
# InfoPath 2010 Enhanced Integration with SharePoint Server 2010 and Its Implications When Designing Forms for Applications

# InfoPath 2010 Enhanced Integration with SharePoint Server 2010 and Its Implications When Designing Forms for Applications

Ira Fuchs
Microsoft Corporation
April 2010

**Applies to: Microsoft Office InfoPath, InfoPath Forms Services in Microsoft SharePoint Server 2010**

**Summary**: This paper describes how to use InfoPath List forms, InfoPath document-based forms, and InfoPath Web Parts to create enterprise-level applications.

## Table of Contents



Microsoft



To comment on this paper or request more documentation on these features, contact itspdocs@microsoft.com.

# Introduction

Microsoft® Office InfoPath® has been part of the Microsoft Office System since version 2003. It is a tool for creating data entry and information gathering forms. This is a simplistic description of what InfoPath does because InfoPath has a number of leveraged capabilities that make it a compelling tool for building demanding enterprise-class applications. It can access and submit information to and from any number of data sources; it can readily apply sophisticated functional logic to the information captured in the form as well as to the behavior of the form itself; and it can accommodate complex information structures handily.

Browser-based support was introduced with the Enterprise version of Microsoft Office SharePoint® Server 2007. With InfoPath Forms Services, a form template could be published to a SharePoint form library and the form could be rendered in a browser. Microsoft Office InfoPath 2010 has been further integrated with Microsoft SharePoint Server 2010 by providing the ability to create forms for SharePoint lists and workflows. The default ASP forms that SharePoint Server generates for lists and workflows can now be replaced with InfoPath forms that take advantage of the powerful features in InfoPath. The automatically generated ASP pages used for creating and editing items in lists, as well as workflow forms, have limited functionality and cannot be enhanced or edited easily. The option to replace these ASP forms with InfoPath forms provides SharePoint lists and workflows with significantly improved information handling and form behavior functions. With InfoPath as the underlying form generation tool, list and workflow forms can be flexibly designed and visually enhanced without writing code.

In addition, SharePoint Server 2010 introduces InfoPath Web Parts. You can simply insert one or more InfoPath browser forms on any SharePoint page as Web Parts and interactively connect them together. This new Web Part feature provides numerous options for creating composite applications in SharePoint Server 2010. With these enhanced InfoPath capabilities baked into SharePoint Server 2010, InfoPath plays a significant role in making SharePoint Server 2010 a robust and highly effective application development platform.

Developers have two design choices available to them when creating applications that will incorporate InfoPath forms. They can use a list form, which stores the information captured in the form directly in a SharePoint list, or a document-based InfoPath form that stores the information captured in the form as a structured XML file. These document-based forms are published and stored in SharePoint form libraries. Both types of forms can be used as InfoPath Web Parts. There are some important functional differences between document-based InfoPath forms and InfoPath list forms. We will examine what these differences are and discuss the considerations for determining when to use each type of form as well as some best practices for doing so.

# InfoPath List Forms

With InfoPath as the underlying SharePoint Server form creation tool, the forms for list and workflow items can be flexibly designed, visually enhanced, and implement sophisticated information handling and form behavior using InfoPath validation, conditional formatting and auto-population capabilities. Essentially, an InfoPath list form is a Web Part page containing an InfoPath Web Part. The form controls are generated from

the list columns. This new capability significantly enhances SharePoint Server as an application development platform.

Illustration 1 below shows the InfoPath list form for creating a new issue-tracking item. When a new list item was created in Office SharePoint Server 2007, the user was brought to a separate page displaying the ASP form. In SharePoint Server 2010, when the user creates a new instance of a list item, the form is opened in the context of the list itself, providing a more intuitive user-interface experience. The same list form can be opened using a URL link from anywhere else as shown in Illustration 2. Notice that when the form is first opened only two fields are displayed, Category and Area. The list values are look-ups from two secondary data sources that happen to be SharePoint lists and are presented through drop down list controls. The values presented in the Area control are filtered based on the value picked in the Category list. Once both values are present, a set of rules are triggered that display the rest of the form, as shown in the second screen capture, and automatically set the Priority for the issue being reported. All of this list form functionality is made possible by InfoPath.

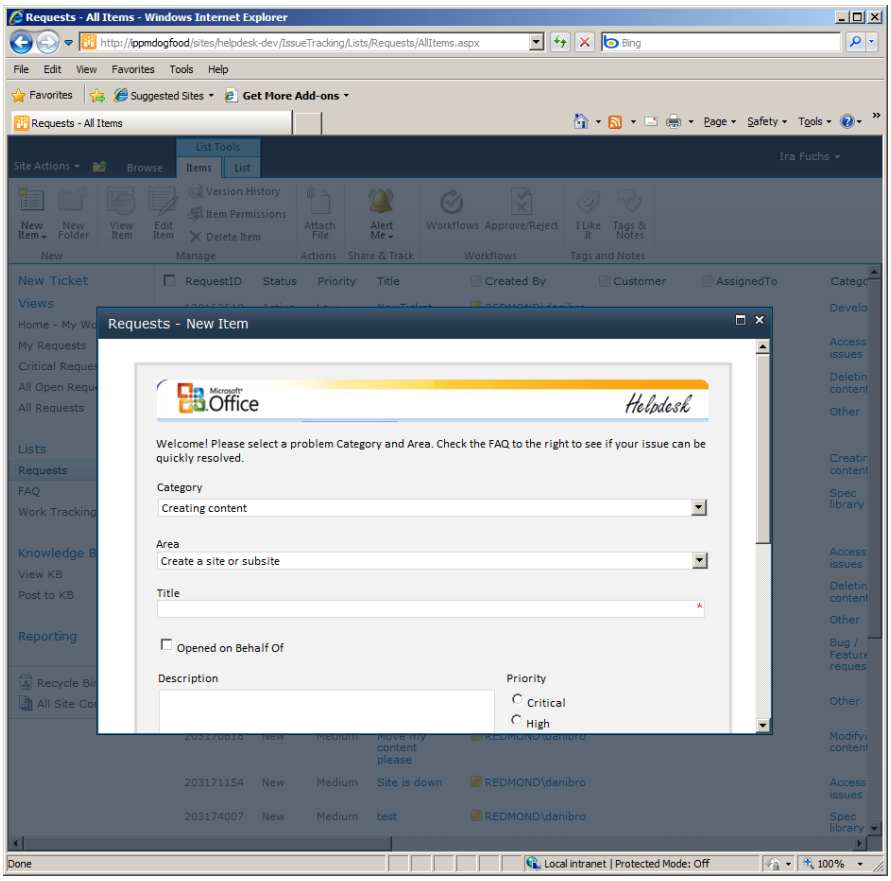



**Illustration 1 – The InfoPath list form for creating a new issue-tracking item displayed in the context of the list.**

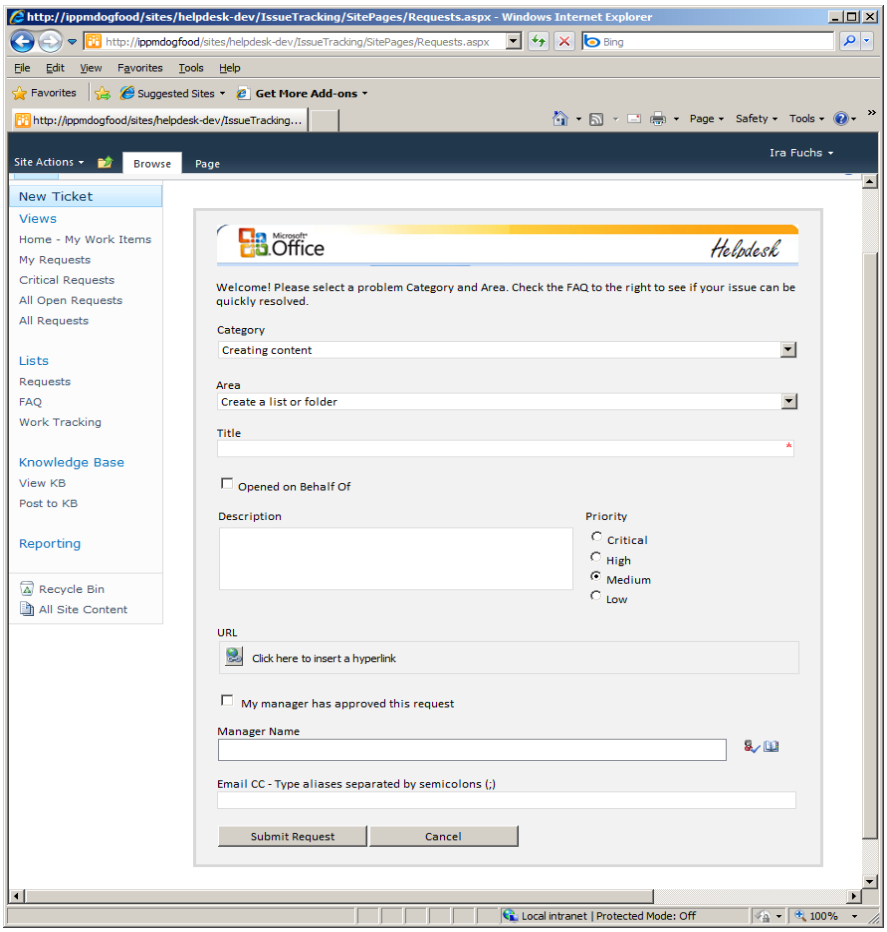



**Illustration 2 – The same InfoPath list form above opened from a URL placed anywhere.**

A list form can have multiple views, use rule set logic for validation, formatting and actions, and execute complex data access and information manipulation functions. This is all accomplished without writing any procedural code, which would be required to implement the same functionality in an ASP list form. For example, right-clicking on any field item in design mode will allow you to bring up its properties dialog box, as shown in Illustration 3 below.

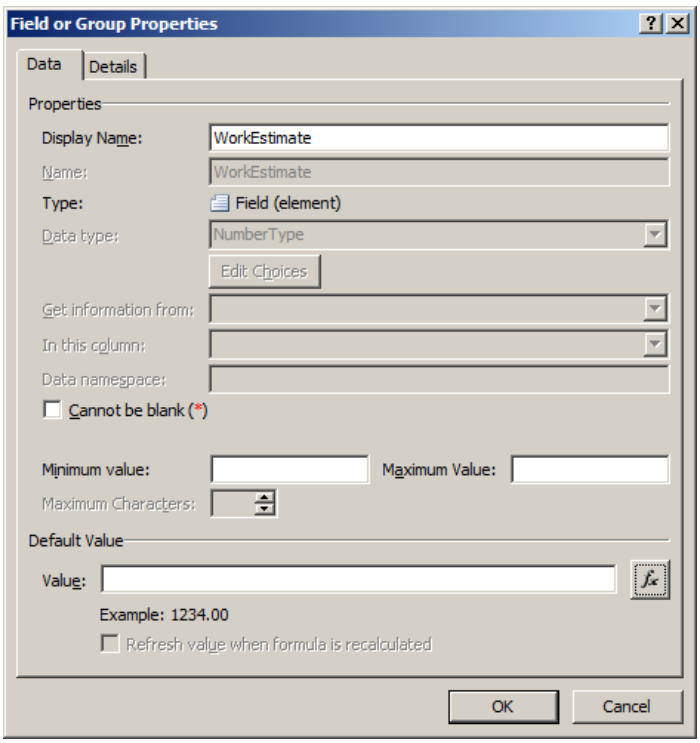
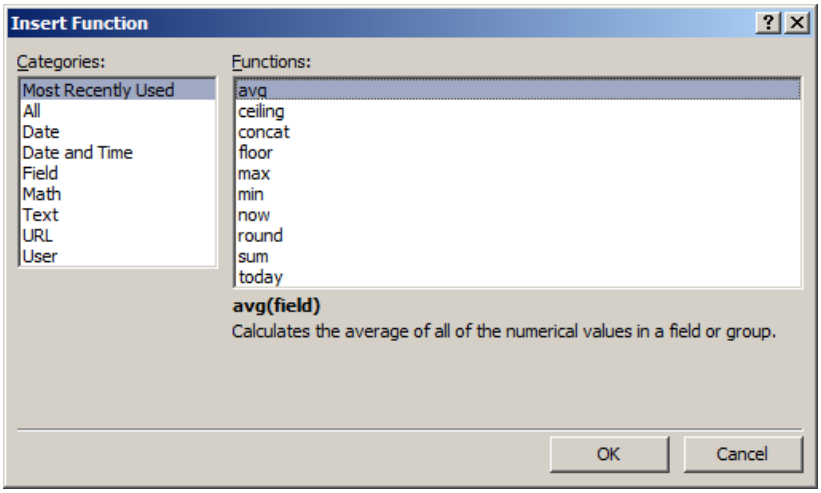



**Illustration 3 – The Properties dialog box for a field item in a list form.**

You can click the formula button next to Value to set a value for the field that can be generated by a function, as shown in Illustration 4, or obtained from other data sources, as shown in Illustration 5.



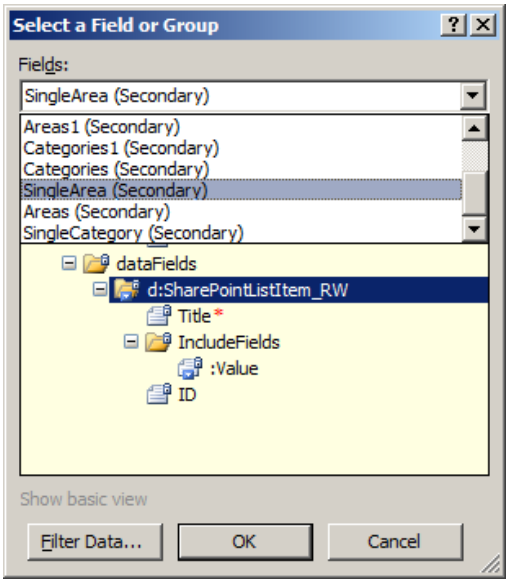




**Illustration 4 (left) – Setting the value for a field item in a list form using a function or formula.**
**Illustration 5 (right) – Setting the value for a field item in a list form by referencing a value in a secondary data source.**

Form fields can also be auto-populated using InfoPath rule actions. InfoPath rules provide a powerful way to populate and manipulate values in the form and control the behavior of the form itself. Rules are used to

validate values entered into controls; apply conditional formatting to controls or form sections; and trigger the actions to “Set a field’s value”, “Query for data”, “Submit data” and “Send data to a Web Part”. Illustration 6 below shows a simple rule that hides the Title field control when there is no value present in the Area field control. This condition only occurs when a new instance of the form is opened, resulting in the initial display of just the Category and Area field controls as shown in Illustration 1 above. Multiple rules can be applied to a control as well as to the load and submit events of the form. In addition, a rule can have multiple conditions and actions. This declarative way to programmatically manipulate information in a form and the form behavior itself is one of the most valuable and leveraged capabilities in InfoPath.

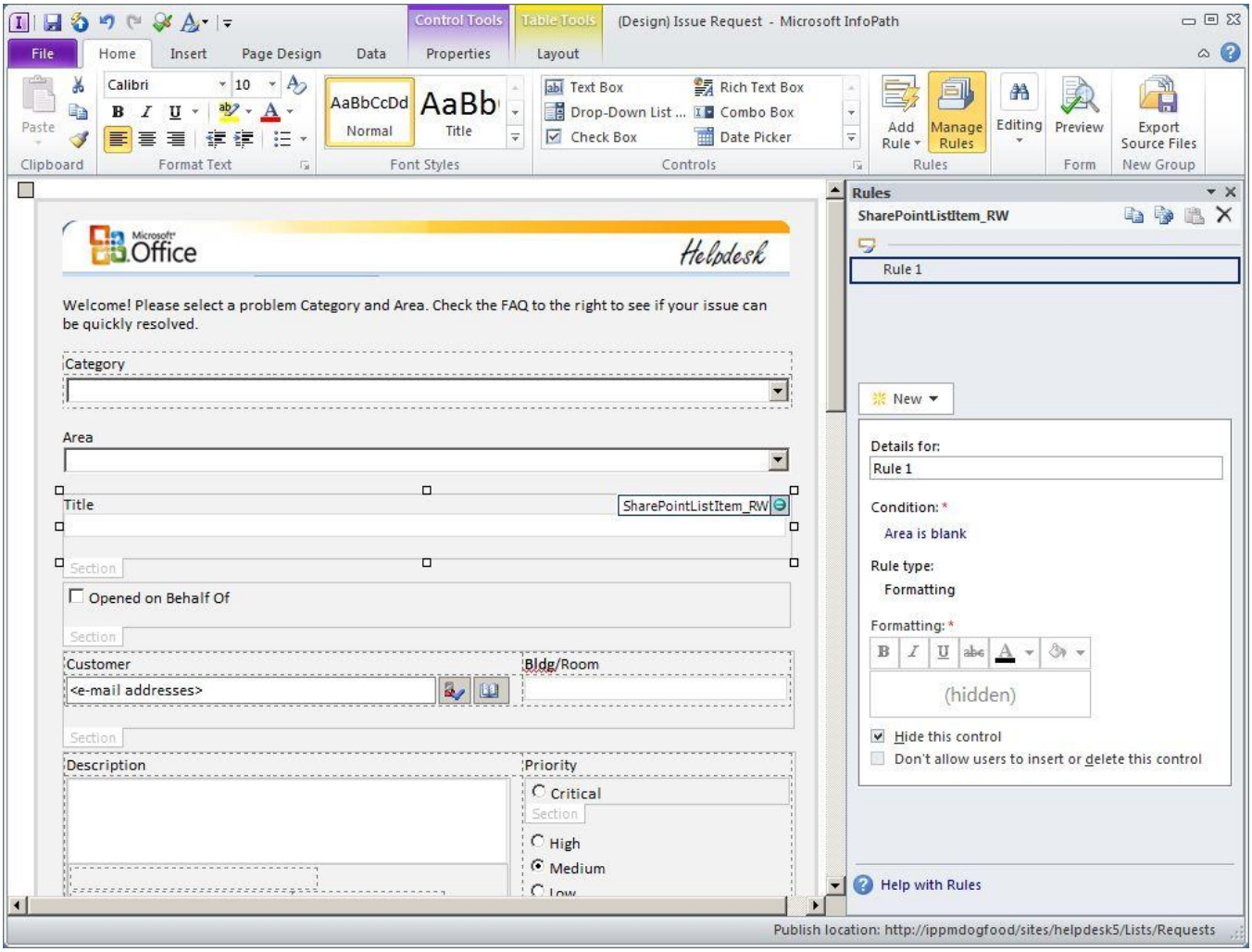



**Illustration 6 – Applying a conditional formatting rule to hide one field control based on a value in another field control.**

In addition to adding columns to a list through the SharePoint Server 2010 user interface or SharePoint Designer, new list columns can also be added directly from InfoPath in design mode in two ways. The first way is by dragging a form control from the design surface and setting its properties using the control properties contextual tab on the ribbon as shown in Illustration 7. The appropriate data types for that control will be displayed as shown in the properties dialog box as shown in Illustration 8.

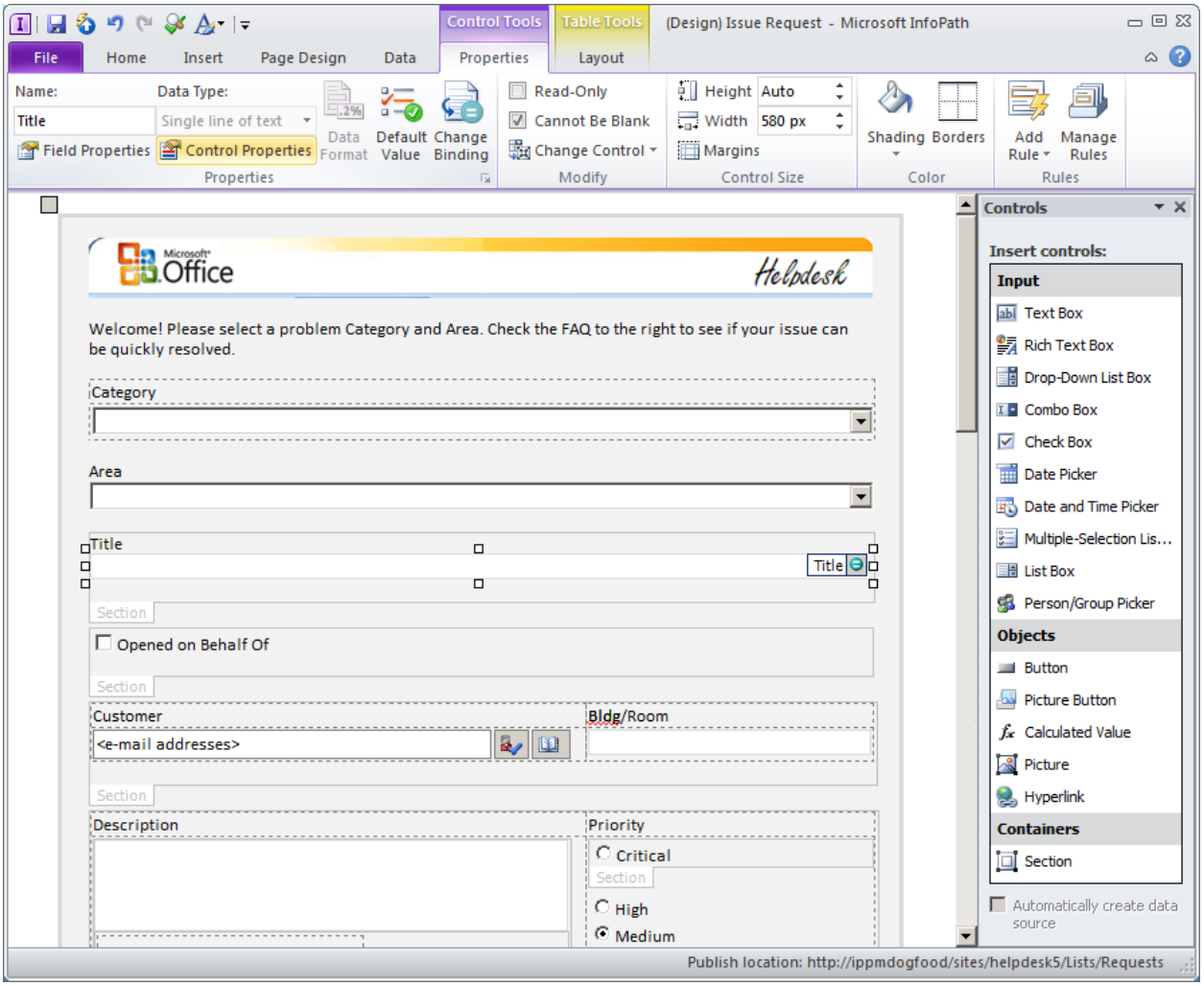



**Illustration 7 – Adding a column to a list from InfoPath by placing a control on the form surface and accessing its properties from the ribbon.**

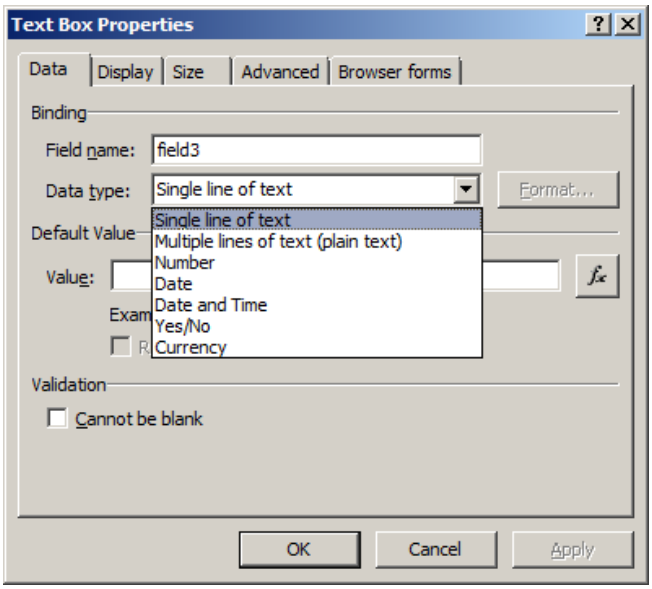



**Illustration 8 – The properties dialog box for a new field control that will generate a corresponding SharePoint list column.**

The form functionality described above is just a small indication of the capabilities of InfoPath. InfoPath provides a built-in library of form controls such as entry boxes for rich text, drop-down lists, scrollable list boxes, date pickers, check boxes, radio buttons, repeating tables, buttons, and numerous other controls, all of which can be customized through property attributes. InfoPath also supports repeating sections, optional sections, dynamic conditional formatting and visibility, custom dialog boxes, and multiple views of forms. Rule sets can be bound to the XML nodes in the underlying schema, to controls and sections in the form, as well as to the form itself, facilitating complex information handling and form behavior such as:

- Multiple form views
- Constraining and validating the information that can be entered in form controls
- Auto-population of fields
- Conditional display of controls, sections and views
- Generating automatic, derived and computed values
- Invoking events, prompts, and instructions
- Multi-conditional information filtering
- Accessing multiple data sources
- Multi-conditional form submit options and destinations
- Conditional opening and closing form behavior

It should be apparent that an InfoPath form is much more than just a data-entry tool. InfoPath creates a highly versatile smart browser form for aggregating, manipulating, presenting and generating structured information that can subsequently be submitted to other applications for consumption.

# Document-based InfoPath Forms

The primary consideration for using a document-based InfoPath form is the complexity of the information set required by an application.

Consider the information set below as represented by the XML Schema of an InfoPath form for a New Initiative Approval application. Every organization requires a process for introducing and evaluating new initiatives and bringing those that best advance the goals of the organization to fruition as sanctioned projects. This process is often quite lengthy and complex as any initiative can impact numerous entities within the organization; and it requires the input, collaboration and buy-in of many people with different responsibilities and obligations. An application that supports such a process must formally manage the assembly and review of all pertinent documentation, the orchestrated and timely input of all necessary stakeholders, and the accounting and evaluation of all factors that will lead to an informed decision.

Illustration 9 on the left shows the top-level group nodes for the New Initiative information set. A separate form view corresponding to each top-level node group (General Information, Impact Section, Budgeting and ROI Section) is used to assemble and display the required information, and only the appropriate people who contribute, review and/or approve the information in these form views can see them based on role and identity information. Note the nested hierarchical structure of the information found in the exploded Organization Section. Illustration 10 on the right shows the exploded view of the Budgeting and ROI schema section.

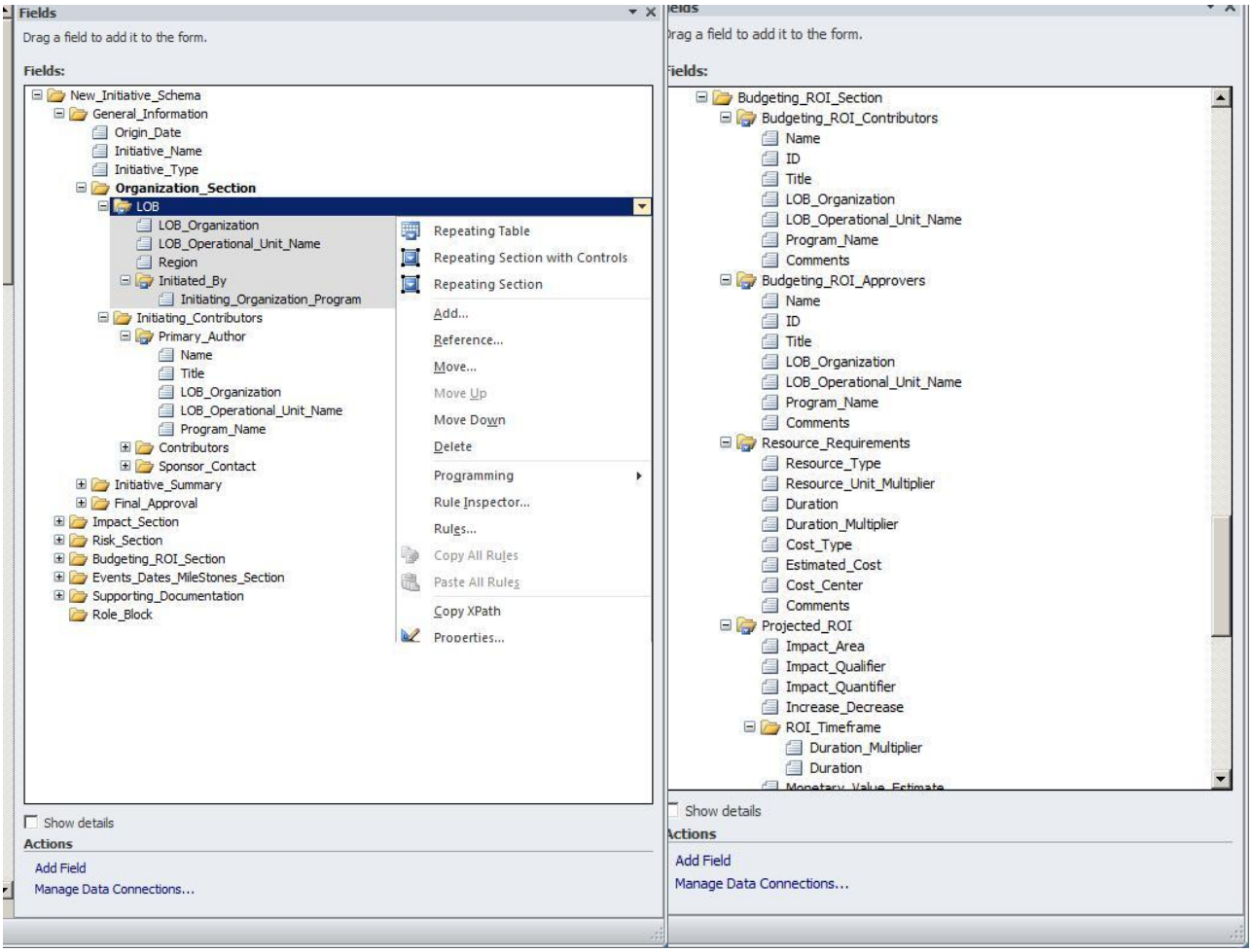



**Illustration 9 (left) – The top-level group nodes for the New Initiative information set.**
**Illustration 10 (right) – The exploded view of the Budgeting and ROI schema section.**

Within the Organization Section is a node group (LOB) that contains the information that defines the Line-of-business entities (LOBs) where the initiative is being originated. An initiative might be originated by a single LOB or it could be a joint venture of multiple LOBs. Any given LOB typically has multiple operational/business units within it, and the initiative may be an endeavor taking place among one or many units within different organizations and levels. In addition, the initiative may only be applicable to a specific geographic region that a LOB organizational unit operates in. Furthermore, the organization sponsoring the initiative might be a separate organizational unit entirely from the unit where the initiative is planned to be implemented. For example the Legal and Compliance organization, which is a corporate organizational unit, might be the sponsor for the deployment of software for email retention and analysis, which would be deployed in only those LOB units in regions that are subject to regulatory oversight where such measures are necessary.

Complex information models that incorporate both repeating one-to-many, and many-to-many relationships such as these can be challenging to implement in any application, even when procedural code is used to build this functionality. However, this type of complex information modeling is a strong suit of InfoPath because of its incorporation of XML Schema. In the highlighted LOB schema segment shown in Illustration 11 below, the

LOB fields, including the Initiated_By repeating group, are embedded in a repeating group, as denoted by the LOB repeating item folder icon.

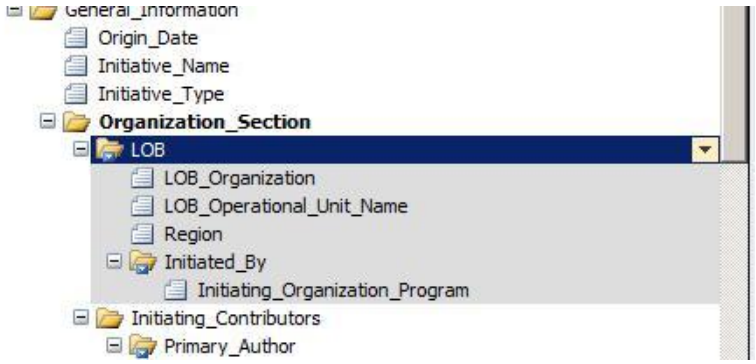



**Illustration 11 – The segment of schema showing structure of embedded repeating groups.**

When creating a document-based InfoPath form, InfoPath will automatically build a schema when controls are placed on the form, but this will result in an ad-hoc, unstructured schema. Alternatively, the schema for the information set can be defined and built directly in InfoPath before controls are laid out in the form. A third option is to create the schema in another XML editing tool and import it into InfoPath as the Main Data Source. Creating the schema within InfoPath first is the best practice for the following two reasons:

- One, InfoPath provides an easy way to build a schema for an information set. Illustration 12 (below left) displays the InfoPath user interface for building a schema by simply adding fields and groups. A field or group can be inserted anywhere in the schema tree and moved later if required, as well as renamed and referenced. Right-clicking on any field or group will display a Properties dialog box, as shown in Illustration 13 below right, where data types and default values can be specified using fixed values, variables, formulas or filtered look-ups. In addition, rule-sets can be created and applied to the schema fields and groups. By binding these properties and rule sets to schema objects themselves, any form control that is bound to the schema object will inherit this logic. It should be noted that a form schema can be modified at any time, even after the form has been deployed, and it will not break the form.

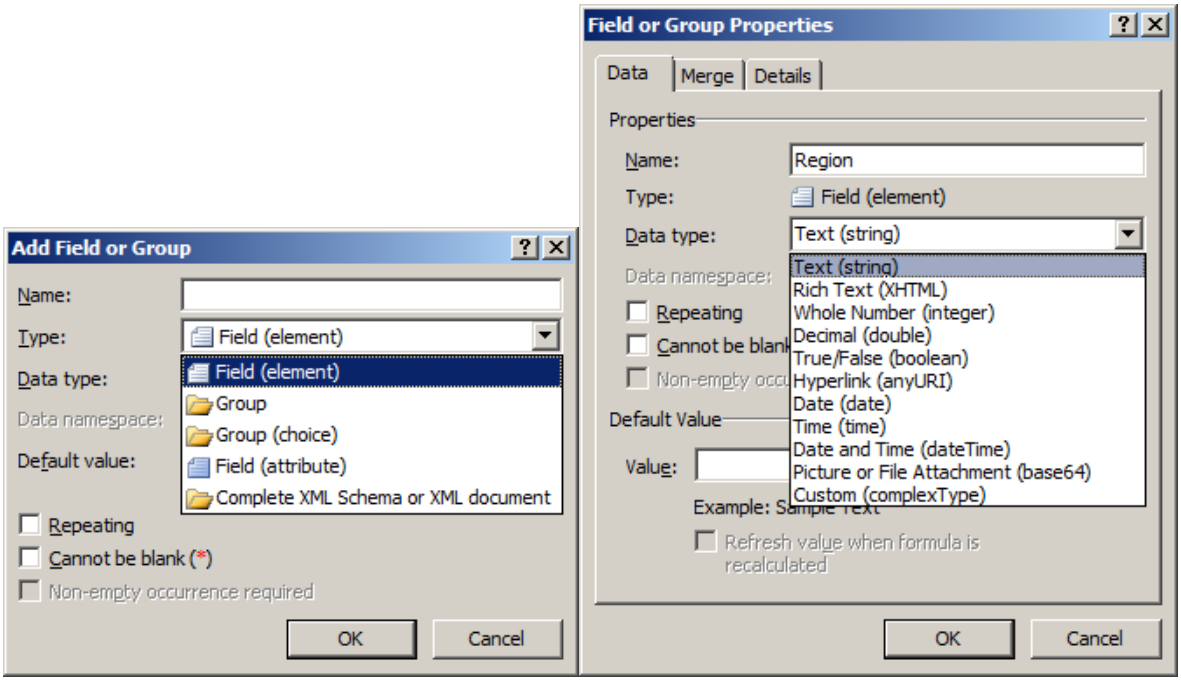



**Illustration 12 (left) – The InfoPath user interface for building a schema by simply adding fields and groups.**
**Illustration 13 (right) – The Properties dialog box where behavioral attributes, data types, and values can be set.**

- Two, by first designing and implementing the structural and behavioral requirements of the information set within the schema, the subsequent effort of designing the form and laying out the controls will be much more efficient and the resultant form will be functionally optimized. Illustration 14 shows the runtime view of a form segment displaying a repeating table that InfoPath automatically generated for the schema nodes in the LOB repeating group shown above. InfoPath interprets the intent of the schema design and provides control and layout options that implement the declared behavior. Note how the Initiated_By group is represented as a single column-repeating table embedded in a column of the LOB repeating table. Creating the same control layout manually would be very labor intensive. Furthermore, the entire table can be moved and placed anywhere in the form. Better still, this section can be saved and reused in any form. It is essentially a composite form object incorporating sophisticated logic that can be reused at any time. Designing the form directly from the schema structure leverages the combined capabilities of XML Schema and the rich library of form controls in InfoPath.

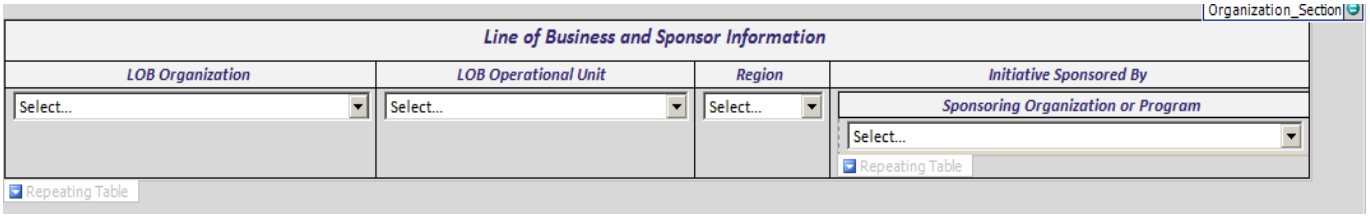



**Illustration 14 – The runtime view of a form segment displaying an automatically generated repeating table containing an embedded repeating table.**

A defining feature of a document-based InfoPath form is that it represents information captured in the form as XML tagged data. The organization and structure of this XML is based on XML Schema, the specification for defining structured XML metadata. One of the benefits of generating XML tagged information that

conforms to a schema is that any XML Schema enabled application can read and process the XML tagged information. This is the same mechanism that makes Web Services work and it is how InfoPath represents information and communicates with external data sources and other applications.

A document-based InfoPath form generates an XML file containing the populated values of the form, as can be seen Illustration 15 below.

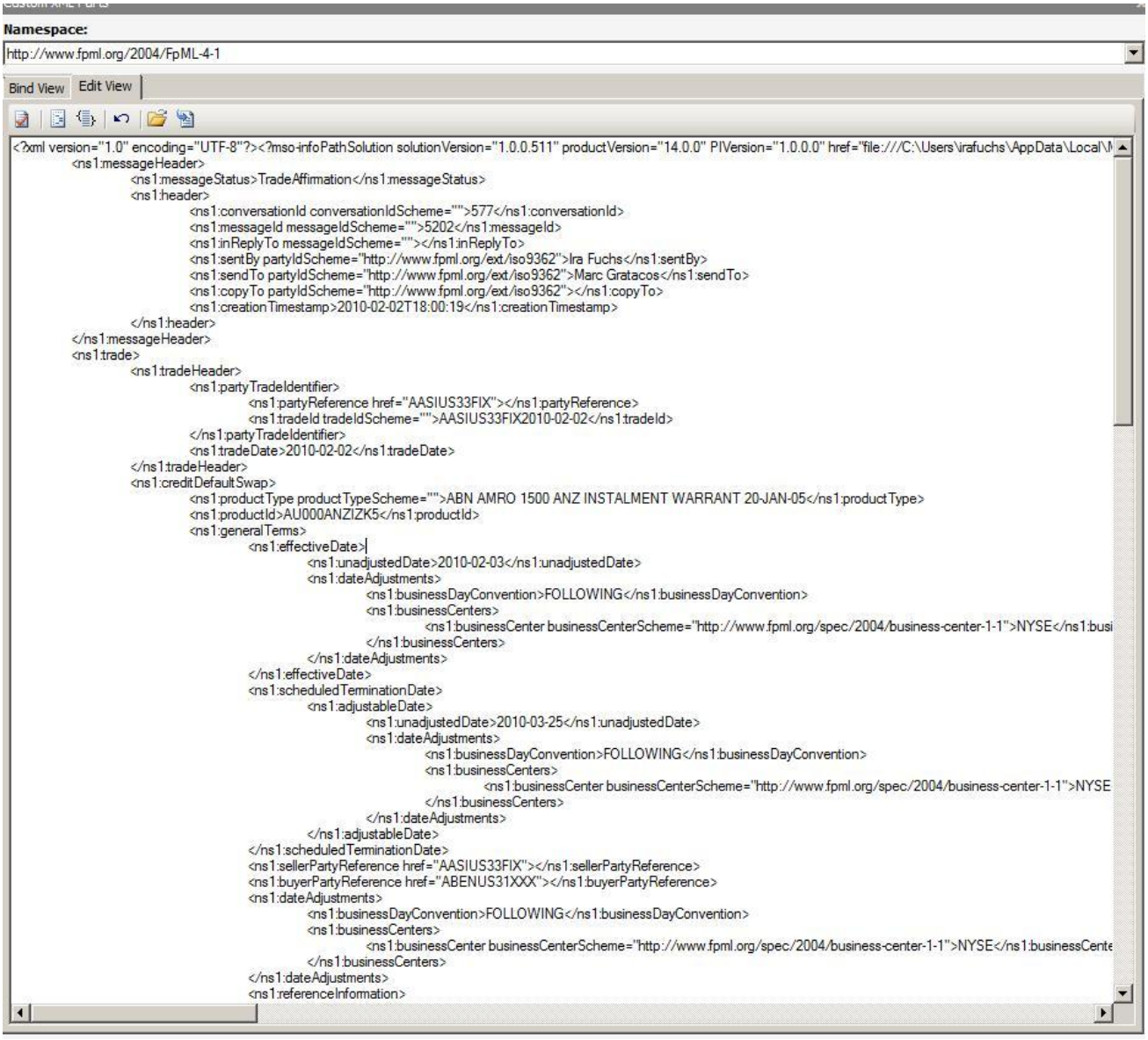
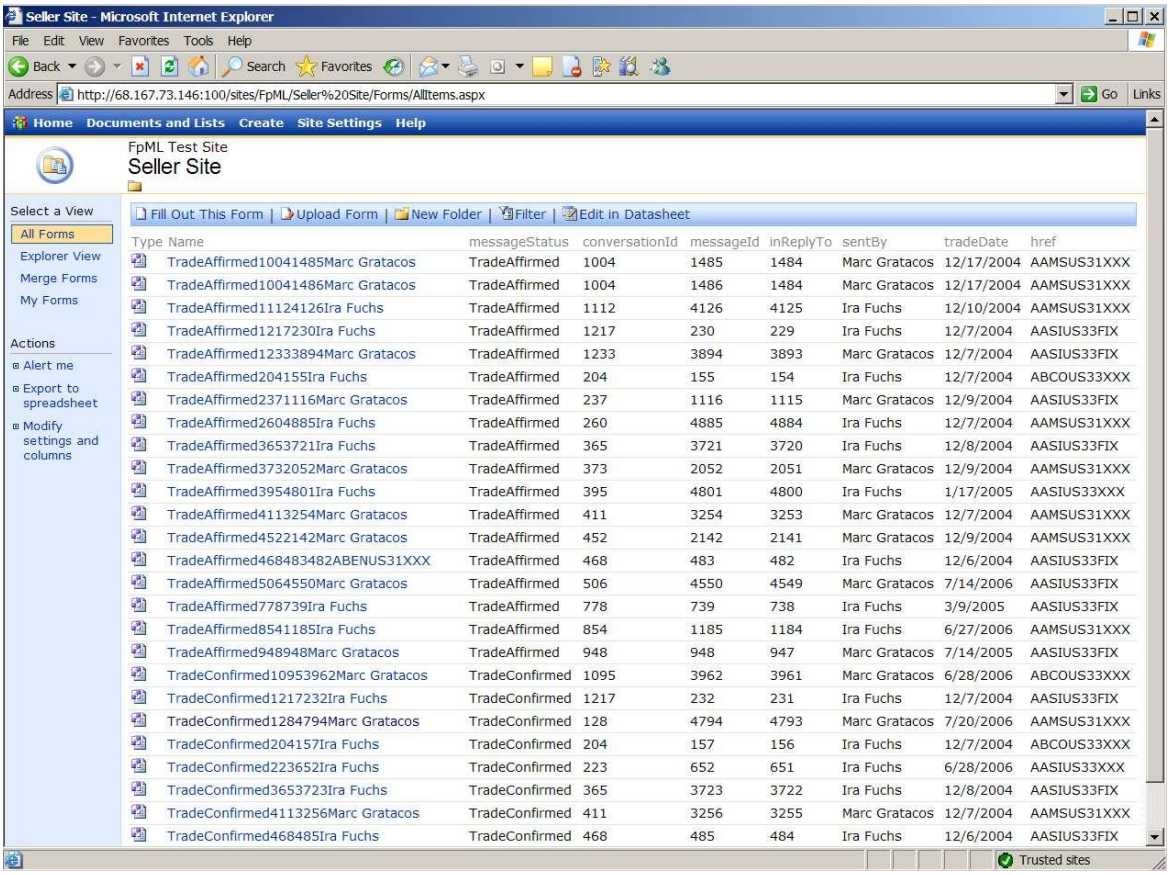



**Illustration 15 – the underlying XML document containing the values populated by the form.**

SharePoint Server uses a form library to store these XML document files as well as host the InfoPath form template for creating new instances of the form. When a document-based form is saved to a form library, any values in the form can be specified to automatically populate the columns in the form library. This can be seen in Illustration 16 below.




| Type | Name | messageStatus | conversationId | messageId | inReplyTo | sentBy | tradeDate | href |
|---|---|---|---|---|---|---|---|---|
| | TradeAffirmed10041485Marc Gratacos | TradeAffirmed | 1004 | 1485 | 1484 | Marc Gratacos | 12/17/2004 | AAMSUS31XXX |
| | TradeAffirmed10041486Marc Gratacos | TradeAffirmed | 1004 | 1486 | 1484 | Marc Gratacos | 12/17/2004 | AAMSUS31XXX |
| | TradeAffirmed11124126Ira Fuchs | TradeAffirmed | 1112 | 4126 | 4125 | Ira Fuchs | 12/10/2004 | AAMSUS31XXX |
| | TradeAffirmed1217230Ira Fuchs | TradeAffirmed | 1217 | 230 | 229 | Ira Fuchs | 12/7/2004 | AASIUS33FIX |
| | TradeAffirmed12333894Marc Gratacos | TradeAffirmed | 1233 | 3894 | 3893 | Marc Gratacos | 12/7/2004 | AASIUS33FIX |
| | TradeAffirmed204155Ira Fuchs | TradeAffirmed | 204 | 155 | 154 | Ira Fuchs | 12/7/2004 | ABCOUS33XXX |
| | TradeAffirmed2371116Marc Gratacos | TradeAffirmed | 237 | 1116 | 1115 | Marc Gratacos | 12/9/2004 | AASIUS33FIX |
| | TradeAffirmed2604885Ira Fuchs | TradeAffirmed | 260 | 4885 | 4884 | Ira Fuchs | 12/7/2004 | AAMSUS31XXX |
| | TradeAffirmed3653721Ira Fuchs | TradeAffirmed | 365 | 3721 | 3720 | Ira Fuchs | 12/8/2004 | AASIUS33FIX |
| | TradeAffirmed3732052Marc Gratacos | TradeAffirmed | 373 | 2052 | 2051 | Marc Gratacos | 12/9/2004 | AAMSUS31XXX |
| | TradeAffirmed3954801Ira Fuchs | TradeAffirmed | 395 | 4801 | 4800 | Ira Fuchs | 1/17/2005 | AASIUS33XXX |
| | TradeAffirmed4113254Marc Gratacos | TradeAffirmed | 411 | 3254 | 3253 | Marc Gratacos | 12/7/2004 | AAMSUS31XXX |
| | TradeAffirmed4522142Marc Gratacos | TradeAffirmed | 452 | 2142 | 2141 | Marc Gratacos | 12/9/2004 | AAMSUS31XXX |
| | TradeAffirmed468483482ABENUS31XXX | TradeAffirmed | 468 | 483 | 482 | Ira Fuchs | 12/6/2004 | AASIUS33FIX |
| | TradeAffirmed5064550Marc Gratacos | TradeAffirmed | 506 | 4550 | 4549 | Marc Gratacos | 7/14/2006 | AASIUS33FIX |
| | TradeAffirmed778739Ira Fuchs | TradeAffirmed | 778 | 739 | 738 | Ira Fuchs | 3/9/2005 | AASIUS33FIX |
| | TradeAffirmed8541185Ira Fuchs | TradeAffirmed | 854 | 1185 | 1184 | Ira Fuchs | 6/27/2006 | AAMSUS31XXX |
| | TradeAffirmed948948Marc Gratacos | TradeAffirmed | 948 | 948 | 947 | Marc Gratacos | 7/14/2005 | AASIUS33FIX |
| | TradeConfirmed10953962Marc Gratacos | TradeConfirmed | 1095 | 3962 | 3961 | Marc Gratacos | 6/28/2006 | ABCOUS33XXX |
| | TradeConfirmed1217232Ira Fuchs | TradeConfirmed | 1217 | 232 | 231 | Ira Fuchs | 12/7/2004 | AASIUS33FIX |
| | TradeConfirmed1284794Marc Gratacos | TradeConfirmed | 128 | 4794 | 4793 | Marc Gratacos | 7/20/2006 | AAMSUS31XXX |
| | TradeConfirmed204157Ira Fuchs | TradeConfirmed | 204 | 157 | 156 | Ira Fuchs | 12/7/2004 | ABCOUS33XXX |
| | TradeConfirmed223652Ira Fuchs | TradeConfirmed | 223 | 652 | 651 | Ira Fuchs | 6/28/2006 | AASIUS33XXX |
| | TradeConfirmed3653723Ira Fuchs | TradeConfirmed | 365 | 3723 | 3722 | Ira Fuchs | 12/8/2004 | AASIUS33FIX |
| | TradeConfirmed4113256Marc Gratacos | TradeConfirmed | 411 | 3256 | 3255 | Marc Gratacos | 12/7/2004 | AAMSUS31XXX |
| | TradeConfirmed468485Ira Fuchs | TradeConfirmed | 468 | 485 | 484 | Ira Fuchs | 12/6/2004 | AASIUS33FIX |

**Illustration 16 – The form library containing the document-based forms and the column metadata automatically populated from the form content.**

# Functional Differences between List Forms and Document-based Forms

There are three fundamental differences between the functionality of a list form and a document-based form. The first is that the schema structure for a list form is flat; it is not possible to build hierarchical information sets with grouped and nested items. Nor can you apply repeating, optional, or choice behavior to individual elements or groups as was required by the New Initiative Approval process described above. This is because SharePoint lists are flat structures and the fields available for use in the form reflect that flatness. Illustration 17 below shows the way columns in a list are represented as InfoPath form fields.

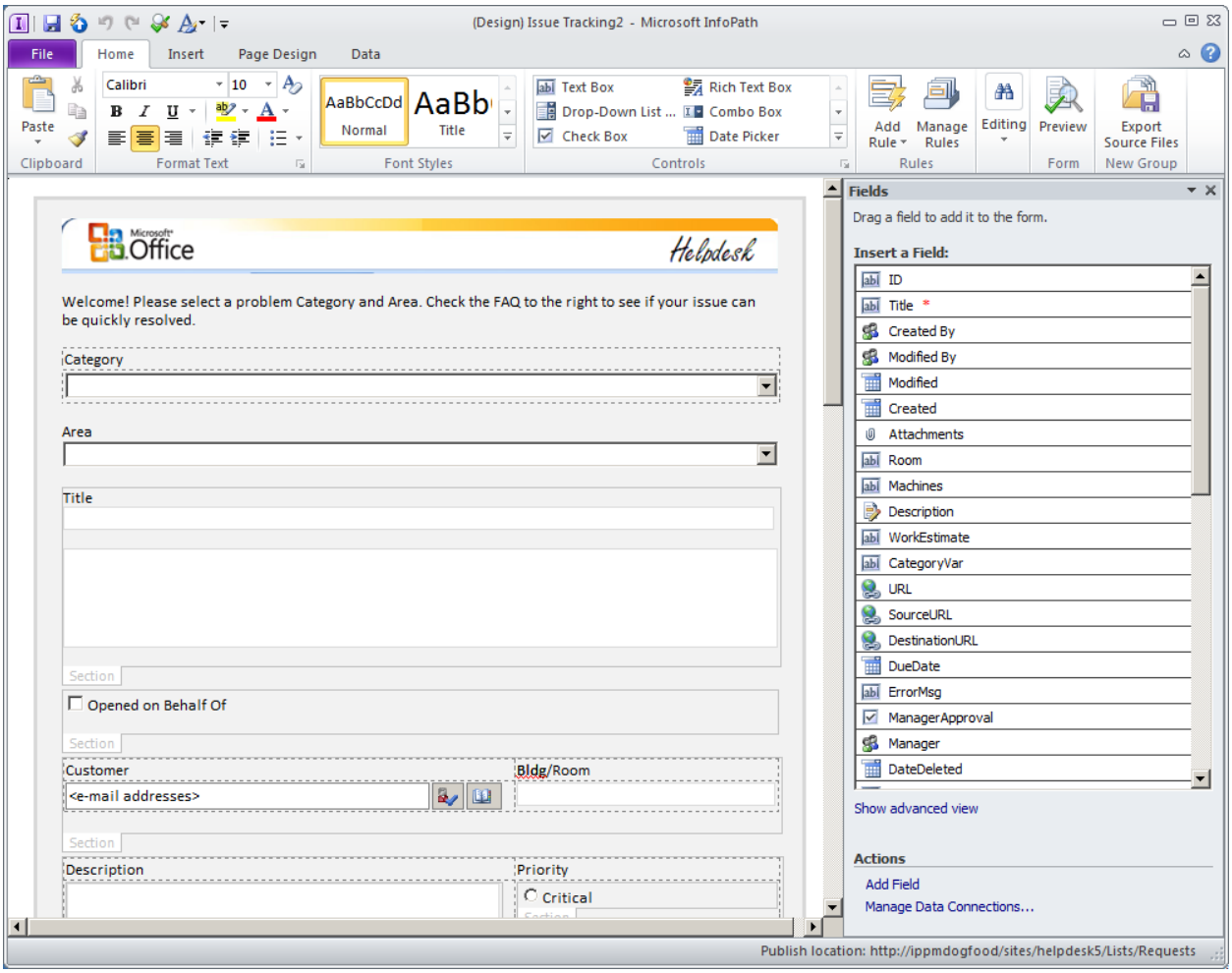



**Illustration 17 – The way list columns are represented as InfoPath form fields.**

The second difference is the reduced set of controls available for use in a list form, reflecting the limitations of the flat structure of a SharePoint list and consequently the form behavior. Illustration 18 (below left) displays the available form controls that can be used in a list form. Illustration 19 (below right) shows the available controls that can be used in a document-based form. It should be note that some of these document-based form controls, such as a Scrolling Region and Horizontal Repeating Table cannot be used in forms that will be rendered in a browser.

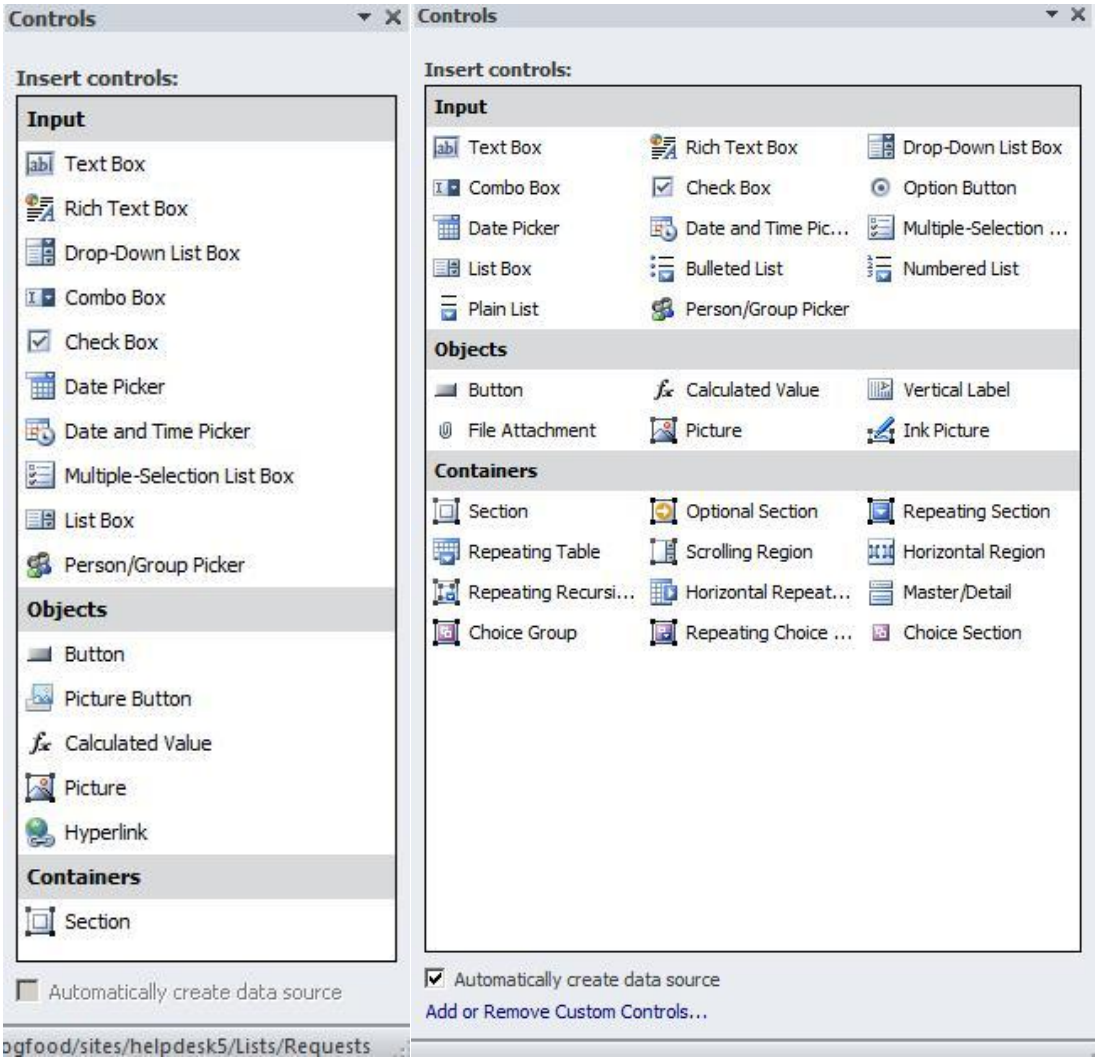



**Illustration 18 (left) – The controls available for use in a list form.**
**Illustration 19 (right) – The controls available for use in a document-based form.**

The last difference is that a list form does not generate an XML document containing the information entered and gathered in the form based on the schema, as does a document-based InfoPath form. A list form is meant to populate the columns in the host list only.

# Choosing the Type of InfoPath Form to Use in an Application

A good rule of thumb to apply here is that a SharePoint list should not contain more than fifty columns. This is not a performance or engineering limitation, simply a practical matter of usability. A list with more than fifty columns becomes unwieldy to display and interact with, even when multiple list views are created to do so. If your application requires more than fifty columns of information, it most likely reflects the documentation requirements of a complex process or transaction and the information handling capabilities of a document-based form will be the appropriate choice.

Illustration 20 displays the group node organization and structure of the information set schema for the New Initiative Approval Process application. Illustration 21 shows the fully exploded section of the schema for the Budgeting_ROI_Section. There are approximately 150 nodes in the entire schema.

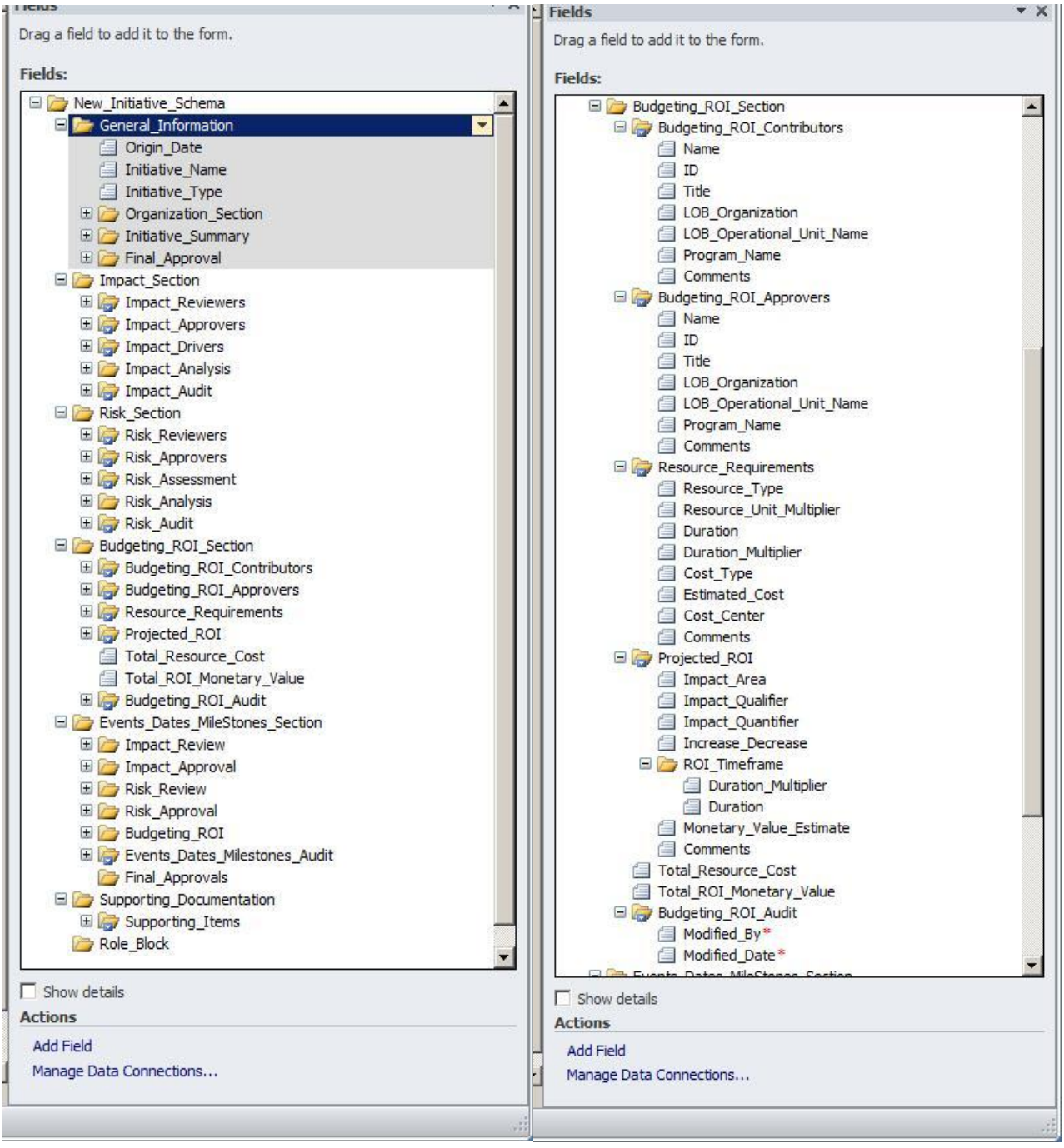



**Illustration 20 (left) – The group node organization and structure for the New Initiative Approval Process information set.**
**Illustration 21 (right) – The fully exploded section of the schema for the Budgeting_ROI_Section.**

Every industry has collaborated to develop standardized transactional information sets based on XML Schema that automate the exchange and processing of information among internal and external parties.

Examples of these are ACORD in the insurance sector, SWIFT in banking, HL7 in Health Care, and EDI in manufacturing. Illustration 22 below shows a segment of an InfoPath form that generates the documentation for an ACORD property and casualty insurance policy. The XML information set generated will be transmitted and processed by different parties involved in the underwriting of the policy. Again, in the scenario where the documentation is exchanged among multiple parties and systems a document-based form is appropriate.

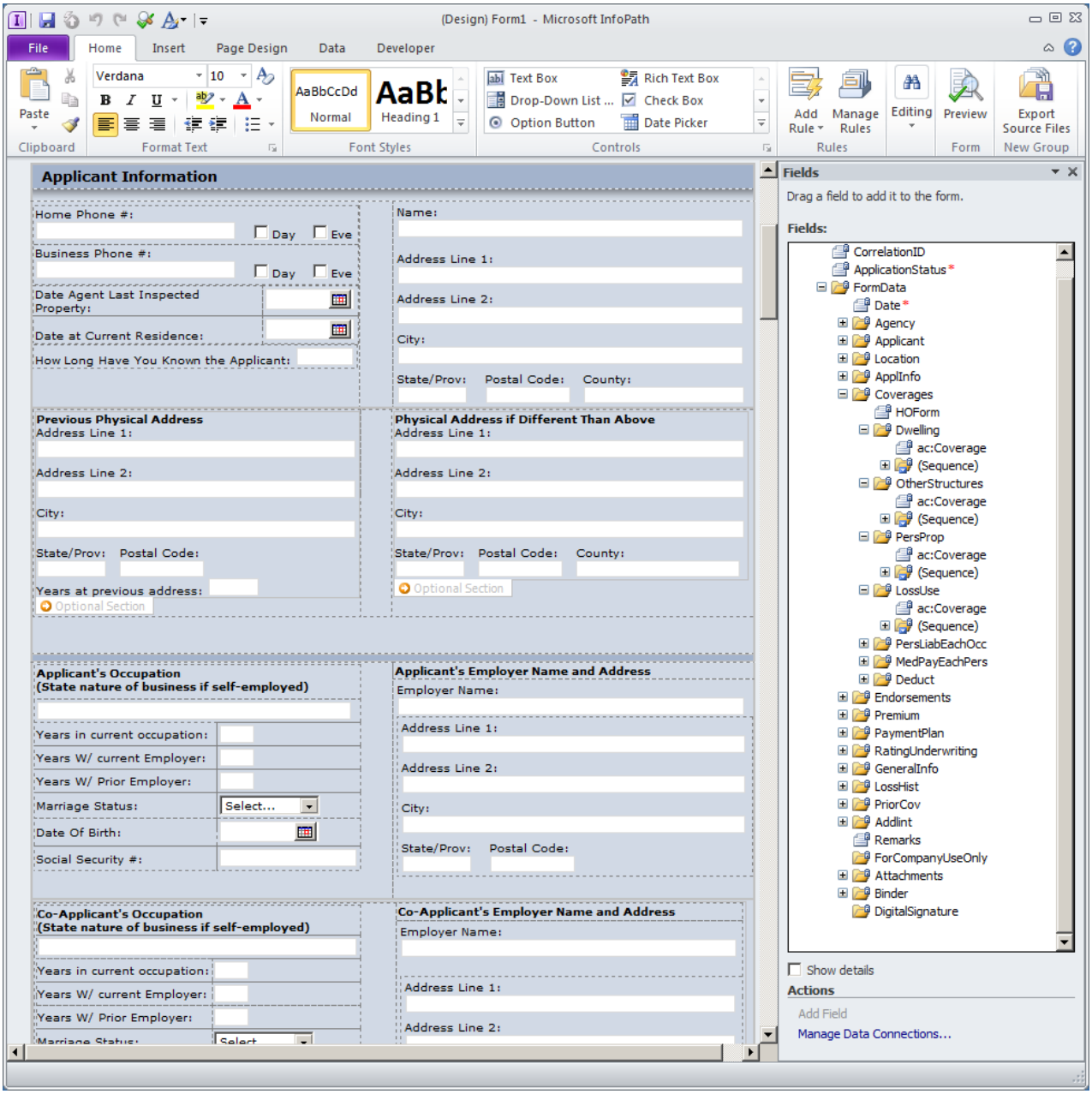



**Illustration 22 – The segment of an InfoPath form for generating the ACORD documentation for a property and casualty insurance policy.**

# Conclusion

The enhanced features of InfoPath 2010 and its tighter integration with SharePoint Server 2010 make InfoPath an enterprise-class development tool. With the ability to access any number of data sources dynamically and apply sophisticated rule sets to manipulate information and form behavior, InfoPath forms and Web Parts can address the most demanding enterprise application requirements. Furthermore, all of this rich functionality is available declaratively without requiring procedural code. This makes InfoPath a highly accessible and enabling tool for both professional developers and sophisticated end-users. InfoPath forms can now be used to create and edit items in SharePoint lists and workflow tasks, thereby significantly enhancing the functionality of SharePoint lists and workflows. For process-centric applications that require the creation and manipulation of complex information sets, the inherent implementation of XML technologies in InfoPath provides unparalleled capabilities for working with the most demanding structural and behavioral information requirements.

# About the Author

Ira Fuchs is a Business Productivity Solution Specialist at Microsoft. In this role, Mr. Fuchs works with SharePoint developers in large organizations, assisting them in designing and developing SharePoint-based applications. He is presently writing a book—*SharePoint 2010 Cookbook: Recipes for Creating Enterprise Class Applications and Components*—that will be published in the latter half of 2010.