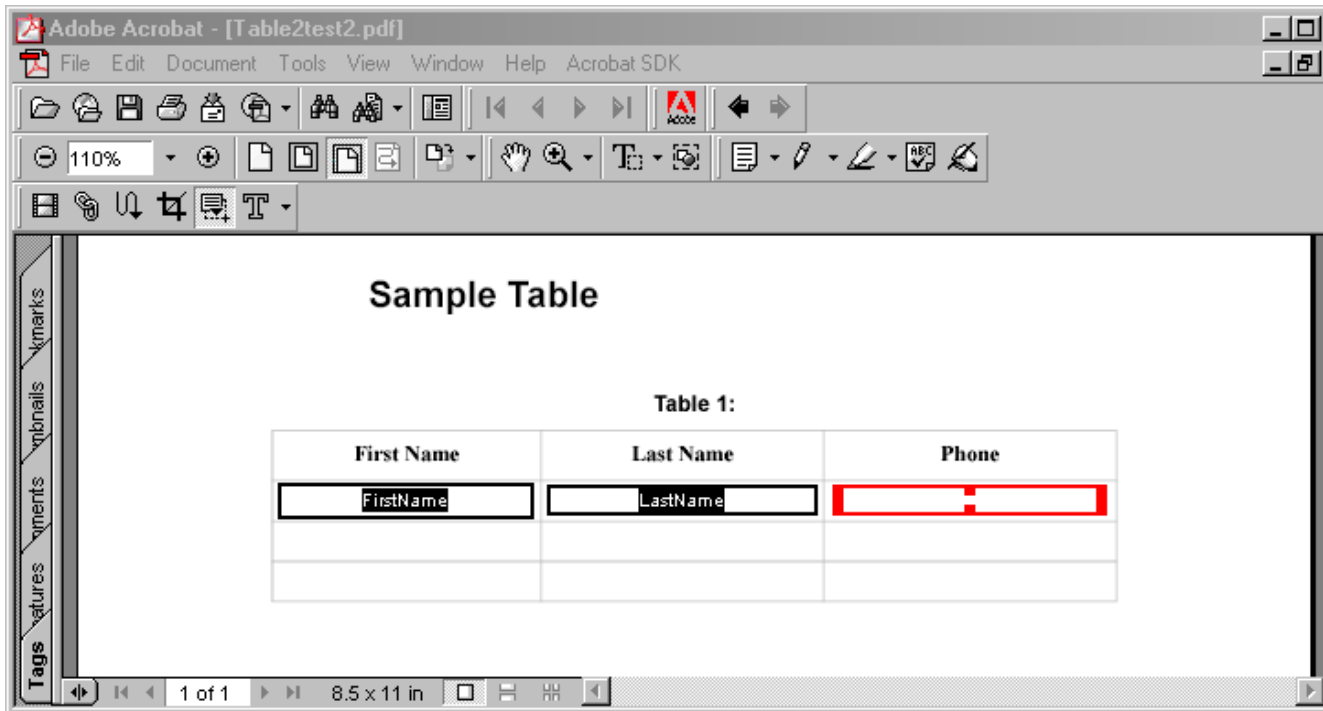
Unfortunately, when PDF Forms Access processes the table, it ignores the appended period and row number and analyzes the table as if all the data fields in each column were copies of one another.

This produces various undesirable effects, depending on the circumstances. When the form table is initialized in Forms Access, the Short descriptions for the Datum elements in each row are given the correct row numbers; in this case, for example, *row 1*; through *row 3*; However:

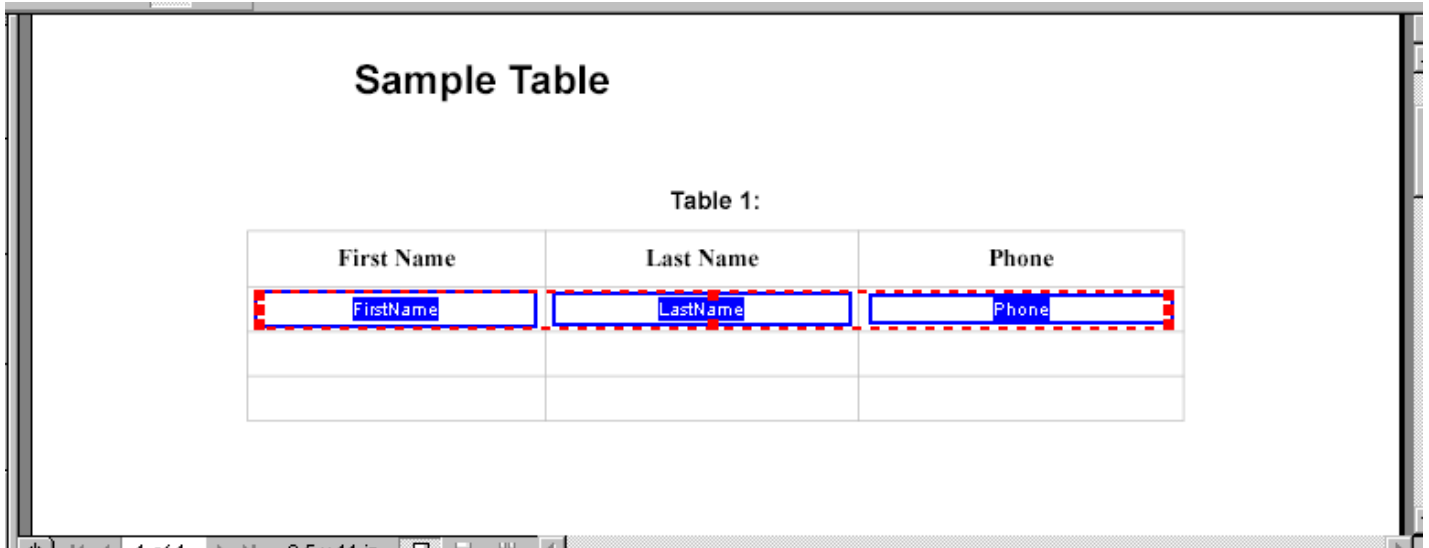
- If the initialized file is immediately Finished, none of Datum elements will have Short descriptions, and screen readers will report *No MSAA data* for each field.
- If the file is Saved or Saved As and then loaded back into PDF Forms Access, all the row numbers in all the Short descriptions will have changed to the last row number; in this case, *row 3*; When the form table is Finished and tested with a screen reader, the screen reader will announce every data field as being on "row 3;". Confusion laced with annoyance is always a deadly combination for users.

The suggested procedure is described below, for your information, but in the interest of time you will not go through the steps as an exercise.

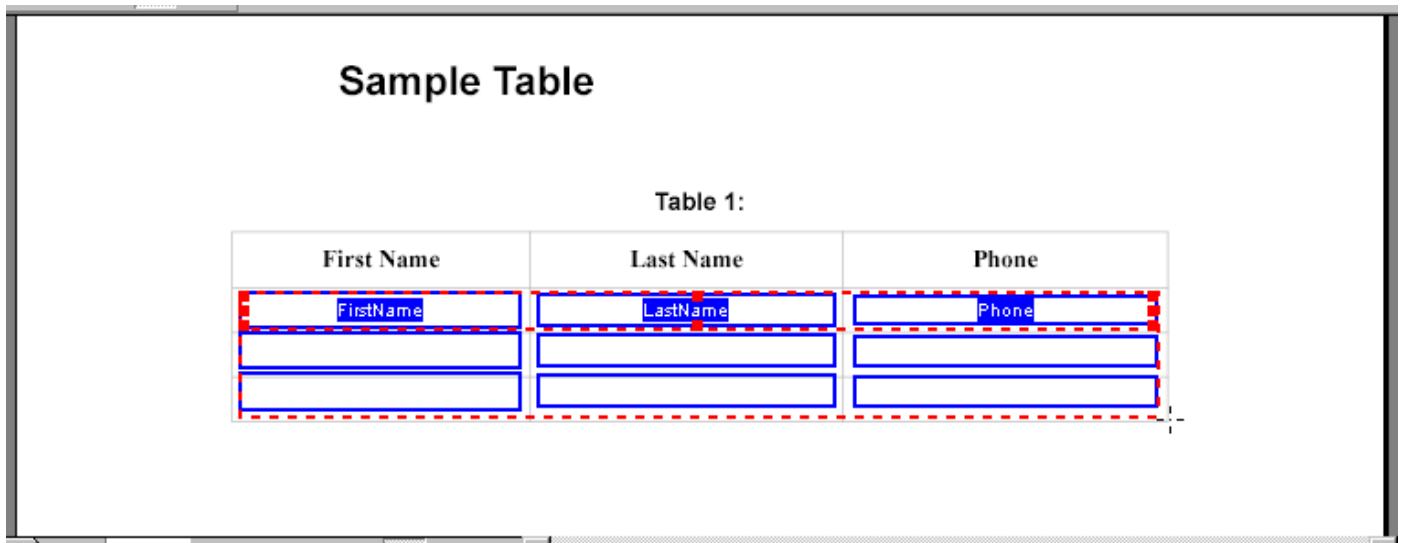
1. The two screen shots below show the first row of data fields in the table, and the Field Properties dialog for the Phone data field in row one. Each field has been created manually and given a unique name.



- Next, a box is drawn around the first row of data fields by holding down the Shift key and the left mouse button.



- While holding down the Control key and the left mouse button, the border around the first row of data fields is grabbed by a bottom or bottom corner handle and dragged to cover the entire table. When the mouse button is released, the fields will be replicated to the rest of the table, as in the screen shot below.



- When you click anywhere inside the group of data fields, the fields are given unique names using a period and the row number, as described earlier, and as shown in the screen shot below:

Sample Table

Table 1:

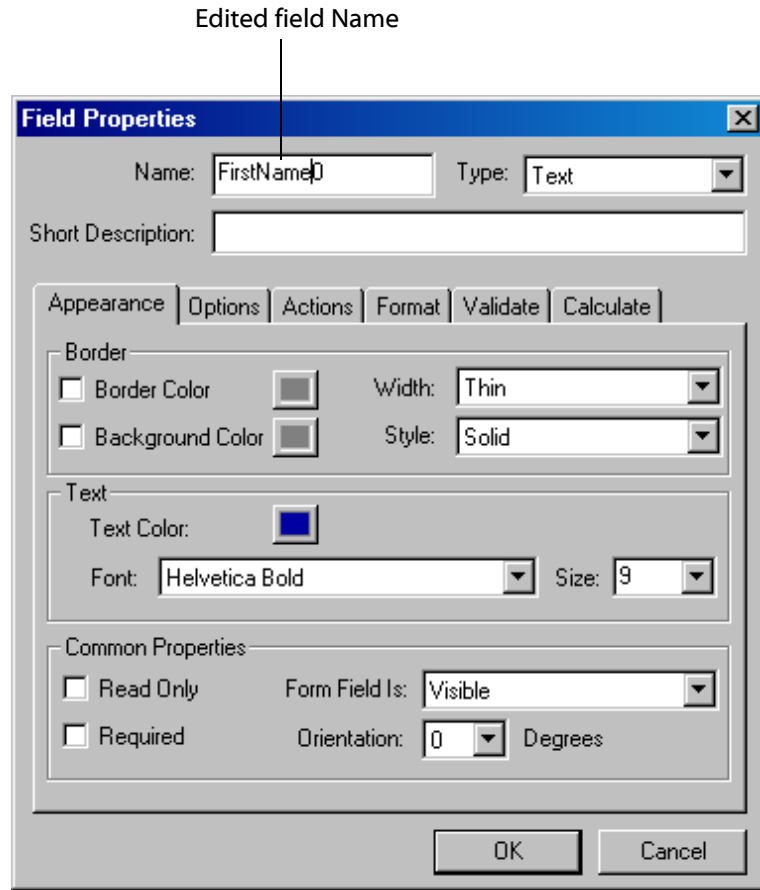
First Name	Last Name	Phone
FirstName.0	LastName.0	Phone.0
FirstName.1	LastName.1	Phone.1
FirstName.2	LastName.2	Phone.2

You may want to do some adjusting of the placements of the new fields to align them better with the table cells.

5. As described earlier, the `<dot><row number>` suffix that Acrobat has given to the field names to make them unique is going to cause problems for PDF Forms Access.

The easiest solution is to simply edit the field Names and remove the dot. This will create unique field Names that Forms Access will understand.

The procedure is to right click on each field, which will bring up the Field Properties dialog. In the Name edit field, delete the period in the name and click OK.



This is an admittedly tedious process, especially for large tables, but it is much easier than creating individual data fields for table cells, and generally results in a more accurate initial Structure Tree.

Importing Forms Structure



Importing Forms Structure and Concluding Topics

In this, the final module, you will see how the structure of a form can be preserved and reapplied to an updated or modified version of that form. A few miscellaneous topics, Stereotypes and Field Groups, will also be introduced. Up until now, you have been working with simplified forms in the exercises, but in the concluding exercise, you will have a chance to work on a serious, heavyweight form. A Summary section will wrap it all up.

Module Contents

Topics	Exercises
Importing Structures	Importing a Form Structure
Wrapping It Up and Putting It All To Use	The Final Test
PDF Forms Access Summary	

File to Download for Exercises in this Section

FormsAccessExamples.zip, which is normally found in the Exercises folder, contains the following files:

- **PersonalData.pdf**
- **PersonalData_underline.pdf**
- **PersonalData_unfinished.pdf**
- **ProblemForm1.pdf**
- **ProblemForm2.pdf**
- **ProblemForm2mod.pdf**
- **ProblemForm4.pdf**
- **Table1.pdf**
- **Table2.pdf**
- **Table3.pdf**
- **Table1_ext.pdf**
- **Table1_org.pdf**
- **Table1_org-new.pdf**
- **Jaws1.wav**
- **Jaws2.wav**
- **form_1040.pdf**
- **f1040_accessible.pdf**
- You should extract these files from the zip archive if they have not been extracted already.



Importing Structures

Imagine the following scenario: last year you painstakingly created a complex electronic form for a client. This year, the client needs miscellaneous revisions made to the form: some additional fields, a different heading, and some wording changes here and there. You have the original form document, which you created in, say, Adobe FrameMaker, so you return to the original document, add some rows to one of the form tables, and make the changes to the heading and other text areas as requested. You save the document as an untagged PDF file. To save time, you open both the original form and the new form in Acrobat, select the entire group of data fields that have not changed from the original form, and copy them to the new form (yes, you can do this). You then create new data fields for the additional cells in the table.

At this point, however, when you go to initialize the new untagged form document with PDF Forms Access, you will lose all the adjustments to the Structure Tree you had to make to the original form prior to Finishing it. Depending on the size and complexity of the form, this could represent a considerable amount of work.


Fortunately, PDF Forms Access provides a method for handling these kinds of form revisions, referred to as the **Import Structure** command. The procedure is to use the command to import the structure of a tagged form that was previously Finished with PDF Forms Access, and then to load an untagged form into Forms Access. The Structure Tree elements from the original form are applied where ever possible to the new form. Content in the new form that does not match that in the old form is highlighted in yellow in the View pane, and given an appropriate element tag which is placed at the bottom of the Structure Tree. The new elements can then be wrapped in parent elements if necessary, and moved to the appropriate place in the Structure Tree.



Exercise: Importing a Form Structure

In order to do any of the exercises in this module, you must have Acrobat 5 or higher and PDF Forms Access 1.0 or higher installed on your machine, and you must have copied the sample files in the archive **FormsAccessExamples.zip** to a folder on your machine.

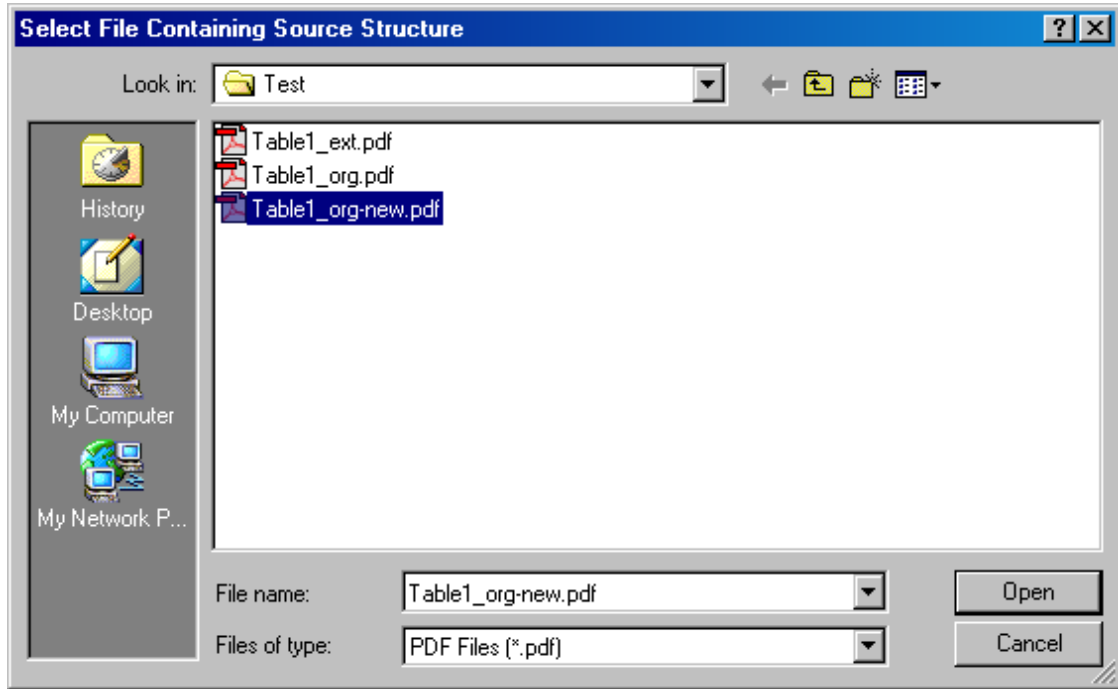
In the exercise that follows, you will import the structure of a Finished version of the form table you worked with earlier. You will then open a slightly expanded version of that file which contains a modified heading, an extra line containing a date, and two additional data field rows at the end of the table. You will observe how Forms Access imports what structure it can, and how it marks the new structure in the revised form. You will then modify the Structure Tree to make it consistent with the imported structure.

1. Close any files that may be open in PDF Forms Access.
2. Select **File > Import Structure** from the Menu bar or click the **Import Structure**  button in the Toolbar.

[Instructor Notes: Make sure steps 3 and 4 are understood. The first dialog calls for a file that has already been processed and Finished with Forms Access. The second dialog calls for an untagged form file -- presumably one that is some variation of the first file. If the second file has additional fields in it, they must be set up first in Acrobat before the import procedure.](#)

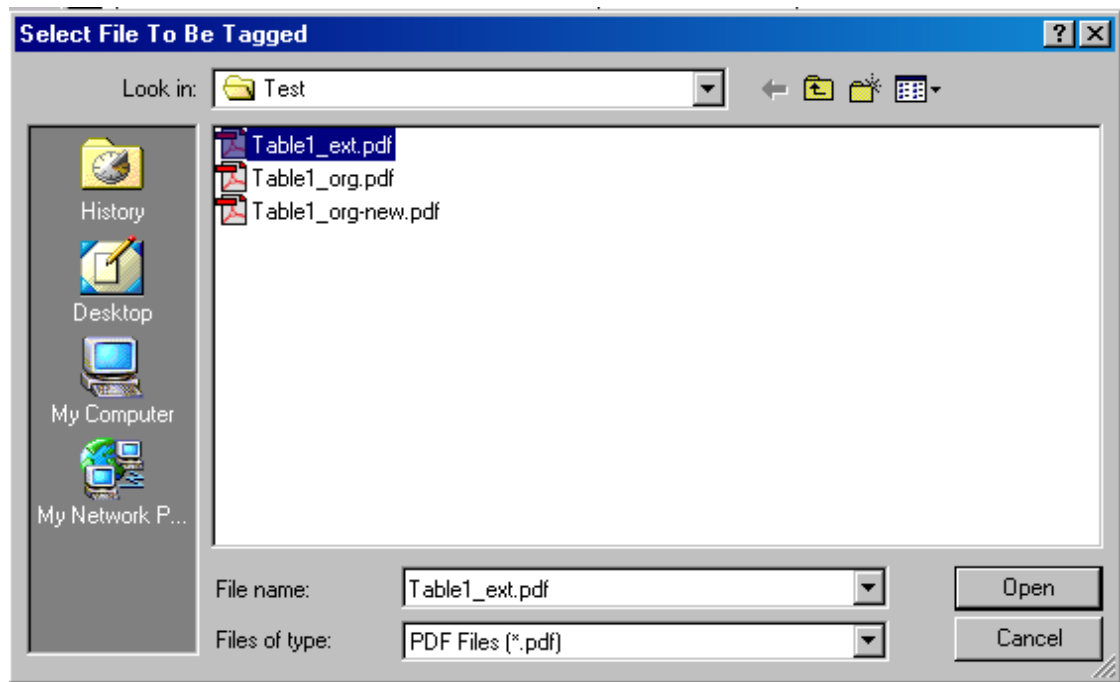
3. The first file selection dialog will appear. In this dialog, you select the file from which you are importing the structure. This *must* be a file that has been previously Finished and tagged in PDF Forms Access.

Select the file **Table1_org-new.pdf**.



4. A second file selection dialog will appear. In this dialog, you select the untagged file into which you will be importing the structure. This *must* be an untagged file, not previously initialized or Finished in Forms Access, with all the data fields in place. The import process tags data fields but does not create them.

Select the file `Table1_ext.pdf`.



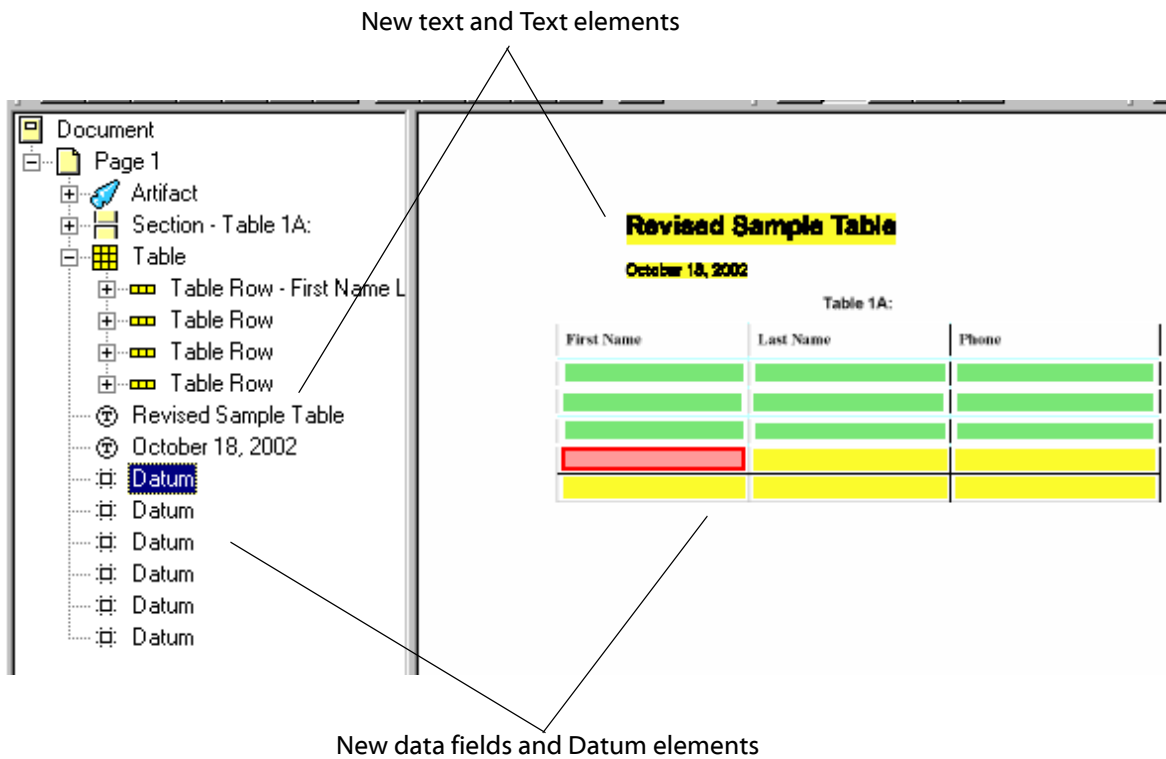
- PDF Forms Access will initialize `Table1_ext.pdf`, and apply the previously imported structure to parts of the form that have not changed from the original.

In the View pane, data fields that are unchanged from the original are highlighted in green. The structure from the original form has been imported for these fields. This can be seen in the Structure Tree under the Table element: if you expand it, you will see the element structure hierarchy from the imported form, complete with Short descriptions.

New content in the new form is highlighted in yellow in the View pane. You can see that the heading and the date and two rows of data fields are different in the new form.

The new elements are placed at the bottom of the Structure Tree. You can see that Forms Access placed the new text there, along with six Datum elements for the additional data fields in the table.

The next steps are to correct the sequential order of the two new Text elements, and wrap the Datum elements in a way that is consistent with the imported structure.



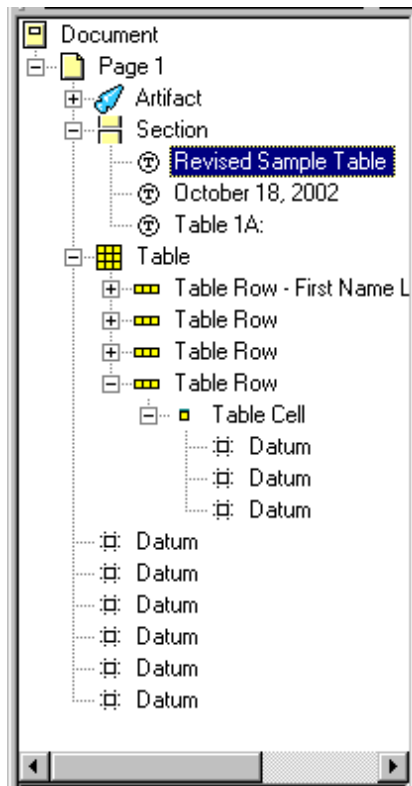
- Expand** the first **Section** tag in the Structure Tree.

Right now, it contains the Text for the table caption, *Table 1A:*. In the Structure Tree, the Text element for the date should be positioned just above that, and the Text element for the heading *Revised Sample Table*, should go just above the date's element.

- Click on each of the new Text elements, drag them up to the Section element, and drop them in the correct sequence as children of the Section element.

The screen shot below illustrates the move.

One of the imported Table Row structures has been expanded in the screen shot to illustrate the pattern that you will create for the new Datum elements in the next step.




You may recall that this is the form table for which Forms Access created Table Rows each with a single Table Cell that contained the three Datum elements for the row. The first Datum element in the row was given a Short Description consisting of the row number, all the column headings, and the text (*part 1*). The other two Datum elements in the row were given the Short descriptions (*part 2*) and (*part 3*), respectively.

Assuming you want to create the same pattern for the new Datum elements, you will need to wrap the Datum items in Table Row and Table Cell elements to match the imported structure.

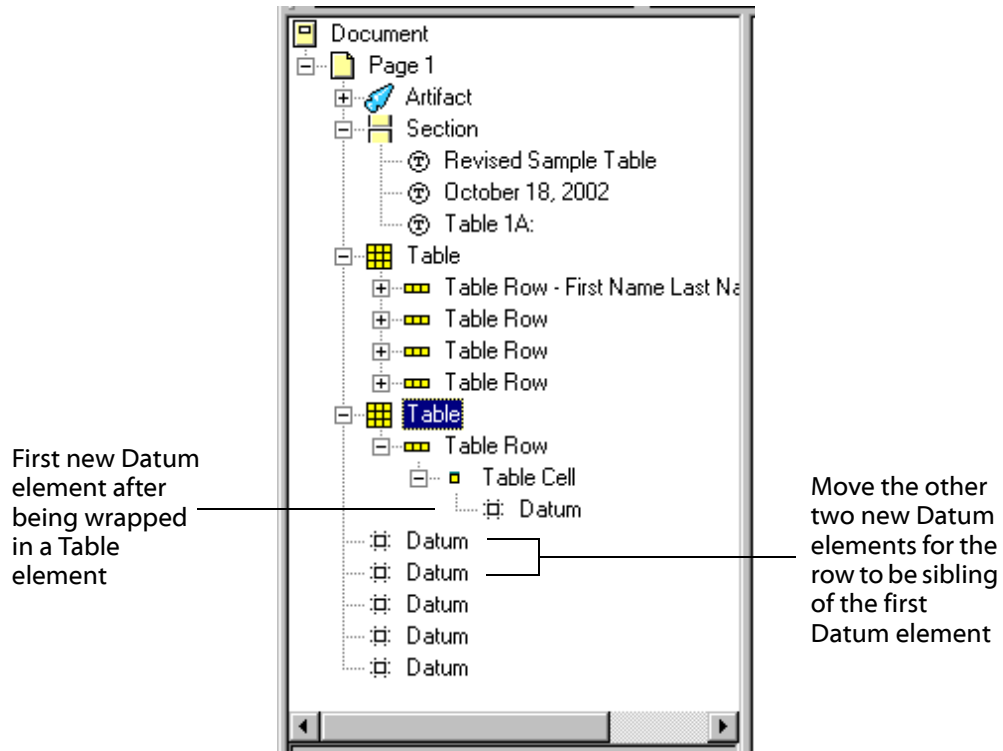
- Click** on the first new **Datum** element.

Open the **Create** menu in the Menu bar.

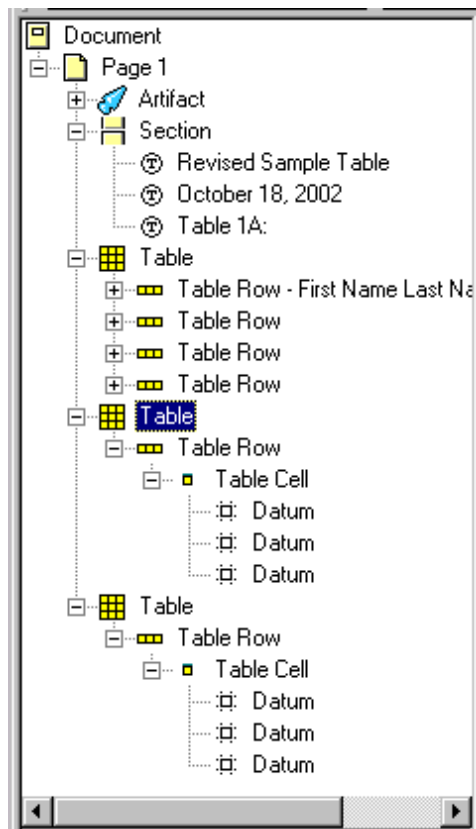
Notice that the only element that is available and relevant to the situation is the *Table* element. Likewise, the only active button in the Toolbar that has anything to do with tables is the *Create Table* button.

9. Select **Create > Table** from the Menu bar or click the **Create Table**  button in the Toolbar.

This will wrap the Datum element in the following Table element hierarchy:



10. Now, **select** each of the other two new **Datum** elements for that row and **drag** them into position under the first Datum element. Make sure you get them into the right order. They will now be siblings of the first Datum element and children of the Table Cell element that was added along with the Table element.
11. **Repeat Steps 8, 9, and 10** for the last three new **Datum** elements. The Structure Tree will now look like this:



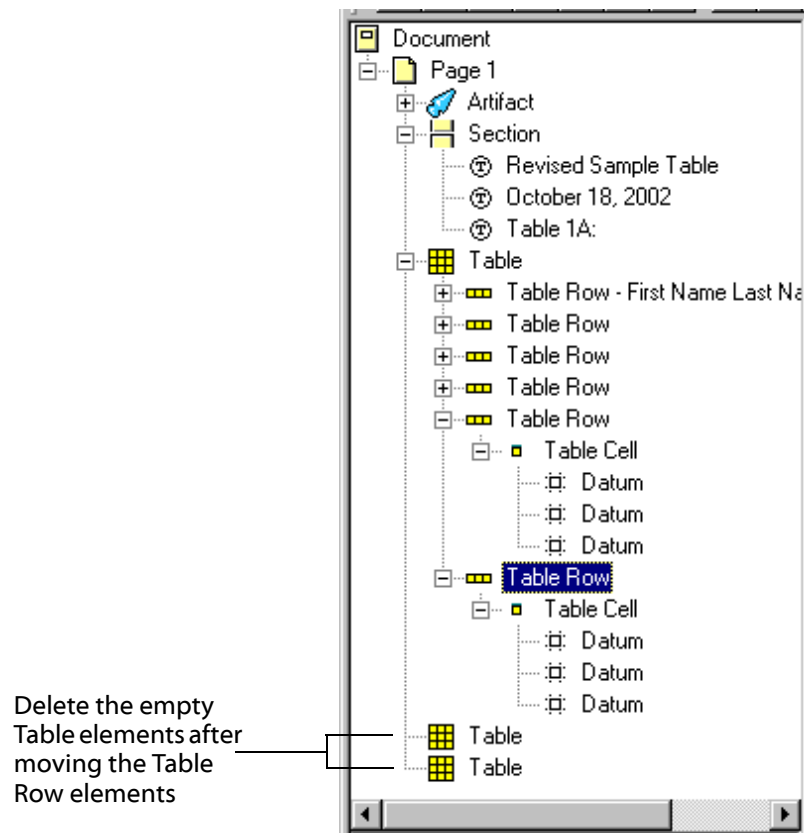
But, what about the two extra Table elements? They do not reflect the structure of the form, and when a screen reader encounters them, it will announce that it is about to read a three column, one row table.

The solution is to move the two new Table Row elements and their children to be children of the first Table, then delete the two new Table elements from the Structure Tree.

12. **Select** the first new **Table Row** element and drag it to a position under the last Table Row element, and as a child of the first **Table** element. You can collapse the new Table Row elements first to make the move operation a little easier.

Repeat the process for the second new **Table Row** element. Make sure you position them in the correct order.

The Structure Tree will look now like this:



13. Next, **select** each of the extra, now empty, **Table** elements at the end of the Structure Tree and press the **Delete** key to remove them.
14. Finally, you will need to **add Short descriptions** to the new **Datum** elements. Follow the pattern of the other Short descriptions for each row of data fields, which is:

Datum 1: <row number>; <all column headings> (*part 1*)

Datum 2: (*part 2*)

Datum 3: (*part 3*)

A relatively easy way to do this is as follows:

- **Click** on the first **Datum** item of one of the imported Table Cells -- in other words, click on a Datum item that has the Short description pattern of **Datum 1**, above.
- **Right click** on the **Short description** field in the **Properties** palette to highlight it, and select **Copy** from the pop up menu.
- **Click** on the first **Datum** element in one of the new Table Rows.

- **Right click** on the **Short description** field in the **Properties** palette and select **Paste** from the pop up menu.
- **Edit** the **Short description** to change the **row number** to reflect the correct row for that Datum element. Nothing else needs to be changed.
- **Click** on each of the new Datum element's **sibling Datum elements**, and add the text **(part 2)** and **(part 3)**, respectively, to their **Short description** fields.
- Repeat the above procedure for the second new row of Datum elements.

15. Select **File > Save As** from the Menu bar and specify that you want to replace **Table1_ext.pdf**.

Close Table1_ext.pdf.

[Instructor Notes: the course author found the following to be the case: you should verify this for yourself.](#)

16. **Open Table1_ext.pdf** again, then select **File > Finish** to complete the tagging process.

(PDF File Access will sometimes not Finish a Saved As form unless you reload it first.)

17. **Open** the Finished file and, if possible, test it with a screen reader.



Wrapping It Up and Putting It All To Use

[Instructor Notes: There is not enough material in Stereotypes and Field Groups to warrant a separate exercise for them alone at this point in the course, but the descriptions of them below are presented in steps, and students can follow the steps informally if they wish.](#)

The course will conclude with a brief introduction to two pieces of Forms Access functionality that you have not used yet: creating Stereotypes, and using Field Groups. These functions will be described, but will not be the subject of specific exercises. However you may find them useful in the exercise that does follow, *The Final Test*, in which you will tackle what could be described as a complex form.

Creating Element Stereotypes

You have already seen examples of element Stereotypes, or container elements, such as the Field and Table elements. A Stereotype is a parent element that has set of associated child elements which, when the Stereotype element is added to the Structure Tree, are added along with it.

You can create your own Stereotypes. You might find this useful if your form document is comprised of many repetitive sections, each containing the same pattern of elements. Suppose, for example, your document consists of multiple sections, each starting with an ID and a Name, followed by a block of text, followed by a data field and a form table. You can create a Stereotype for the Section element that will bring in child elements for *ID*, *Name*, *Paragraph*, *Field*, and *Table* each time you add it to the Structure Tree.

The following example describes the steps you would use to create the Stereotype described above. The example uses the equivalent of the file `PersonalData_2.pdf`.

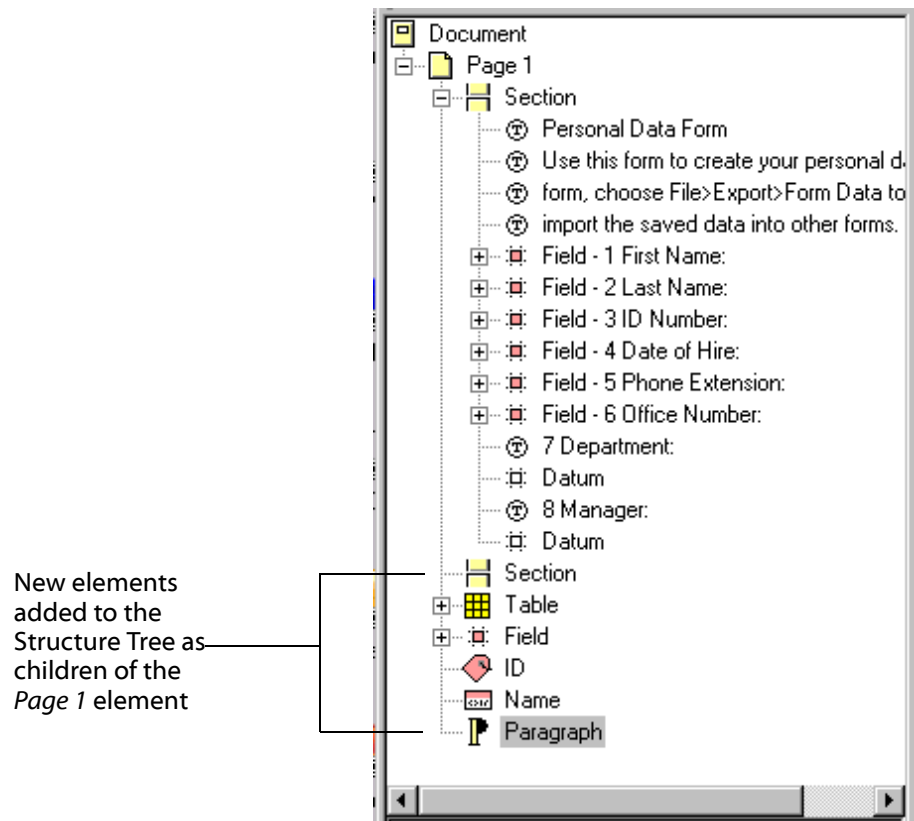
- The first step is to build the Stereotype out of elements in the Structure Tree. The elements may already exist, or you may add them to the tree for the express purpose of creating a Stereotype. In this example, they will be added to the Structure Tree.
- In the Structure Tree, collect the elements needed for the Stereotype. In this case, they are a *Section* element, an *ID* element, a *Name* element, a *Paragraph* element, a *Field* element, and a *Table* element.

The easiest way to do this is simply to create new elements as children of the topmost element for the page -- *Page 1*. The new elements will be added to the bottom of the Structure Tree.

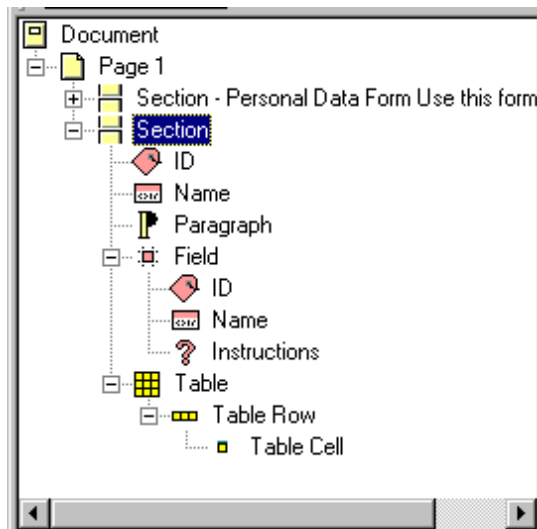
Select the *Page 1* element in the Structure Tree and use the Create menu or the buttons in the Toolbar to create, in any order, a new *Section*, *ID*, *Name*, *Paragraph*, *Field*, and *Table* element.

You *must* re-select the *Page 1* or *Document* elements before you add each new element because you want all the new elements to be children of the *Page 1* element. If you forget to do this, you will end up adding the new element as a nested child of the last element you added to the tree.

The screen shot below shows the new elements added to the end of the Structure Tree as children of Page 1.



- The next step is to arrange the elements in the order you want them as children under the Section element, which will then become their parent. The screen shot below, with the first Section element folded, illustrates this:



- The Section element is going to become a Stereotype, containing all the child elements you see in the screen shot. To complete the process, select the Section parent element, and then select **Tools > Set Element Stereotype**.

You will see some little marquees flying up and away from the Section element when you do this, signifying that the Section element has achieved Stereotype status.


- As a test, Delete the new Section Stereotype from the Structure Tree. Then, click on the *Page 1* element, and use the Create menu to add a new *Section* element. It should appear along with the same pattern of child elements you gave it when you Set the Stereotype.

You might find use for a Stereotype like the hypothetical one described above to organize a lengthy, repetitive form. Adding multiple Stereotype elements to the Structure Tree, one for each repetitive section of the form, would provide container structures into which you could then move the original initialized elements of the form.

From the Tools menu, you can also **Reset All Stereotypes** to their default settings, or **Clear All Stereotypes**, which removes all Stereotype settings.

Field Groups

Field Group elements represent another useful organizational tool. They can be used to wrap groups of related Fields, and can be used to hold information about the entire group. The procedure is simple:

- Select the first Field element that you want to group.
- Select **Create > Field Group** from the Menu bar or click the **Create Field Group**  button in the Toolbar.

This will wrap the Field element in a Field Group element. The Field Group element is a Stereotype that mirrors the Field Stereotype -- it has ID, Name, and Instructions child elements.

- Next, move the other Fields associated with the group into place as children of the Field Group element.
- Move any relevant headings, identifiers, and instructional Text elements from the original form into the ID, Name, and Instructions child elements of the Field Group element.
- Alternately, add accessible text to the Actual text or Alt text attributes of the ID, Name, and Instructions child elements of the Field Group element.

Coming Next: The Final Test

You may find a use for Field Groups in the next exercise, *The Final Test*. Do as much of the next exercise as you care to, or take it home. There are no instructions -- you are more or less on your own, using only your wits and the vast amount of knowledge you have now absorbed from this course! Good luck!



Exercise: The Final Test

In order to do any of the exercises in this module, you must have Acrobat 5 or higher and PDF Forms Access 1.0 or higher installed on your machine, and you must have copied the sample files in the archive **FormsAccessExamples.zip** to a folder on your machine.

[Instructor Notes: This is a completely open-ended and unstructured exercise. Offer advice if asked, otherwise let students go at it on their own. You can use the exercise to run out the clock if need be, or give it to the speedier ones in the class to keep them occupied. If you can't get to the exercise in the time allotted for the course, encourage students to look at it on their own later as a good example of a complex form.](#)

The form you are about to open -- **Internal Revenue Service Form 1040** -- should exercise just about every PDF Forms Access method and technique you have learned so far. In regard to Forms Access, Form 1040 could be considered the mother of all forms, since IRS forms were the original target of Adobe's development of PDF Forms Access.

1. **Open** the file `form_1040.pdf` in PDF Forms Access.

Initializing the sample 1040 form will produce a mixture of hits and misses in regard to element creation in the Structure Tree. Your task is to:

- Decide what the logical read order of the form should be and arrange the Structure Tree to reflect that order.
- Insure that any special instructions are in proper juxtaposition to their fields in the Structure Tree.
- Create consistency in the tree by wrapping Datum elements in parent Field or Table elements where appropriate, and using consistent patterns for accessible text for IDs and labels.
- Insure that all Datum elements have Short descriptions.
- Add any necessary accessible text to the Structure Tree element attributes to supplement or replace missing or confusing text in the visible form.
- Save your work-in-progress, and Finish the form once you have completed it, or have completed as much of it as you care to. Remember, there are two pages.

2. Open `form_1040-new.pdf` in Acrobat once you have Finished it in Forms Access, and examine the PDF Tags Palette. Use *Turn On Associated Content Highlighting* and check the read order of the tags in the Tags Palette.

If you have a chance, use a screen reader to test the form.

3. You can see how your solution compares with the professionals' by opening the sample file `f1040_accessible.pdf`. You can also find this file at the Internal Revenue Service's web site at http://www.irs.gov/pub/irs-pdf/f1040_accessible.pdf.

Although you cannot load the tagged PDF file back into Forms Access, you can open it in Acrobat, then examine the Tag structure or use a screen reader to see how closely it resembles yours.



PDF Forms Access Summary

Adobe PDF Forms Access automates the tagging process required to create accessible Adobe PDF forms, significantly decreasing the amount of time and effort that would be required to do so manually. PDF Forms Access provides a number of useful tools for editing and modifying the Structure Trees of initialized forms, thereby providing a good deal of control over the logical reading order of accessible form documents.

The Import Structure feature in PDF Forms Access allows a tag structure from a form document to be imported to an untagged form document, making it possible to preserve the PDF Tag structure of a form between successive updates without having to completely reconstruct the tag structure in the updated document.

Looking Back

Looking back on the course, here is a synopsis of where you have been so far, and what you have learned:

- You were introduced to the Forms Access Work Area, the View pane, Structure Tree, and Properties palette, and you initialized a form by opening it in Forms Access.
- You learned that the elements in the Structure Tree determine what will be spoken as accessible text by screen readers, and in what order it will be spoken. You learned how to move and change elements in the tree, and how to wrap existing elements in parent elements.
- You learned how to create and arrange accessible text for fields in the form. You learned various methods for adding accessible text to the child elements of a Field element.
- You learned how to save a work in progress and how to Finish a form.
- You were introduced to some basic methods for testing and verifying the form's accessibility.
- You were presented with some examples of how different form layouts can affect the way PDF Forms Access analyzes and process a form, along with some recommendations for avoiding problems.
- You learned how to create accessible form tables, and saw some examples of how different form table characteristics can affect the way Forms Access processes a table. You learned how to modify and recreate a table using the Recognize Table command.
- You learned how to import a tag structure from one form to another untagged form using the Import Structure command, and how to include new elements from the target form into the imported tag structure.
- Finally, you were introduced to creating Stereotype and using Field Groups, and had a chance to work on a real form.

Congratulations on a job well done!