

About the InfoPath Developer's Reference

The InfoPath Developer's Reference contains Help in the following areas:

Programming Concepts Contains conceptual information about Microsoft Office InfoPath 2003 form development. Subject areas include form architecture, the [form definition \(.xsf\) file](#), the programming environment, and using the InfoPath [object model](#).

Developer Sample Forms Contains documentation for a set of forms designed to demonstrate various development techniques for customizing and implementing InfoPath features.

InfoPath Object Model Reference Contains documentation for the InfoPath object model, including all of its collections, objects, properties, methods, events, and enumerations.

InfoPath XSF Reference Contains documentation for the InfoPath .xsf file, including all of its namespaces, types, groups, elements, and attributes.

About ActiveX controls

You can host ActiveX controls in forms. These controls can be preexisting (with some constraints) or can be written specifically for InfoPath.

- ▶ [Write an ActiveX control](#)
- ▶ [Add an ActiveX control to the InfoPath design environment](#)
- ▶ [Deploy an ActiveX control](#)

About ADO.NET dataset integration

You can easily connect forms to Web services that receive data from and submit data to ADO.NET datasets.

- ▶ [Connecting forms to Web services to receive data](#)
- ▶ [Connecting forms to Web services to receive and submit data](#)
- ▶ [Connecting forms to Web services to submit data](#)

There are a few limitations on datasets:

Only one dataset is allowed when a form is edited.

Copying and pasting of nested datasets updates the foreign key to point to the parent values.

DeleteRule and **UpdateRule** properties are not supported.

Constraints or relationships cannot be disabled.

About backward compatibility

In Microsoft Office 2003 Editions Service Pack 1, existing forms function the same way they did in Microsoft Office InfoPath 2003, while taking advantage of the improvements and new features included in the service pack. To ensure compatibility for users of Microsoft Office InfoPath 2003 without Service Pack 1 or later installed, however, certain features are not available to existing forms.

The following sections describe areas of functionality that may be affected when you are working with more than one version of InfoPath.

- ▶ [Upgrading existing forms](#)
- ▶ [Creating new form templates](#)
- ▶ [Removing new features from a form template](#)
- ▶ [Disabling new features](#)
- ▶ [Opening form templates that contain form code with updated object model members](#)
- ▶ [Launching and automating InfoPath from another application](#)

About changing a form's main data source

At times, you may need to modify a form to use a different [XML Schema](#), database server, or Web service before or after deployment. You do not have to modify the [form definition file \(.xsf\)](#) to accomplish this task, however, because with Microsoft Office InfoPath 2003 Service Pack 1 or later, you can change the main data source of your [form](#) through the InfoPath designer.

The steps you need to take to do this differ, depending on whether your form's main data source is an XML Schema, an XML document, or a data connection.

- ▶ [Change the main data source of a form based on an XML Schema or XML document.](#)
- ▶ [Change the main data source of a form based on a data connection.](#)

About digital signatures

Take advantage of the following new functions that have been added to the digital signatures feature:

Enable signatures for the entire form, or for specific sets of data in the form that can be signed separately.

For each set of data that can be signed, specify whether a single or multiple signatures are allowed and what their relationship will be. For example, you can specify whether they are parallel cosignatures or whether each new signature countersigns all the earlier ones.

Specify a message to be shown to form users as they sign the form.

Insert and see a signature in the document, and view the form as it was presented to each signer.

View verifiable nonrepudiation information that has been added to each signature for increased security. This additional information is part of the signature and cannot be removed without invalidating the signature. At any time, you can recall this data by clicking on the signature in the form.

Add custom information to the signature block in fully trusted forms through an extensive digital signature object model.

Access a signature through a snapshot. The snapshot is a file in .png format. It contains a view of the signature within the form and has all its nonrepudiation information.

About enabling forms to submit data

Microsoft Office InfoPath 2003 allows you to submit data entered into a [form](#) to a Web service or a Windows SharePoint Services [form library](#), or as an attachment to an e-mail message.

- ▶ Set up a connection to a SharePoint Services form library that is enabled for form submission.
- ▶ Set up an e-mail message that is enabled for form submission.
- ▶ Set up a connection to a Web service that is enabled for data submission.

About form architecture

Microsoft Office InfoPath 2003 forms are composed of several files and components that are combined to provide specific functionality for a particular end user scenario or business need. InfoPath forms can vary in complexity depending on the type of need that they address.

An InfoPath form is essentially a type of application that creates a specified class of XML documents, defines their layout and editing behavior, enforces their data consistency, and provides the routing information that indicates where they should be stored.

It is important to understand that InfoPath forms are composed of several different files of many different types; these files are collectively known as the [form files](#). Usually, an InfoPath form is composed of the following types of files.

| Name | Extension | Description |
|------------------------|------------------------------|---|
| Form definition | .xsf | An InfoPath-generated file that contains information about all of the other files and components used in a form. This file serves as the manifest for the form. |
| XML Schema | .xsd | The XML Schema files that are used to constrain and validate a form's underlying XML document files. |
| View | .xsl | The presentation logic files that are used to present, view, and transform the data contained in a form's underlying XML document files. |
| XML template | .xml | The .xml file that contains the default data that is displayed in a view when a new form is created. |
| XML component template | .xct | The .xml file representations of the editing controls that are used when creating and filling out a form. |
| Presentation | .htm, .gif, .bmp, and others | The files used in conjunction with the view files to create a custom user interface. |

| | | |
|----------------|------------|---|
| Business logic | .js, .vbs | The script files (Microsoft JScript and Microsoft VBScript) that contain programming code used to implement specific editing behavior, data validation , event handlers , control of data flow, and other custom business logic. |
| Binary | .dll, .exe | The custom Component Object Model (COM) components that provide additional business logic. |
| Form template | .xsn | The compressed file format (.cab) that packages all the form files into one file. |

About Human Workflow Services support

Human Workflow Services (HWS) is a service provided by Microsoft BizTalk Server 2004 that enables client applications to build and manage human-oriented workflow. Microsoft Office InfoPath 2003 provides access to this service by way of the **Workflow** task pane. In this task pane, users can start and track HWS actions and respond to HWS tasks. InfoPath does not provide a designer interface to enable the **Workflow** task pane; all HWS functionality is enabled by modifying the [form definition file \(.xsf\)](#).

Using the Workflow Task Pane

The **Workflow** task pane allows the user to start or extend a workflow or respond to a task.

The three sections of the **Workflow** task pane are:

Start Workflow Consists of the action buttons that can be used to start a workflow. This section is only visible for forms that are HWS enabled and that do not have an existing workflow associated with them.

Action Tracks the status of actions and their associated tasks. Once a user starts an action, the action runs and is tracked by the HWS service. InfoPath displays the status of the action and who is assigned the task.

Task Allows the user to work with tasks. When a task is assigned to a user, InfoPath displays the status of the task, a button to respond directly to the task, and one or more buttons to start new actions.

- ▶ [Enabling the Workflow task pane](#)
- ▶ [Adding allowed actions to forms](#)
- ▶ [Adding allowed tasks to forms](#)
- ▶ [Using the OnClick event to add action and task buttons](#)

Creating an HWS Adapter

Because the **Workflow** task pane does not submit data to the HWS service automatically, you must create an HWS adapter to submit the form.

There is no designer interface associated to the submit adapters. To enable these adapters, modify the [form definition file \(.xsf\)](#) to include the **hwsAdapter** element within the **dataAdapters** element.

Like other InfoPath Web service adapters, the element must include a name for the adapter and the location of the Web Services Description Language (WSDL), and must specify whether the adapter allows submitting and querying. Because these adapters are only used to submit data to an HWS service, the **submitAllowed** attribute must be set to "yes". The **queryAllowed** attribute can be omitted, which corresponds to a default value of "no", or it can be included and set to "no".

As part of the design of actions and tasks, you can specify additional parameters that are specific to each action or task. Below is an example of an adapter that starts a workflow with an **Approval** action. The value to use for the **hwsOperation** element type is "addActionToNewActivityFlow". The **typeID** attribute uses the **actionTypeID** attribute value, which is a GUID.

```
<xsf:hwsAdapter name="Start Approval" wsdlUrl="http://.../HWSServ
<xsf:hwsOperation type="addActionToNewActivityFlow" typeID="{
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/param
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/param
  </xsf:input>
</xsf:hwsOperation>
</xsf:hwsAdapter>
```

The **hwsAdapter** element has the same default interface as the other submit adapters supported by InfoPath, but the HWS submit adapters do not appear in the interface. Instead, to invoke a submit action, you must

add it to your code, as shown in the following example:

```
XDocument.DataAdapters.Item("Start Approval").Submit();
```

At runtime, when the submit operation is invoked, a button with the caption, **Get Status**, appears on the **Workflow** task pane. This button allows the user to refresh the task pane so that it shows the current workflow information.

- ▶ [Defining HWS submit operations for actions](#)
- ▶ [Defining HWS receive operations](#)

About programming InfoPath

Microsoft Office InfoPath 2003 provides developers with a rich development environment for customizing forms. Forms can be customized by writing programming code to respond to form and [data validation](#) events, to access and manipulate a form's underlying XML document, to implement custom data submission and merges, and to implement access to external data sources. Customizing a form can also involve altering one of the [form files](#) to suit a specific requirement.

It is important to understand a few key concepts that are involved in programming an InfoPath form. These concepts include the InfoPath programming components, programmatic and declarative programming, the InfoPath programming languages, and the InfoPath programming environment.

- ▶ [InfoPath programming components](#)
- ▶ [Programmatic and declarative development](#)
- ▶ [InfoPath programming languages and the programming environment](#)

About secondary data sources

Microsoft Office InfoPath 2003 Service Pack 1 adds the ability to connect multiple [data sources](#) to a form simultaneously. Using multiple data sources in your forms allows you to build complex functionality such as lookup lists or offline data stores.

Main and secondary data sources

The *main data source* corresponds to the groups and fields containing the data that underlies the form and is saved as the form file.

A *secondary data source* is used to store data from additional data sources. Whenever a new data connection is created for retrieving data — for example, by using the **Add** button of the **Data Connections** dialog box— the retrieved data is not stored in the main data source but in a secondary data source. The secondary data source receives the data from any data connection used for querying, whether it is an XML document, a database, a Web service, or a SharePoint library or list.

The **Data Source** task pane displays the structure (groups and fields) of each data source, both main and secondary, and is used for binding groups or fields in the data source to controls in the form. When a user fills out the form, the controls display the data from the data source and allow it to be updated during editing.

New features for working with secondary data sources

The following sections describe the changes made to working with secondary data sources in InfoPath Service Pack 1.

- ▶ [The Data Connections dialog box](#)
- ▶ [Data Source task pane](#)
- ▶ [Fields and groups](#)
- ▶ [Controls available for binding to secondary data sources](#)

About security and deployment

Additional security features and deployment functionality have been added to Microsoft Office InfoPath 2003 in Service Pack 1. Support has been added to allow form templates to be moved from one location to another or sent as an attachment to an e-mail message. In addition, support has been added to the InfoPath design mode to facilitate the creation and deployment of fully trusted forms.

Security levels

Form templates can have one of three different security levels, depending on where the form is located. These security levels are as follows:

- ▶ [Restricted](#)
- ▶ [Domain](#)
- ▶ [Full Trust](#)

Note All forms generated in the InfoPath designer have a security level associated with them. InfoPath will attempt to open forms at their associated security level. If the security level associated with the form is higher than the security level that can be granted to it, the form will not open.

Forms are granted security levels based on the location from which the form was opened. For more information, see the **Trust levels** section.

Trust levels

The highest level of trust granted to a form template is determined by the "cached from" location (that is, where the form is cached from) and other verification code, as described in the following table. The attributes listed in the table (for example, HTTP, UNC, **requireFullTrust**) are cache-based entries that are used to determine the level of trust granted to a form.

| | Trust Level Granted | Trust Level Granted | Trust Level Granted |
|---|---------------------|------------------------------------|-----------------------------|
| Highest Level of Trust Granted | Full Trust | Client Computer (Sandboxed) | Intranet (Sandboxed) |
| file: LocationId=CachedFromLocation | | X | |
| file: LocationId<>CachedFromLocation or no LocationId (regardless of where the form came from) | | | |
| CachedFromLocation: Intranet HTTP or HTTPS | | | X |
| CachedFromLocation: Internet HTTP or HTTPS | | | |
| CachedFromLocation: UNC | | | X |
| Installed Template (requireFullTrust="yes") | X | | |
| Installed Template (requireFullTrust="no") | | X | |
| Template with trusted publisher certificate | X | | |
| Extracted Form Files | | X | |

All form files opened in the InfoPath editor are bound by a set of conditions that determine the security level in which the form will open

and whether it will open. When an InfoPath form is opened in the editor, it will either be opened with an appropriate security level, or it will fail to load. If a form requests a higher security level than it can be granted (a form can request a specific security level using the **trustLevel** or **requireFullTrust** attribute), it will not be permitted to load. Otherwise, it will be loaded with the security level it requests. If the form template is not permitted to open with the requested security level, the user will not be able to open the form and will receive the "Insufficient Security Privilege Warning" error message.

The following table describes the conditions required for opening a form at each security level and the resultant behavior when the user attempts to open the form:

| | | Form asks for: | Form asks for: |
|---|---|--|--|
| | Editor Opens/Fails | Full Trust (requireFullTrust="yes") | Domain Trust (trustLevel="Domain" or blank) |
| Highest trust level InfoPath can grant based on evidence | Trusted (installed or trusted certificate) | Editor opens at Full Trust level | N/A |
| Highest trust level InfoPath can grant based on evidence | Domain Trust: Client Computer | Fails to open | Editor opens at Domain level |
| Highest trust | | Fails to open | Editor opens at Domain level |

| | | | |
|--|------------------------------|---------------|---------------------------------|
| level InfoPath can grant based on evidence | Domain Trust: Intranet | | |
| Highest trust level InfoPath can grant based on evidence | Domain Trust: Internet | Fails to open | Editor opens at Domain level |
| Highest trust level InfoPath can grant based on evidence | Restricted | Fails to open | Fails to open |

► Specifying a security level

Mail deployment and mobile form templates

Microsoft Office InfoPath 2003 Service Pack 1 allows you to send your form templates as an attachment to an e-mail message and to move them from one location to another. Mail deployment is an easy and effective way to distribute forms for interoffice use as well as to deploy forms to remote users.

- ▶ [Understanding form identity](#)
- ▶ [Designing a form to send as an attachment to an e-mail message](#)
- ▶ [Sharing forms by e-mail message or from a common shared location](#)
- ▶ [Compatibility](#)

About the form definition file

The Microsoft Office InfoPath 2003 form definition (.xsf) file, commonly referred to as the .xsf file, serves as the manifest of an InfoPath form. The .xsf file is automatically created by InfoPath when a new [form template](#) is created and saved in [design mode](#). As form designers and developers change the form template or add new features to it, InfoPath updates the .xsf file to reflect those changes.

Note The .xsf file can also be modified directly by using any text or XML editor, such as Microsoft Notepad. However, care should be taken when making modifications to the .xsf file: if an invalid entry is made, the form that the file is associated with may be left in an unusable state. However, some customizations can be made to the .xsf file that cannot be generated in design mode.

The .xsf file is the core file of an InfoPath form because it contains information about the form as a whole. Some of the data it contains includes processing metadata, user interface customizations, schema definitions, views, business logic declarations, [event handlers](#), and deployment information.

The following are some of the items that the .xsf file contains:

A unique identifier for the form

Global metadata information about the form, including deployment and publishing information

The [XML Schema](#) definitions for the XML document that the form produces

Definition of views and their associated user interface components (menus, toolbars, and buttons)

Definition of editing actions that are made available using user interface components, and how their availability will be determined contextually

Workflow and routing information

Event handlers, [data validation](#), and business logic that is associated with individual XML nodes of the form's underlying XML document, or with the

XML document itself

Event handlers associated with the form as a whole

Packaging information about all of the files contained within the form template

The .xsf file is based on the [xsf](#) namespace. Its [root element](#) is the [xDocumentClass](#) element.

Note A complete reference to the .xsf file, including all of its elements, attributes, and other entities, can be found in the InfoPath XSF Reference that is part of the InfoPath Developer's Reference, available in the InfoPath Help system and the [Microsoft Script Editor \(MSE\)](#) Help system.

About the programming environment

Microsoft Office InfoPath 2003 uses the [Microsoft Script Editor \(MSE\)](#) as its primary integrated development environment. Microsoft [JScript](#) and Microsoft [VBScript](#) are the programming languages that are used in MSE to create custom business logic for a form.

MSE can be opened when working with a form in [design mode](#) by pointing to **Script** on the **Tools** menu and clicking **Microsoft Script Editor**, or by pressing ALT+SHIFT+F11. When you open MSE from InfoPath, the MSE code editor appears and the form's default scripting file (with either a .js or .vbs extension, depending on the scripting language set for the form) opens in the code editing window.

Working with MSE in an InfoPath form involves several tasks, including setting the default scripting language, creating an event handler, and debugging scripting code.

- ▶ [Setting the default scripting language](#)
- ▶ [Creating an event handler](#)
- ▶ [Debugging scripting code](#)

Note For more information about using MSE, click **Microsoft Script Editor Help** on the **Help** menu when working in MSE.

About Visual Studio .NET integration

Microsoft Office InfoPath 2003 Service Pack 1 or later allows you to use Microsoft Visual Studio .NET to write managed code instead of script for form code in new InfoPath form templates or to add managed code to existing form templates. To use Visual Studio .NET to create, debug, and build InfoPath projects that use Microsoft Visual C# or Visual Basic .NET managed code, you must download the Microsoft® Office InfoPath™ 2003 Toolkit for Visual Studio® .NET from the [InfoPath Developer Center](#) on the Microsoft Developer Network (MSDN) Web site and install it with Visual Studio .NET 2003. The integration features provided by the toolkit allow you to use a combination of InfoPath for form design and Visual Studio .NET for writing and debugging form code. These integration features allow you to take advantage of such features of the Visual Studio .NET development environment as:

Building your project in debug or release configurations.

Using IntelliSense support in the Code Editor such as statement completion, members lists, and inline parameter information.

Pressing F1 in the Code Editor to display context-sensitive help information about types, members, and other code keywords.

Using Visual Studio .NET debugging features.

Additionally, using managed code in InfoPath projects allows you to take advantage of the features of the .NET Framework common language runtime (CLR) and to make calls into the comprehensive, object-oriented collection of reusable types provided by the .NET Framework class library.

For more information on using Visual Studio .NET to create InfoPath projects, see the documentation installed with the Microsoft Office InfoPath 2003 Toolkit for Visual Studio .NET.

About XML Schema improvements

Many improvements have been made for [XML Schema](#) (.xsd) support in Microsoft Office InfoPath 2003 Service Pack 1.

Native support for the following XML Schema constructs have been added to InfoPath:

Abstract types

Abstract elements and substitution groups

Required wildcard (<any>)

Repeating or optional model groups in the schema (sequence, choice, group, and all with **minOccurs** or **maxOccurs** different from 1)

Inline schemas

Interactive incremental creation of the data source

To support abstract types or elements and required wildcards, creation of the data source is now interactive. For example, schema creation is not possible if the schema delegates part of its syntax to other schemas. This can happen when a type or an element is defined as abstract or when the content model of an element contains an <any>. When one of these conditions occurs, InfoPath will prompt the user to specify a schema source that contains the needed data before proceeding. This process is repeated until the schema collection that is loaded contains enough information to generate a complete data source.

Better model group support

Model groups specified as "choice" or "sequence" and with **minOccurs** or **maxOccurs** not equal to 1 are now shown in the **Data Source** task pane. They can be bound to controls, and they can be treated as a group in the data source.

Note Model groups designated as "group" or "all" are not displayed in the **Data Source** task pane.

Native support for inline schemas

Inline schemas are natively supported during design. This feature allows you to use an XML document that contains its own schema as a data source.

Support for data source change

The underlying data structure of a form can now be changed during form creation. Clicking **Convert Main Data Source** on the **Tools** menu allows you to convert the current data source while retaining the information that depends on it.

Changing a data source is restricted to the following paths:

Converting from a blank form, XML Schema, or XML file to another XML Schema or XML file.

Converting from a Web service or other data connection to another Web service or data connection.

Once you specify the new data source, all dependent data will be mapped to the new data source. If the data cannot be mapped, controls will appear unbound in the view.

Note Changing the data source does not update the code associated with the form. Also, after using the **Convert Main Data Source** command, you cannot use the **Undo** command on the **Edit** menu to revert to the original data source.

Recursive template support

The **Repeating Recursive Section** control adds recursive structures of arbitrary depth to a form.

This control is now the default control when you drag a recursive structure from the **Data Source** task pane.

The properties of the recursive section correspond to the properties of an ordinary section, with the exception that digital signatures are not allowed. The only property that can be set in the internal recursive block is the default value of the XML fragment for additional recursive sections inserted within the outermost instance.

About the InfoPath object model

The Microsoft Office InfoPath 2003 object model is a [COM](#)-based object model that can be used to interact with InfoPath forms and their underlying XML documents. It is similar to other Microsoft Office application object models in that it implements interfaces for collections, objects, properties, methods, and events. However, the InfoPath object model is primarily used in scripting code, using the [Microsoft Script Editor \(MSE\)](#) that is built into InfoPath.

Note Unlike other Office applications that support complete application automation, the InfoPath object model supports automation of only certain features of the run-time editing environment.

The following sections discuss the InfoPath functional areas that the object model represents, the locations from which the object model can be accessed, and the levels of object model security.

- ▶ [Object model functional areas](#)
- ▶ [Object model access](#)
- ▶ [Object model security](#)
- ▶ [Viewing Service Pack 1 object model changes from the Object Browser](#)

Accessing application data

The Microsoft Office InfoPath 2003 object model provides objects and collections that can be used to gain access to information about the InfoPath application, including information related to a form's underlying XML document and the [form definition \(.xsf\) file](#). This data is accessed through the top-level object in the InfoPath object model hierarchy—the [Application](#) object.

Using the **Application** object, InfoPath form developers can access information about the currently installed instance of InfoPath, including its name and version number. In the following example, the [Name](#) and [Version](#) properties of the **Application** object return data in a message box to the user:

```
Application.XDocuments(0).UI.Alert("Application name: " + Application.  
  "\nApplication version: " + Application.Version);
```

Note that the [Alert](#) method of the [UI](#) object, which is accessed through the [XDocument](#) object, is used to display a message box to the user. Because the **XDocument** object that represents the currently open form's underlying XML document is embedded in the InfoPath script engine, the previous example can also be written as follows:

```
XDocument.UI.Alert("Application name: " + Application.Name +  
  "\nApplication version: " + Application.Version);
```

Note The \n character references in the text for the alert message is a standard new line feed that causes the text to break and be placed on a new line in the message box.

- ▶ [Accessing data about a form's XML document](#)
- ▶ [Accessing data about a form's .xsf file](#)

Accessing external data sources

When working with a Microsoft Office InfoPath 2003 form, you can write scripting code to access the form's [secondary data sources](#), which are typically external to the primary data source of a form, and then manipulate the data that they contain. The InfoPath object model supports access to a form's external data sources through the use of the [DataObject](#) object in association with the [DataObjects](#) collection.

The InfoPath object model also provides a set of [data adapter](#) objects, containing information about the secondary data sources, and access to the data that they contain. The type of data adapter that is returned by the [DataObject](#) object depends on the type of data source that was selected when the secondary data source was created in [design mode](#).

- ▶ [Overview of the DataObjects collection](#)
- ▶ [Overview of the DataObject object](#)
- ▶ [Overview of the data adapter objects](#)
- ▶ [Using the DataObjects collection and the DataObject object](#)

Accessing form data

When you want to extend the functionality of a Microsoft Office InfoPath 2003 form, it is often necessary to programmatically access information about the form's underlying XML document, access the data that the XML document contains, or perform some action on the XML document. The InfoPath object model supports access and manipulation of a form's underlying XML document through the use of the [XDocument](#) object in association with the [XDocuments](#) collection.

The **XDocument** object is one of the most useful objects within the InfoPath object model because it provides a variety of properties, methods, and events that not only interact with a form's underlying XML document, but also perform many of the actions that are within the InfoPath user interface.

- ▶ [Overview of the XDocuments collection](#)
- ▶ [Overview of the XDocument object](#)
- ▶ [Using the XDocuments collection and the XDocument object](#)

Automating InfoPath

Microsoft Office InfoPath 2003 Service Pack 1 offers expanded application automation in the form of methods of the [Application](#) object and the [XDocuments](#) collection.

- ▶ [Overview of the Application and XDocument objects](#)

For backward compatibility, the automation of InfoPath is accomplished by using the [ExternalApplication](#) object from a programming language or environment that supports the [Component Object Model \(COM\)](#).

- ▶ [Overview of the ExternalApplication object](#)

- ▶ [Using the ExternalApplication object](#)

Displaying alerts and dialog boxes

When writing programming code to extend the functionality of a Microsoft Office InfoPath 2003 form, it is often useful to provide the user with information in a dialog box. Programmatically displaying a dialog box is accomplished in InfoPath by using the [UI](#) object.

- ▶ [Overview of the UI object](#)
- ▶ [Using the UI object](#)

Handling errors

When creating custom applications, developers must often perform error handling that involves writing programming code to check for errors raised by the application or to create and raise custom errors. The Microsoft Office InfoPath 2003 object model supports error handling through the use of the [Error](#) object in association with the [Errors](#) collection.

In InfoPath, errors can occur when a form's [XML Schema](#) is validated, when a custom validation constraint fails, when an error is generated by the [ReportError](#) method of the [DataDOMEvent](#) object, or when an error is created using the [Add](#) method of the [Errors](#) collection.

- ▶ [Overview of the Errors collection](#)
- ▶ [Overview of the Error object](#)
- ▶ [Using the Errors collection and the Error object](#)

Responding to form events

You can write scripting code to respond to various events that can occur in Microsoft Office InfoPath 2003 as a user fills out a form. In InfoPath, events take the form of [event handlers](#) that are created when working with a form in [design mode](#).

InfoPath event handlers must be initially created in design mode because, in addition to the scripting declarations that are created in a form's primary scripting file, entries are also made in the [form definition \(.xsf\) file](#). After you have created an event handler, you should not alter its declaration in the primary scripting file.

For information about creating the InfoPath event handlers, see [About the programming environment](#).

- ▶ [Overview of the event objects](#)
- ▶ [Using the event objects](#)

Working with form windows

When working programmatically with a Microsoft Office InfoPath 2003 form, you can write scripting code to access the form's windows, and then customize some of the items that they contain. The InfoPath object model supports access to a form's windows through the use of the [Window](#) object in association with the [Windows](#) collection.

There are two types of windows in an InfoPath form: the editing window that is used as the form area when a user fills out a form, and the designing window that is used as the design mode when a user designs a form. When writing scripting code in a form, it is the editing window that provides the most useful functionality, because you can use the **Window** object associated with it to access a variety of properties and methods that can be used to customize a form.

- ▶ [Overview of the Windows collection](#)
- ▶ [Overview of the Window object](#)
- ▶ [Using the Windows collection and the Window object](#)

Working with views

When working with a Microsoft Office InfoPath 2003 form, you can write scripting code to access the form's views, and then perform a variety of actions on the data that the views contain. The InfoPath object model supports access to a form's views through the use of the [View](#) object.

- ▶ [Overview of the View object](#)
- ▶ [Using the View object](#)

About the developer sample forms

The Microsoft Office InfoPath 2003 developer sample forms are a set of forms designed to demonstrate various development techniques for customizing and implementing InfoPath features. The following table lists each of the developer sample forms and the features that they illustrate.

| Name | File name | Description |
|------------------------------------|--------------|--|
| Data Validation | D_VALID.XSN | Demonstrates the ways in which data can be validated in a form. Methods covered include schema-based, custom, and script-based validation. |
| Events | EVENTS.XSN | Demonstrates the events that can be used in a form. Events covered include form events, data validation events, and control events. |
| Structural Editing | CD_EDIT.XSN | Demonstrates the ways in which the source XML document of a form may be edited, and how the editing actions that are available to a user are based on context. Methods covered include field and structural editing, with examples of editing using controls and scripting code. |
| User Interface | UIBASICS.XSN | Demonstrates the ways in which the InfoPath user interface can be customized in a form. Methods covered include menu items, toolbars, custom task panes , and custom dialog boxes. |

► [Using the developer sample forms](#)

About the Data Validation developer sample form

The Data Validation developer sample form demonstrates the variety of ways in which data can be validated in Microsoft Office InfoPath 2003. This sample form is divided into three sections, each demonstrating one of the three methods of [data validation](#): schema-based validation, custom validation, and script-based validation.

To test the features of the sample form, simply choose any field and enter data that does not comply with the validation constraint that is associated with it. The validation constraints can be viewed by resting the mouse pointer over the question mark icon next to each field. When you move the cursor out of the field, you receive one of two possible error indications: an [inline alert](#) or a [dialog box alert](#).

Note To see all of the files that make up the Data Validation developer sample form, open the form and click **Extract Form Files** on the **File** menu while in [design mode](#). This extracts all of the [form files](#) from the [form template](#) file and saves them to a specified folder on your hard disk.

To learn how each of the data validation methods is implemented in the Data Validation developer sample form, see the following topics:

[Schema-based validation in the Data Validation developer sample form](#)

[Custom validation in the Data Validation developer sample form](#)

[Script-based validation in the Data Validation developer sample form](#)

Schema-based validation in the Data Validation developer sample form

Schema-based [data validation](#) in Microsoft Office InfoPath 2003 is determined by the [XML Schema](#) associated with a form. Schema-based validation occurs by default whenever a user fills out a form. After the user enters data into a field and moves out of that field, the data is immediately checked against the XML Schema.

In the Data Validation developer sample form, schema-based validation is used to verify that a number falls within a certain range, verify the type of data in a field, verify that a field contains data, and limit the number of rows that can be added to a table.

Note InfoPath supports the creation of only data type and required field constraints in XML Schemas in [design mode](#). However, the use of other kinds of constraints in the schema is supported. To create other kinds of schema-based validation, you must edit your XML Schema files using Microsoft Notepad or some other text editor.

- ▶ [Range checking and data-type validation in the schema](#)
- ▶ [Required fields in the schema](#)
- ▶ [Structural validation in the schema](#)

Custom validation in the Data Validation developer sample form

Custom (or declarative) validation in Microsoft Office InfoPath 2003 is used to check for required fields, verify a range of values, and ensure the accuracy of a field. You can use the **Data Validation** dialog box to create complex validation constraints that perform calculations and validate the data in other fields. Validation errors can be displayed as [inline alerts](#) or [dialog box alerts](#). In either case, you can write error messages that tell the user how to fix the invalid entry. As with schema-based validation, custom validation constraints are attached to fields in the form, and the validation is invoked when a user changes the data in a field.

Note The **Data Validation** dialog box is available via the **Properties** dialog box for each of the Office InfoPath 2003 controls.

When you create a custom data validation constraint in the **Data Validation** dialog box, Office InfoPath 2003 creates an [errorCondition](#) element within the [customValidation](#) section of the [form definition \(.xsf\) file](#). The following is an example of the validation constraints created in the .xsf file for the cost field in the Data Validation developer sample form:

```
<xsf:customValidation>
  <xsf:errorCondition
    match="/sampleData/travelExpenses/expense/cost"
    expressionContext="."
    expression=". &lt; 0 or . &gt; 500">
    <xsf:errorMessage
      type="modeless"
      shortMessage="The cost of the item must be more than
        $0.00 and cannot go over $500.00.">The cost of the
        item must be more than $0.00 and cannot go over $500.00.
    </xsf:errorMessage>
  </xsf:errorCondition>
</xsf:customValidation>
```

The [match](#) attribute of the **errorCondition** element specifies the [XPath](#) expression that is used to bind the custom data validation constraint to the field, and the [expression](#) attribute is used to specify the constraint.

The [errorMessage](#) element is nested within the **errorCondition** element and is used to specify the error message that appears when the value of the associated field violates the constraint. The [shortMessage](#) attribute equates to the **ScreenTip** field in the **Data Validation** dialog box, and the value of the **errorMessage** element equates to the **Message** field in the **Data Validation** dialog box. The [type](#) attribute is used to specify error type.

Script-based validation in the Data Validation developer sample form

Script-based validation is specified in the business logic script file associated with a Microsoft Office InfoPath 2003 form, and it provides more flexibility than custom validation or schema-based validation. Using script-based validation, you can specify when a field should be validated (for example, when the user first types data into a field, after the user leaves a field, or after data is validated by the schema). You can also write script that runs when a form is opened or when the user switches views. The script you write can perform a variety of functions, including comparing fields, calculating values, showing error messages, updating values, and revising other fields.

Note InfoPath supports writing script in Microsoft [VBScript](#) and Microsoft [JScript](#). However, you cannot mix the scripting languages used within a single form.

To create script-based data validation, you use InfoPath [design mode](#) to create an [event handler](#) (also called an event function), and then you write scripting code for the event handler using the [Microsoft Script Editor \(MSE\)](#). For each event handler that you create, InfoPath places an entry in the [form definition \(.xsf\) file](#) that contains the name of the event handler and the XML [Document Object Model \(DOM\)](#) node that the event handler is associated with.

Because the event handler is referenced in the .xsf file, you cannot create a new event handler entirely from within MSE— you must initially create the event handler in InfoPath design mode. Since InfoPath is used to create the declaration of the event handler, you cannot modify the event handler name or its arguments once it has been created.

- ▶ [Event handlers in the form definition file](#)
- ▶ [Script-based event handlers and functions](#)
- ▶ [Using the DataDOMEvent object](#)

About the Events developer sample form

The Events developer sample form demonstrates some of the events that may be used in a Microsoft Office InfoPath 2003 form. In InfoPath, events are used to implement client-side business logic, which may include [data validation](#), controlling the behavior of a form, responding to the actions of controls on a form, or submitting a form.

The Events developer sample form is based on the scenario of a customer contact system. It contains three views that are used to manage customer information:

New Customer View Used to enter new customer information. Customer information must be entered in this view before the other views can be used.

Contact Customer View Used to set the dates on which the customer should be contacted and to record that the customer has been contacted.

Archive Customer View Used to show customer information, record notes about the customer, and record that the customer record has been archived.

To test the features of the sample form, enter customer information in New Customer view, and then click **Switch To Contact Customer View** to switch views. After entering customer contact information in Contact Customer view, you can click **Switch To Archive Customer View** to switch to the list of archived customer contact information.

Note To see all of the files that make up the Events developer sample form, open the form and click **Extract Form Files** on the **File** menu while in [design mode](#). This extracts all of the [form files](#) from the [form template](#) file and saves them to a specified folder on your hard disk.

To learn how each event is implemented in the Events developer sample form, see the following topics:

[Form events in the Events developer sample form](#)

[Data validation events in the Events developer sample form](#)

[Control events in the Events developer sample form](#)

Form events in the Events developer sample form

The Events developer sample form implements a variety of Microsoft Office InfoPath 2003 form events that can be used to respond to specific activities that occur when a form is opened and while it is being filled out. Form events can occur when a form's version number is validated, when a form is opened, when a form view is changed, and after a form has been merged with another form.

The following table lists each of the form events implemented in the Events developer sample form, along with a description of how they are used.

| Event | Description |
|----------------------------------|---|
| OnVersionUpgrade | Used to verify that the version number of the form being opened matches the version number of the form when it was originally designed. If the version numbers do not match, this event occurs and scripting code can be used to update the form or display an error message. |
| OnLoad | Used to initialize the form as it is being opened. When this event occurs, scripting code can be used to set the appropriate view based on data contained in the form. |
| OnSwitchView | Used when changing from one view to another. This event occurs when a user changes views, and scripting code can be used to insert data into the form. |
| OnAfterImport | Used to provide additional processing after a form has been merged with another form. This event occurs after data is imported from another form, and scripting code can be used to set the appropriate view based on the merged data contained in the form. |

Note The [OnSubmitRequest](#) event is also a form event, but it was not used in the Events developer sample form.

- ▶ Using the OnVersionUpgrade event
- ▶ Using the OnLoad event
- ▶ Using the OnSwitchView event
- ▶ Using the OnAfterImport event

Data validation events in the Events developer sample form

The Events developer sample form implements a variety of Microsoft Office InfoPath 2003 [data validation](#) events that can be used to validate data that is entered into a form. Data validation events can occur after a change has been made to a [field](#) or [group](#), after a change has been made to a field or group but before the data is committed, and after a change has been made to a field or group and after the data is committed.

To use one of the data validation events, you must first create the event in the **Field or Group Properties** dialog box that is available from the **Data Source** task pane. To access this dialog box, right-click one of the fields or groups in the task pane, and then click **Properties**. On the **Validation and Script** tab, select the event you wish to create, and then click **Edit**; this opens [Microsoft Script Editor \(MSE\)](#) and displays the InfoPath-generated [event handler](#).

Note Event handlers in InfoPath must be created in [design mode](#).

The following table lists each of the data validation events implemented in the Events developer sample form, along with a description of how they are used.

| Event | Description |
|--------------------------------|--|
| OnValidate | Used to validate the data contained in the ContactDates group. When this event occurs, scripting code is used to verify that the dates used are within a specified range. |
| OnBeforeChange | Used to validate the data contained in the Email Campaign Start, Phone Contact Start, and Representative Visit date fields. When this event occurs, scripting code is used to implement business logic that checks for the existence of certain dates before others can be entered or removed. |
| OnAfterChange | Used to call a function that calculates the total campaign costs and updates the Campaign Cost |

field.

- ▶ Using the OnValidate event
- ▶ Using the OnBeforeChange event
- ▶ Using the OnAfterChange event

Control events in the Events developer sample form

The Events developer sample form implements a single Microsoft Office InfoPath 2003 control event that can respond to a button click; this is the [OnClick](#) event. The **OnClick** event occurs after a user clicks a button on a form.

Note The **OnClick** event is the only control event supported by InfoPath.

To use the **OnClick** event, you must first create the event in the **Button Properties** dialog box that is available on the shortcut menu for the button control. On the **General** tab, set **Script** as the **Action**, and then click **Microsoft Script Editor**. This will open [Microsoft Script Editor \(MSE\)](#) and display the InfoPath-generated [event handler](#).

Note Event handlers in InfoPath must be created in [design mode](#).

In the Events developer sample form, the **OnClick** event is implemented for the three buttons that are used to switch between views. The following example from the Events developer sample form shows the **OnClick** event handler for the **Switch to New Customer View** button:

```
function btnSwitchNew::OnClick(eventObj)
{
    XDocument.View.SwitchView("New Customer")
}
```

Note The name of the button used in the event handler declaration is determined by the value set for the **Script ID** box in the **Button Properties** dialog box.

About the Structural Editing developer sample form

The Structural Editing developer sample form demonstrates some of the ways in which the underlying XML document of a Microsoft Office InfoPath 2003 form may be edited using a combination of controls, menu items, toolbar buttons, and script. When filling out a form in InfoPath, users are essentially editing an XML document in an easy-to-use, graphical format. Although the Structural Editing developer sample form is primarily intended to demonstrate structural editing, it is used to illustrate two types of editing that can occur in a form:

Field editing Editing that modifies the text in a field (an [element](#) or [attribute](#) in the form's underlying XML document). For example, entering data directly into a [text box](#) control changes the underlying data in the element or attribute that the control is [bound](#) to.

Structural editing Editing that changes the structure of the form's underlying XML document. Structural editing allows fragments of XML (elements and their associated [child elements](#), attributes, and content) to be inserted, removed, or replaced in a single operation. For example, adding items to a [repeating section](#), [repeating table](#), or [list control](#) creates new elements and attributes within the part of the XML document that the control is bound to.

Both field editing and structural editing are usually implemented by using controls that connect a form to an underlying XML document, allowing users to enter or modify the data that the XML document contains. However, you may also implement field or structural editing by using scripting code in conjunction with the InfoPath object model.

In addition to field and structural editing, InfoPath provides the concept of *editing context*, which means that editing actions can be dependent on the current selection or insertion point. For example, the **Add Part Before Current** button on the **Part Toolbar** custom toolbar of the Structural Editing developer sample form inserts a new row in the Parts table at a position based on the current context. The button is enabled only if the current selection is on or within one of the rows in the table.

To test the features of the sample form, simply type data directly into the fields on the form, or use the **Insert** menu, the **Part Catalog** [custom task](#)

[pane](#), the shortcut menu, or the **Part Toolbar** custom toolbar to insert or remove rows in the Parts table.

Note To see all of the files that make up the Structural Editing developer sample form, open the form and click **Extract Form Files** on the **File** menu while in [design mode](#). This extracts all of the [form files](#) from the [form template](#) file and saves them to a specified folder on your hard disk.

To learn how editing features are implemented in the Structural Editing developer sample form, see the following topics:

[Field editing in the Structural Editing developer sample form](#)

[Structural editing in the Structural Editing developer sample form](#)

[Editing context in the Structural Editing developer sample form](#)

Field editing in the Structural Editing developer sample form

Field editing in a Microsoft Office InfoPath 2003 form occurs when data is entered directly into a form [field](#) (usually a text box control) as a user fills out a form. The data that is entered is stored as a value of the [element](#) or [attribute](#) of the XML field (element or attribute) in the underlying XML document that the control is [bound](#) to. For example, in the Structural Editing developer sample form, [text boxes](#) are used for editing the data contained in the Part Number, Description, Quantity, and Unit Price fields.

Field editing usually occurs when a user enters data within a control on a form; the data that the user enters is stored in the form's underlying XML document. However, you can also use scripting code to manipulate the data contained in the form's underlying XML document by modifying the XML document itself. If the form includes a control such as a text box that is bound to the modified data, the data displayed in the form will be updated automatically.

- ▶ [Using controls for field editing](#)
- ▶ [Using script for field editing](#)

Structural editing in the Structural Editing developer sample form

Structural editing in a Microsoft Office InfoPath 2003 form occurs when a user is filling out a form, and the type of editing actions that the user performs results in a structural change to the form's underlying XML document. Structural changes involve inserting or removing [elements](#) and [attributes](#) in the underlying XML document that the form is [bound](#) to. For example, in the Structural Editing developer sample form, structural changes are made when a user inserts or removes items from the Parts table or the Notes bulleted list.

Structural editing controls have predetermined editing actions that can be associated with toolbar buttons and menu items in InfoPath user interface areas. The **Commands** dialog box that is available from a structural editing control's **Properties** dialog box allows you to choose which editing actions of the control are available to users, which user interface area they appear in, and the labels that are used for the associated button or menu items. The structural editing controls used in this sample form include a [repeating table](#) and a bulleted [list](#).

Structural editing can be implemented by using an InfoPath structural editing control, or by using scripting code to directly manipulate the data contained in the form's underlying XML document. If the form includes a control such as a [repeating section](#) that is bound to the modified (inserted or deleted) data, the data displayed in the form will be updated automatically.

- ▶ [Using controls for structural editing](#)
- ▶ [Using script for structural editing](#)

Editing context in the Structural Editing developer sample form

Editing context in a Microsoft Office InfoPath 2003 form is the dependence of editing actions on the current selection or insertion point. For example, if a row in the Structural Editing developer sample form Parts table is selected, the buttons for adding or removing parts on the **Part Toolbar** custom toolbar are enabled and will insert or remove rows in the Parts table at a position based on the current context. If current selection is not on or within one of the rows in the table, the buttons are disabled.

Note The **Part Toolbar** custom toolbar was created by modifying the default Form toolbar.

The following are some of the areas in the InfoPath user interface in which editing context is applicable:

Insert menu Menu items available on the **Insert** menu. These typically include menu items for inserting items in a table or section.

Shortcut menu Menu items available on the shortcut menu. These typically include menu items for inserting or removing items in a table, section, or list.

Table menu Menu items available on the **Table** menu. These typically include menu items for inserting or removing items in a table.

Custom toolbar Buttons and menu items available on a custom toolbar. These typically include buttons or menu items for inserting or removing items in a table or section, or buttons that call scripting code.

In the Structural Editing developer sample form, a [repeating table](#) control is used to implement a table that allows users to edit a list of parts in an invoice. When a user chooses to edit the table using the **Insert** menu, the **Part Toolbar** custom toolbar, or the shortcut menu, InfoPath determines the control being used, and based on the editing actions set in [design mode](#) for that control, performs the appropriate editing of the source XML document.

The structural editing controls have predetermined editing actions that

can be associated with toolbar buttons and menu items in InfoPath user interface areas. The **Commands** dialog box that is available from a structural editing control's **Properties** dialog box allows you to choose which editing actions of the control are available to users, which user interface area they appear in, and the labels that are used for the associated buttons or menu items. In the Parts table of the Structural Editing developer sample form, the following editing actions are enabled:

Insert Enabled on the **Insert** menu. Allows a user to insert a new row in the Parts table.

Insert Above Enabled on the shortcut menu and the **Part Toolbar** custom toolbar. Allows a user to insert a new row in the Parts table that is above the currently selected row.

Insert Below Enabled on the shortcut menu and the **Part Toolbar** custom toolbar. Allows a user to insert a new row in the Parts table that is below the currently selected row.

Remove Enabled on the shortcut menu and the **Part Toolbar** custom toolbar. Allows a user to remove the currently selected row.

Note Other editing actions can be used in a repeating table control that are not implemented in the Structural Editing developer sample form.

About the User Interface developer sample form

The User Interface developer sample form demonstrates the variety of ways in which the user interface can be customized in a Microsoft Office InfoPath 2003 form. The user interface features customized in this sample form include menu items on the menu bar, shortcut menus, toolbar buttons on the **Form** toolbar, and a custom task pane.

To test the features of the sample form, simply add, modify, or remove CDs by using the **Insert** menu, the **CD Collection Toolbar**, or the shortcut menu buttons while filling out the form. You can switch between CD Collection and All Tracks views by using the **View** menu or the [custom task pane](#).

Note To see all of the files that make up the User Interface developer sample form, open the form and click **Extract Form Files** on the **File** menu while in [design mode](#). This extracts all of the [form files](#) from the [form template](#) file and saves them to a specified folder on your hard disk.

To learn how each of the user interface features is implemented in the User Interface developer sample form, see the following topics:

[Custom menus in the User Interface developer sample form](#)

[Custom toolbars in the User Interface developer sample form](#)

[Custom task panes in the User Interface developer sample form](#)

Custom menus in the User Interface developer sample form

Microsoft Office InfoPath 2003 provides the ability to control where and how you want menu items to appear within the user interface, primarily by enabling or disabling them on various built-in menus used with a form. Some of the menus that you can customize include **View**, **Insert**, and **Table**, as well as shortcut menus that are displayed when a user clicks the shortcut menu button associated with a certain [repeating table](#), [repeating section](#), or [optional section](#). Menus can be customized for repeating tables, repeating sections, optional sections, and [views](#).

Note Many of the other standard InfoPath menus can also be customized; the User Interface developer sample form demonstrates how to customize a select few.

Several custom menus are implemented in the User Interface developer sample form. The following table lists the views, tables, and sections of the User Interface developer sample form, along with the custom menu items that were implemented for each of them.

| Name | Type | Custom menu items |
|---------------|-------------------|--|
| CD Collection | View | CD Collection menu item on the View menu. |
| All Tracks | View | All Tracks menu item on the View menu. |
| CD | Repeating section | CD menu item on the Section submenu of the Insert menu. Insert CD above , Insert CD below , and Remove CD menu items on the shortcut menu for the CD table. |
| Track | Repeating table | Track menu item on the Section submenu of the Insert menu. Insert Track menu item on the shortcut menu for the CD table. Insert Tracks and Remove Tracks menu items on |

the **Table** menu.

Insert Track above, **Insert Track below**, and **Remove Track** menu items on the shortcut menu for the Track table.

| | | |
|-------|------------------|--|
| Label | Optional section | Label menu item on the Section submenu of the Insert menu. Insert Label menu item on the shortcut menu for the CD table. Remove Label menu item on the shortcut menu for the Label control. |
|-------|------------------|--|

Note When you customize menus in InfoPath, those customizations are applied at the view level. This means that the customizations you make are not globally applied to all views. To apply the same custom menus to more than one view, you must make the menu customizations in each view.

- ▶ [Customizing the View menu](#)
- ▶ [Customizing the Insert menu](#)
- ▶ [Customizing the Table menu](#)
- ▶ [Customizing shortcut menus](#)

Custom toolbars in the User Interface developer sample form

The User Interface developer sample form implements a single custom toolbar, the **CD Collection Toolbar**, that allows users to add or remove a CD, Track, or Label. By default, Microsoft Office InfoPath 2003 has one built-in custom toolbar, the Form toolbar, that can be used to create custom menu items that users can click to perform various actions on a form.

To customize the Form toolbar in [design mode](#), you use the **Properties** dialog box for the particular [repeating section](#), [repeating table](#), or [optional section](#) that you are working with. The **Properties** dialog box is accessed by right-clicking the section or table and clicking the **Properties** menu item on the shortcut menu. In the **Properties** dialog box for repeating tables and optional sections, you click **Customize Commands** to open the **Commands** dialog box. For repeating sections, the **Customize Command** button is available in the **Section Properties** dialog box that opens when you choose to modify the default settings of a section. The **Commands** dialog box allows you to associate certain editing actions with various command locations available on the menus and toolbars in an InfoPath form. To create a custom toolbar, you select the **Form Toolbar** location.

Note In the User Interface developer sample form, the command location is the **CD Collection Toolbar**, since the default name of the Form toolbar was modified. Modifications to the default name of a toolbar can be made manually in the [form definition \(.xsf\) file](#) by changing the value of the [caption](#) attribute of the [toolbar](#) element.

When you customize a toolbar, InfoPath creates entries in the .xsf file using [button](#) elements within the **toolbar** element. The caption that appears as the name of the toolbar is determined by the **caption** attribute of the **toolbar** element, and the caption that appears as the name of a button on the toolbar is determined by the [caption](#) attribute of a **button** element.

As you associate editing actions with the custom toolbar using the **Commands** dialog box, menu items are placed directly on the toolbar.

However, it is possible to create menus on the toolbar and then add menu items to those menus. This is accomplished by nesting the **button** elements of the toolbar within a **menu** element in the .xsf file.

The following is a section from the .xsf file of the User Interface developer sample form that contains the **toolbar** element. Note the use of the **menu** element to create a menu on the toolbar.

```
<xsf:toolbar
  caption="CD Collection Toolbar"
  name="CD Collection Toolbar">
  <xsf:button
    action="xCollection::insert"
    xmlToEdit="CD_10"
    caption="New CD"
    showIf="always">
  </xsf:button>
  <xsf:button
    action="xCollection::insert"
    xmlToEdit="Track_14"
    caption="New Track"
    showIf="always">
  </xsf:button>
  <xsf:button
    action="xOptional::insert"
    xmlToEdit="Label_16"
    caption="New Label"
    showIf="always">
  </xsf:button>
  <xsf:menu
    caption="Remove">
    <xsf:button
      action="xCollection::remove"
      xmlToEdit="CD_10"
      caption="CD"
```

```
    showIf="always">
</xsf:button>
<xsf:button
  action="xCollection::remove"
  xmlToEdit="Track_14"
  caption="Track"
  showIf="always"></xsf:button>
<xsf:button
  action="xOptional::remove"
  xmlToEdit="Label_16"
  caption="Label"
  showIf="always">
</xsf:button>
</xsf:menu>
</xsf:toolbar>
```

Notes

Adding menus to a toolbar is a feature of the .xsf file and is not available when using the **Commands** dialog box in design mode.

While InfoPath implements a single custom toolbar, you can add multiple toolbars to a form by creating additional **toolbar** elements within the .xsf file. Once you have created the additional toolbars by manually editing the .xsf file, those toolbars are available in the **Commands** dialog box, and you can then associate editing actions with them.

Custom task panes in the User Interface developer sample form

The User Interface developer sample form implements a [custom task pane](#) that is used to switch [views](#) and perform other general-purpose operations, such as sorting the list of CDs. Custom task panes are [.html](#) files that are displayed in the Microsoft Office InfoPath 2003 task pane as a user fills out a form. There can be only one custom task pane associated with a form.

To create a custom task pane, you must first create an [.html](#) file by using an HTML editor such as Microsoft FrontPage. You associate this [.html](#) file with a form by using the **Advanced** tab in the **Form Options** dialog box that is available on the **Tools** menu in [design mode](#). The **Advanced** tab allows you to add the [.html](#) file and any other supporting files, such as a cascading style sheet ([.css](#) file), using the Resource Manager. (The Resource Manager can also be accessed by clicking **Resource Manager** on the **Tools** menu in design mode.) After the [.html](#) file has been added as a resource, you can select the **Enable custom task pane** check box and set the name and location of your custom task pane.

Note You must add the [.html](#) file to the form using the Resource Manager before you can make the file a custom task pane.

When you add a custom task pane to a form, InfoPath creates entries in the [form definition \(.xsf\) file](#) using the [taskpane](#) element. The [caption](#) attribute of the [taskpane](#) element is used to store the name of the custom task pane, while the [href](#) attribute is used to store the [.html](#) file name. The following is a section from the [.xsf](#) file of the User Interface developer sample form that contains the [taskpane](#) element:

```
<xsf:taskpane
  caption="Custom Task Pane"
  href="taskpane.htm">
</xsf:taskpane>
```

In addition to using standard HTML markup, you can also use scripting code within the task pane that calls the InfoPath object model. In the

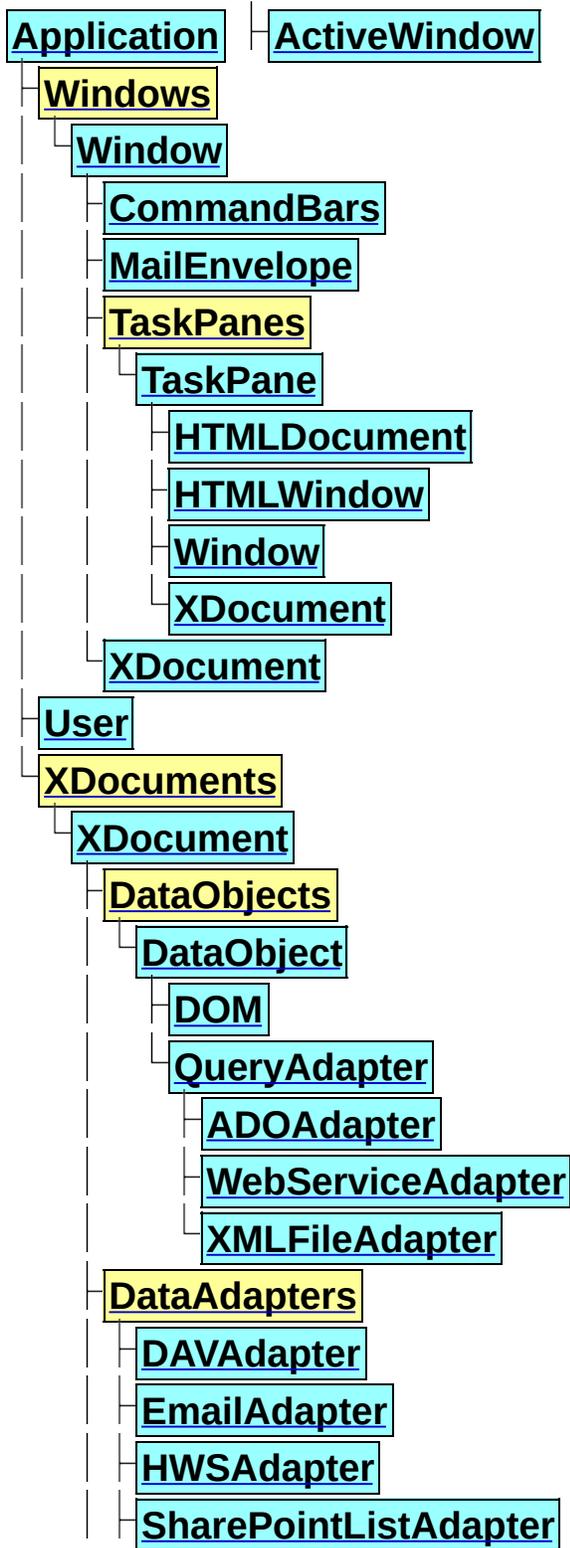
User Interface developer sample form, the [Extension](#) property of the [XDocument](#) object is used to gain access to the business logic functions contained in the form's primary script file.

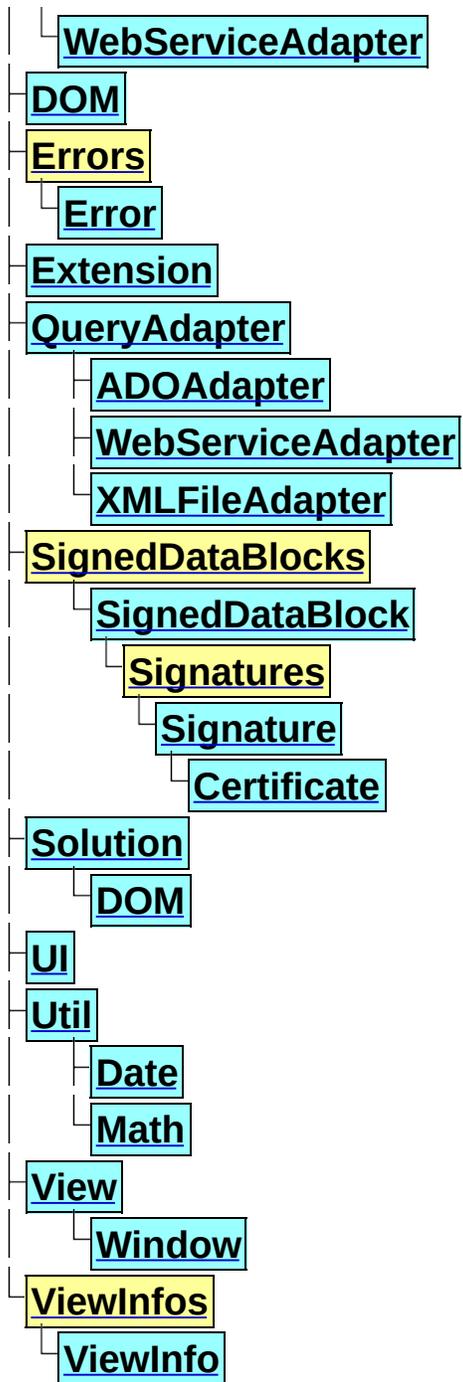
The following is a section from the .html file used as the custom task pane in the User Interface developer sample form. The call to the InfoPath object model's **Extension** property is used in the **onClick** event for the **Sort CDs** hyperlink. When the user clicks this link, the Sort function within the form's primary script file is called. The Sort function then takes the source XML document for the form and sorts the collection of CDs alphabetically according to artist and track.

```
<div class="action">
  <a href=""
    onClick="gobjXDocument.Extension.Sort();return false;">
    Sort CDs by artist/title</a>
</div>
```

Note To view the business logic functions for the User Interface developer sample form, you can open [Microsoft Script Editor \(MSE\)](#) in InfoPath design mode by clicking the **Tools** menu, pointing to **Script**, and clicking **Microsoft Script Editor**, or by pressing ALT+SHIFT+F11.

InfoPath Object Model Diagram





DataAdapters Collection

Contains a data adapter object corresponding to each data connection used within a Microsoft Office InfoPath 2003 form.

Remarks

Each data connection is used to retrieve data (inserted into the main data source or into a secondary data source) or to submit data.

A data connection used to retrieve data for the main data source will correspond to one of the following data adapter object types:

[WebServiceAdapter](#)

[ADOAdapter](#)

Note An **AdoAdapter** object used to retrieve data for the main data source can also submit data.

A data connection used to retrieve data for a secondary data source will correspond to one of the following data adapter object types:

[ADOAdapter](#)

[SharepointListAdapter](#)

[WebServiceAdapter](#)

[XMLFileAdapter](#)

A data connection used only for submitting data will correspond to one of the following data adapter object types:

[EmailAdapter](#)

[DAVAdapter](#)

[HWSAdapter](#)

[WebServiceAdapter](#)

The **DataAdapters** collection can be accessed using the [DataAdapters](#) property of the [XDocument](#) object.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service

Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

DataObjects Collection

DataObjects

DataObject

Contains a **DataObject** object for each [secondary data source](#) used within a Microsoft Office InfoPath 2003 form. Each **DataObject** object provides access to the particular data adapter object that was used to connect to the external data source.

Remarks

The **DataObjects** collection implements properties that can be used to access a form's associated **DataObject** objects, and it is accessible through the [DataObjects](#) property of the [XDocument](#) object.

Note The **DataObjects** collection can be used only to get the count of **DataObject** objects that it contains or to return a reference to a specified **DataObject** object. It cannot be used to create, add, or remove **DataObject** objects.

Using the DataObjects collection

In the following example, implemented as an OnClick [event handler](#) for a button on a form, the **DataObjects** property of the **XDocument** object is used to set a reference to the **DataObjects** collection. The code then loops through the collection and displays the positional index and name of each **DataObject** object that it contains:

```
function ShowDataObjectNames::OnClick(eventObj)
{
    // Set a reference to the DataObjects collection.
    var objDataObjects = XDocument.DataObjects;

    // Loop through the collection and display the name
    // of each DataObject object that it contains.
    for (i=0; i < objDataObjects.Count; i++)
    {
        XDocument.UI.Alert("Data object " + i + ": " +
            objDataObjects(i).Name);
    }
    objDataObjects = null;
}
```

For more information about using the **DataObjects** collection, see [Accessing external data sources](#).

Errors Collection



Contains an **Error** object for each error within a Microsoft Office InfoPath 2003 form. An **Error** object contains information about an InfoPath error, including its detailed message, short message, code, type, and the XML node that is associated with it.

Remarks

The **Errors** collection is a full-featured collection— it provides properties and methods for adding, deleting, and gaining access to the **Error** objects that it contains.

In addition to managing the errors generated by InfoPath, the **Errors** collection can also be used to create custom errors using the [Add](#) method.

Note Custom errors can also be created using the [ReportError](#) method of the [DataDOMEvent](#) object.

Using the Errors collection

The **Errors** collection is accessed through the [Errors](#) property of the [XDocument](#) object. The **Errors** collection is associated with a form's underlying XML document so that when an error occurs, it occurs within the XML document.

In the following example, the [Count](#) property of the **Errors** collection is used to get the count of errors currently contained in the collection:

```
XDocument.UI.Alert("Count of errors: " + XDocument.Errors.Count);
```

To delete a specific error from the **Errors** collection, use the [Delete](#) method, passing the XML code and condition name as arguments:

```
XDocument.Errors.Delete(myXMLNode, "MyErrorName");
```

To delete all of the errors in the **Errors** collection, use the [DeleteAll](#) method:

```
XDocument.Errors.DeleteAll();
```

To set a reference to an error within the **Errors** collection, use the [Item](#) property:

```
var objError;
```

```
objError = XDocument.Errors.Item(0);
```

```
// Or...
```

```
objError = XDocument.Errors(0);
```

For more information about using the **Errors** collection, see [Handling errors](#).

Signatures Collection

Contains a collection of [Signature](#) objects for each signature in the Microsoft Office InfoPath 2003 form or [SignedDataBlock](#) object.

Remarks

The **Signatures** collection implements properties and a method that can be used to access a form's associated **Signature** objects and to create a signature. It is accessible through the **SignedDataBlock** object.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the Signatures collection

When using the [Create](#) method, the signature is not written until the [Sign](#) method is called on the **Signature** object. These methods can be called only from the [OnSign](#) event handler.

SignedDataBlocks Collection

The collection of the [SignedDataBlock](#) objects in the [form template](#) as defined in the [form definition file \(.xsf\)](#).

Remarks

The **SignedDataBlocks** collection implements properties that can be used to access the **SignedDataBlock** objects associated with a form. The **SignedDataBlocks** collection is accessible through the [SignedDataBlocks](#) property of the [XDocument](#) object.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

TaskPanels Collection

TaskPanels

└ **TaskPane**

Contains a collection of **TaskPane** objects that represent the **task panes** associated with a window in Microsoft Office InfoPath 2003.

Remarks

The **TaskPanes** collection provides properties that can be used to access a collection of task pane objects, and it is accessed through the [TaskPanes](#) property of the [Window](#) object.

Note The **TaskPanes** collection can be used only to get the count of **TaskPane** objects that it contains and to return a reference to a specified **TaskPane** object. It cannot be used to add or remove **TaskPane** objects.

Using the TaskPanes collection

In the following example, the **TaskPanes** property of the **Window** object is used to set a reference to the **TaskPanes** collection:

```
var objTaskPanes;
```

```
objTaskPanes = XDocument.View.Window.TaskPanes;
```

In the following example, the **Item** property of the **TaskPanes** collection is used to set a reference to a specified **TaskPane** object. Then the code uses the **Visible** property of the **TaskPane** object to make the task pane visible.

```
var objTaskPane;
```

```
// Show the built-in Help task pane.
```

```
objTaskPane = XDocument.View.Window.TaskPanes(4);
```

```
objTaskPane.Visible = true;
```

Note The **Item** property argument is the type of task pane to return, based on the **Type** property of the **TaskPane** object, not the position of the **TaskPane** object in the **TaskPanes** collection.

ViewInfos Collection

ViewInfos

ViewInfo

Contains a [ViewInfo](#) object for each view within a Microsoft Office InfoPath 2003 form. **ViewInfo** objects contain descriptive information about the views that they are associated with.

Remarks

The **ViewInfos** collection implements properties that can be used to access a form's associated **ViewInfo** objects, and it is accessible through the [ViewInfos](#) property of the [XDocument](#) object.

Note The **ViewInfos** collection can be used only to get the count of **ViewInfo** objects that it contains or to return a reference to a specified **ViewInfo** object. It cannot be used to create, add, or remove **ViewInfo** objects.

Using the ViewInfos collection

In the following example, implemented as an `OnClick` [event handler](#) for a button on a form, the **ViewInfos** property of the **XDocument** object is used to set a reference to the **ViewInfos** collection. The code then loops through the collection and displays the positional index and name of each **ViewInfo** object that it contains.

```
function ShowViewNames::OnClick(eventObj)
{
    // Set a reference to the ViewInfos collection.
    var objViewInfos = XDocument.ViewInfos;

    // Loop through the collection and display the name
    // of each ViewInfo object that it contains.
    for (i=0; i < objViewInfos.Count; i++)
    {
        XDocument.UI.Alert("View name " + i + ": " +
            objViewInfos(i).Name);
    }
    objViewInfos = null;
}
```


Windows Collection

Windows

└ **Window**

Contains a **Window** object for each window within a Microsoft Office InfoPath 2003 form. **Window** objects represent the two types of windows that are used in the InfoPath application: the editing window that is used as the **form area** when a user fills out a form, and the designing window that is used as the **design mode** when a user designs a form.

Remarks

The **Windows** collection implements properties that can be used to access a form's associated **Window** objects, and it is accessible through the [Windows](#) property of the [XDocument](#) object.

Note The **Windows** collection can be used only to get the count of **Window** objects that it contains or to return a reference to a specified **Window** object. It cannot be used to create, add, or remove **Window** objects.

Using the Windows collection

In the following example, implemented as an `OnClick` [event handler](#) for a button on a form, the **Windows** property of the **Application** object is used to set a reference to the **Windows** collection. The code then loops through the collection and displays the type of each **Window** object that it contains.

```
function ShowWindowTypes::OnClick(eventObj)
{
    // Set a reference to the Windows collection.
    var objWindows = Application.Windows;
    var strWindowType;

    // Loop through the collection and display the type
    // of each Window object that it contains.
    for (i=0; i < objWindows.Count; i++)
    {
        switch (objWindows(i).Type)
        {
            case 0:
                strWindowType = "Editing window";
                break;
            case 1:
                strWindowType = "Designing window";
                break;
        }

        XDocument.UI.Alert("Window type " + i + ": " + strWindowType)
    }
    objWindows = null;
    strWindowType = null;
}
```

For more information about using the **Windows** collection, see [Working with form windows](#).

XDocuments Collection

XDocuments

XDocument

Contains an **XDocument** object for each Microsoft Office InfoPath 2003 form that is currently open. The **XDocument** object represents a form's underlying XML document and can be used to interact with the XML data that a form contains.

Remarks

The **XDocuments** collection implements a number of properties and methods that can be used to access a form's associated **XDocument** object, or to create and open the forms themselves. The **XDocuments** collection is accessed through the [XDocuments](#) property of the [Application](#) object.

Using the XDocuments collection

In the following example, the [Open](#) method of the **XDocuments** collection is used to open an existing form:

```
Application.XDocuments.Open("C:\\MyForm.xml");
```

You can also create a new form based on an existing form using the [New](#) method:

```
Application.XDocuments.New("C:\\MyForm.xml");
```

To access an **XDocument** object contained in the **XDocuments** collection, you can pass the positional index or the location path to the [Item](#) method:

```
var objXDoc;
```

```
objXDoc = Application.XDocuments(0);
```

```
// or...
```

```
objXDoc = Application.XDocuments("C:\\MyForm.xml");
```

For more information about using the **XDocuments** collection, see [Accessing form data](#).

XMLNodes Collection

XMLNodes

Contains a collection of XML [Document Object Model \(DOM\)](#) nodes. The **XMLNodes** collection is a general-purpose collection that is used by a number of Microsoft Office InfoPath 2003 methods.

Remarks

The **XMLNodes** collection provides properties that can be used to access a collection of XML DOM nodes, and it is returned by both the [GetSelectedNodes](#) and [GetContextNodes](#) methods of the [View](#) object.

After you have set a reference to one of the XML DOM node objects that the **XMLNodes** collection contains, you can use any of the properties and methods that the XML DOM provides for interacting with an XML node object.

Note To learn more about the XML DOM and all of the properties and methods that it supports, see the MSXML 5.0 SDK documentation in the [Microsoft Script Editor \(MSE\)](#) Help system.

Using the XMLNodes collection

In the following example, a reference is set to a collection of XML DOM nodes returned by the **GetSelectedNodes** method of the **View** object. Then the code displays the name and source XML of the first node found in the collection using a message box:

```
var objXMLNodes;  
  
objXMLNodes = XDocument.View.GetSelectedNodes();  
  
if (objXMLNodes.Count > 0)  
{  
    XDocument.UI.Alert(objXMLNodes(0).nodeName + "\n\n" + objXMLNodes(0).sourceXML);  
}
```


ADOAdapter Object

ADOAdapter

Represents a connection to a Microsoft ActiveX Data Objects/OLEDB data source. The **ADOAdapter** object is a type of Microsoft Office InfoPath 2003 [data adapter](#) that contains all the information necessary for retrieving and submitting data to an external data source.

Remarks

For [secondary data sources](#), the **ADOAdapter** object provides properties that can be used to get and set information about the data adapter's connection string, SQL command text, and timeout value. It also provides a method for creating an SQL command text fragment based on a specified XML node.

If an ADO/OLEDB data source is used as the primary data source for a form, the **ADOAdapter** object is accessible through the [QueryAdapter](#) property of the [XDocument](#) object.

Note The **ADOAdapter** object is limited to work only with Microsoft SQL Server and Microsoft Access databases.

Using the ADOAdapter object

The **ADOAdapter** object is accessible through the [QueryAdapter](#) property of the [DataObject](#) object, and **DataObject** objects are accessible through the [DataObjects](#) property of the **XDocument** object.

In the following example, the [Timeout](#) property of the **ADOAdapter** object is used to set the timeout for a particular query operation, returning the timeout to its original value after the query operation is complete:

```
function RunLongQuery()
{

    var objADOAdapter;
    var intTimeout;

    // Set a reference to the ADOAdapter object.
    objADOAdapter = XDocument.DataObjects("CityDropDownList").(

    // Save the original timeout value.
    intTimeout = objADOAdapter.Timeout;

    // Set a longer timeout value and then run the query.
    objADOAdapter.Timeout = 60;
    XDocument.DataObjects("CityDropDownList").Query();

    // Restore the original timeout value.
    objADOAdapter.Timeout = intTimeout;

    objADOAdapter = null;
    intTimeout = null;
}
```

For more information about using the **ADOAdapter** object, see

[Accessing external data sources.](#)

Application Object

Application

Represents the Microsoft Office InfoPath 2003 application. The **Application** object includes properties and methods that return references to the high-level objects of the InfoPath object model. For example, the [XDocuments](#) property returns a reference to a collection of [XDocument](#) objects.

Remarks

The **Application** object is the top-level object in the InfoPath object model, and it provides a number of properties and methods that can be used to access lower-level collections and objects in the object model, and to perform a variety of general purpose functions.

Using the Application object

The **Application** object can be used directly in scripting code as in the following Microsoft **JScript** example, which uses the **Alert** method of the **UI** object, accessed through the **XDocument** object, to display a message box that indicates the version number of the current instance of InfoPath:

```
XDocument.UI.Alert("Application version: " + Application.Version);
```

Note that the **XDocument** property was not qualified with the name of the **Application** object. This is because both the **Application** object and the **XDocument** object are embedded directly in the InfoPath script engine, and therefore they do not have to be explicitly declared. However, they do need to be declared when used in an expression or argument, as the example above demonstrates.

Note The **Application** object contains the following methods that can be used within InfoPath:

| Method | Description |
|--|--|
| FormatString | Formats the specified string or XML node according to the specified category and options parameters. |
| IsDestinationReachable | Returns a Boolean value indicating whether the specified Uniform Resource Locator (URL), universal naming convention (UNC) path, or IP address of the destination computer can be connected to from the client computer. |
| NewADODBConnection | Creates and returns a reference to an empty ActiveX Data Objects (ADO) Connection object. |
| NewADODBRecordset | Creates and returns a reference to an empty ActiveX Data Objects (ADO) Recordset object. |

Note The **Application** object contains the following methods that can be

used for external automation:

| Method | Description |
|------------------------------------|---|
| CacheSolution | Examines the form template in the cache and, if necessary, updates it from the published location of the form template. |
| Quit | Quits the Microsoft Office InfoPath 2003 application. |
| RegisterSolution | Installs the specified Microsoft Office InfoPath form template. |
| UnregisterSolution | Uninstalls the specified Microsoft Office InfoPath 2003 form template. |

The **XDocuments** collection contains the following additional methods that can be used for external automation:

| Method | Description |
|---|---|
| Close | Closes the specified Microsoft Office InfoPath 2003 form. |
| New | Creates a new Microsoft Office InfoPath 2003 form based on the specified form. |
| NewFromSolution | Creates a new Microsoft Office InfoPath 2003 form based on the specified form template. |
| NewFromSolutionWithData | Creates a new Microsoft Office InfoPath 2003 form using the specified XML data and form template. |
| Open | Opens the specified Microsoft Office InfoPath 2003 form. |

Although the [ExternalApplication](#) object is deprecated in Microsoft Office InfoPath 2003 Service Pack 1, replaced with the methods in the above tables, **ExternalApplication** methods are still available for backward compatibility.

For more information about using the **Application** object, see [Accessing application data](#).

Button Object

The **Button** object is the Microsoft Office InfoPath 2003 object that is used to implement the [OnClick](#) event that is associated with a button on an InfoPath form. This object cannot be used directly in code.

Certificate Object

Represents the X.509 [digital certificate](#) that has been used to create a signature.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the Certificate object

Use the [Certificate](#) property of the [Signature](#) object to return a **Certificate** object.

The **Certificate** object contains the following properties, which can be used to return information about a digital certificate:

| Property | Description |
|--------------------------------|---|
| IssuedTo | Returns to whom the digital certificate is issued. |
| IssuedBy | Returns the issuer of the digital certificate. |
| ExpirationDate | Returns the expiration date of the digital certificate. |
| Status | Returns the status of the digital certificate. |

DataDOM Object

DataDOM

The **DataDOM** object is the Microsoft Office InfoPath 2003 object that is used to implement the [data validation](#) events that are associated with the [XDocument](#) object. This object cannot be used directly in scripting code.

DataDOMEvent Object

DataDOMEvent

An event object that is used during Microsoft Office InfoPath 2003 [data validation](#) events. The **DataDOMEvent** object provides a number of properties and a method that can be used within a data validation event to programmatically interact with the data that is being validated and to provide a response if the data is not valid.

Remarks

The **DataDOMEvent** object is passed as a parameter to the [OnBeforeChange](#), [OnValidate](#), and [OnAfterChange](#) events.

Note The **OnValidate** event can occur without a change in the form's underlying XML document.

The **DataDOMEvent** object is used to get information about the XML [Document Object Model \(DOM\)](#) node that is being changed, and it also provides a property for getting a reference to a form's underlying XML document. In addition, it provides properties for handling the change in data, including rejecting the change and creating an error message.

Note The **DataDOMEvent** object is passed as an argument to one of the data validation event handlers. Its properties and method are only available during the event that it is passed to.

Using the DataDOMEEvent object

In the following example from the Data Validation developer sample form, the **DataDOMEEvent** object is used to check the value of the node using the **Site** property. If the data validation fails, the **ReportError** method is used to create a custom error.

```
function msoxd__itemB_quantityListB::OnValidate(eventObj)
{
  if (parseInt(eventObj.Site.nodeTypeValue, 10) > 50)
    eventObj.ReportError(eventObj.Site, "Invalid quantity. " +
      "The total number of each type of block cannot exceed 50.", false)

  if (parseInt(eventObj.Site.nodeTypeValue, 10) < 0)
    eventObj.ReportError(eventObj.Site, "Invalid quantity. " +
      "The total number of each type of block cannot be less than 0.", false)
}
```

For more information about using the **DataDOMEEvent** object, see [Responding to form events](#).

DataObject Object

DataObjects | **DataObject**

Represents a link to the data adapter of a [secondary data source](#). The **DataObject** object acts as an intermediary between a Microsoft Office InfoPath 2003 form and the [data adapter](#) object that is used to access data that is contained in an external data source.

The **DataObject** object provides properties and methods that can be used to programmatically interact with data adapter objects, including retrieving information about the data adapter objects and accessing the data that they connect to. The **DataObject** object is accessible through the [DataObjects](#) property of the [XDocument](#) object.

Remarks

An external data source can take the form of a Microsoft Access or Microsoft SQL Server database, an .xml file, or an XML Web service. The type of data adapter object used to access the external data source depends on the type of data source. The **DataObject** object provides a common set of properties and a method that can be used for all types of data adapter objects, and each of the data adapter objects provides its own set of properties and methods.

Microsoft Office InfoPath 2003 supports three types of data adapters:

1. **ActiveX Data Objects** Represented by the [ADOAdapter](#) object.
2. **Web services** Represented by the [WebServiceAdapter](#) object.
3. **XML file** Represented by the [XMLFileAdapter](#) object.

Using the DataObject object

In the following code sample, the name of the secondary data source is passed to the [Item](#) property of the [DataObjects](#) collection, which returns a reference to the **DataObject** object, which, in this case, is associated with an **ADOAdapter** data adapter object. Using the [QueryAdapter](#) property of the **DataObject** object, the **ADOAdapter** data adapter object's [Connection](#) property is used to display the ADO connection string in a message box.

```
function TestDataObjects()
{
    var objDataObject;

    // Set a reference to the specified data object.
    objDataObject = XDocument.DataObjects("CityList");

    // Display the connection information for the ADOAdapter object.
    XDocument.UI.Alert("Data Adapter: " + objDataObject.QueryAdapt

    objDataObject = null;
}
```

For more information about using the **DataObject** object, see [Accessing external data sources](#).

Date Object

Provides several date-related methods that can be used in Microsoft Office InfoPath 2003 [forms](#).

Remarks

Use the [Date](#) property of the [Util](#) object to return a **Date** object.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the Date object

The **Date** object contains the following methods:

| Method | Description |
|-----------------------|--|
| Now | Returns a Variant that represents the current system date and time in ISO format. |
| Today | Returns a Variant containing the current system date in ISO format. |

DAVAdapter Object

Represents a connection to submit form information to a Microsoft Windows SharePoint Services server, or other servers that support DAV connections.

Remarks

Use the [Item](#) property of the [DataAdapters](#) collection to return a **DAVAdapter** object.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the DAVAdapter object

The **DAVAdapter** object contains the following properties:

| Property | Description |
|-------------------------------|---|
| FileName | Returns or sets the file name for the current Microsoft Office InfoPath 2003 form when it is submitted by the DAVAdapter object. |
| FolderURL | Returns or sets the the Uniform Resource Locator (URL) to which the form will be submitted by the DAVAdapter object. |
| Name | Returns the name of a DAVAdapter object. |
| QueryAllowed | Corresponds to the queryAllowed attribute in the form definition file (.xsf) . Always returns False for the DAVAdapter object. |
| SubmitAllowed | Corresponds to the submitAllowed attribute in the form definition file (.xsf). Always returns True for the DAVAdapter object. |

The **DAVAdapter** object contains the following methods:

| Method | Description |
|----------------------------|---|
| Query | Because the DAVAdapter object is available for submitting data only, this method will always generate a run-time error when it is called on that object. |
| Submit | Executes the submit operation on the associated adapter. |
| SubmitData | Submits the specified DOM element or DOM document to a data adapter. |

DocActionEvent Object

DocActionEvent

An event object that is used during a Microsoft Office InfoPath 2003 button click event. The **DocActionEvent** object provides a number of properties that can be used within a button click event to programmatically interact with the data in a form's underlying XML document and to control the success or failure of the event.

Remarks

The **DocActionEvent** object is passed as a parameter to the [OnClick](#) event button that is contained in a view of an InfoPath form. Its properties are available only during the **OnClick** event.

Using the DocActionEvent object

In the following example, the [Source](#) property of the **DocActionEvent** object is used to display the source XML data of the inner-most XML [Document Object Model \(DOM\)](#) node of the form's underlying XML document, which contains the button:

```
function ShowDocActionEventSource::OnClick(eventObj)
{
    XDocument.UI.Alert("Source: " + eventObj.Source.xml);
}
```

For more information about using the **DocActionEvent** object, see [Responding to form events](#).

DocContextChangeEvent Object

An event object that is used during a Microsoft Office InfoPath 2003 context change event. The **DocContextChangeEvent** object provides a number of properties that can be used within a context change to programmatically interact with the data in a form's underlying XML document, to provide contextual feedback to the user, or to perform actions for the user.

Remarks

The **DocContextChangeEvent** object is passed as a parameter to the [OnContextChange](#) event.

The **DocContextChangeEvent** object is used to get information about the XML Document Object Model (DOM) node that is the current context of the form's underlying XML document. In addition, it provides information about the type of context change and whether the change happened in response to an undo or redo operation performed by the user.

As described in the [OnContextChange](#) event topic, the **Type** property returns only the value "ContextNode" for context changes in Microsoft InfoPath 2003 Service Pack 1. Nevertheless, if code in an event handler performs actions that depend on current functionality, that code should still be designed to check the value of the **Type** property, because future versions of InfoPath may use different values for different context changes.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the DocContextChangeEvent object

When the [IsUndoRedo](#) property of the **DocContextChangeEvent** object is **True**, the context change was caused by an undo or redo operation rather than an explicit user context change. Operations performed in the **OnContextChange** event handler that modify the XML DOM should be avoided in response to undo or redo actions, because they may interfere with the user's intention to revert data to a previous state.

Example

In the following example, a node named lastChanged is updated in response to context changes:

```
function XDocument::OnContextChange(eventObj)
{
    if ( eventObj.Type == "ContextNode" && !eventObj.IsUndoRedo )
    {
        var oContextNode = eventObj.Context;
        var oLastChangedNode =
            XDocument.DOM.selectSingleNode("my:lastChanged");

        oLastChangedNode.text = oContextNode.nodeName;
    }
}
```

DocEvent Object

DocEvent

An event object that is used during a Microsoft Office InfoPath 2003 merge or view switching event. The **DocEvent** object provides the [XDocument](#) property that can be used within a merge or view switching event to programmatically interact with the data in a form's underlying XML document.

Remarks

The **DocEvent** object is passed as a parameter to the [OnSwitchView](#) and [OnAfterChange](#) events of an InfoPath form. The **XDocument** property that it provides is available only during these events.

Using the DocEvent object

In the following example, the **XDocument** property of the **DocEvent** object is used to display the source XML of a form's underlying XML document using the [DOM](#) property of the **XDocument** object:

```
function XDocument::OnSwitchView(eventObj)
{
    XDocument.UI.Alert("The source XML: " + eventObj.XDocument.I
}
```

For more information about using the **DocEvent** object, see [Responding to form events](#).

DocReturnEvent Object

DocReturnEvent

An event object that is used during a Microsoft Office InfoPath 2003 load or submission event. The **DocReturnEvent** object provides the [XDocument](#) property that can be used within a load or submission event to programmatically interact with the data in a form's underlying XML document. It also provides the [ReturnStatus](#) property that is used to specify whether the event is successful.

Remarks

The **DocReturnEvent** object is passed as a parameter to the [OnLoad](#) and [OnSubmitRequest](#) events of an InfoPath form. The properties that it provides are available only during these events.

Using the DocReturnEvent object

In the following example, the **XDocument** property of the **DocReturnEvent** object is used to display the source XML of a form's underlying XML document using the [DOM](#) property of the **XDocument** object:

```
function XDocument::OnLoad(eventObj)
{
    XDocument.UI.Alert("The source XML: " + eventObj.XDocument.I
    eventObj.ReturnStatus = true;
}
```

For more information about using the **DocReturnEvent** object, see [Responding to form events](#).

EmailAdapter Object

Represents the [data adapter](#) for submitting form information in e-mail by using Microsoft Office Outlook.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the EmailAdapter object

Use the [Item](#) property of the [DataAdapters](#) collection to return an **EmailAdapter** object.

In the following example, a reference to the **EmailAdapter** object is set by passing the name of the **EmailAdapter** object to the **Item** property of the **DataAdapters** collection:

```
var objEmailAdapter;  
objEmailAdapter = XDocument.DataAdapters("Main Submit");
```

After the reference has been set, you can use the properties of the **EmailAdapter** object as shown in the following example, which sets the **To** properties and the **Subject** property, then submits the form.

```
objEmailAdapter.To = "list@example.com";  
objEmailAdapter.Subject = "Status Report";  
objEmailAdapter.Submit();
```

The **EmailAdapter** object contains the following properties:

| Property | Description |
|------------------------------------|---|
| AttachmentFileName | Returns or sets the file name to be used for the current form when attached to the e-mail message for an EmailAdapter object. |
| BCC | Returns or sets the BCC recipients for an EmailAdapter object. |
| CC | Returns or sets the CC recipients for an EmailAdapter object. |
| Intro | Returns or sets the introduction in the body of the e-mail message for an EmailAdapter object. |
| Name | Returns the name of an EmailAdapter object. |
| QueryAllowed | Corresponds to the queryAllowed attribute in the form definition file (.xsf) . Always returns False for the EmailAdapter object. |

| | |
|--------------------------------------|--|
| <u>Subject</u> | Returns or sets the subject of the e-mail message for the specified EmailAdapter object. |
| <u>SubmitAllowed</u> | Corresponds to the submitAllowed attribute in the form definition file (.xsf). Always returns True for the EmailAdapter object. |
| <u>To</u> | Returns or sets the To recipients for the specified EmailAdapter object. |

The **EmailAdapter** object contains the following methods:

| Method | Description |
|-----------------------------------|---|
| <u>Query</u> | Because the EmailAdapter object is available for submitting data only, this method will always generate a run-time error when it is called on that object. |
| <u>Submit</u> | Executes the submit operation on the associated data adapter. |
| <u>SubmitData</u> | Submits the specified DOM element or DOM document to a data adapter. |

Error Object



Represents an error in a Microsoft Office InfoPath 2003 form. Each **Error** object in InfoPath is each associated with an XML [Document Object Model \(DOM\)](#) node from a form's underlying XML document. When [data validation](#) fails for a particular XML DOM node, InfoPath creates an **Error** object and places it in the [Errors](#) collection.

Remarks

That are three types of data validation errors that can occur in an InfoPath form:

SCHEMA_VALIDATION Data validation failed as a result of an [XML Schema](#)-defined constraint.

SYSTEM_GENERATED Data validation failed as a result of constraints defined in the [form definition \(.xsf\) file](#) or as a result of scripting code calling the [ReportError](#) method of the [DataDOMEvent](#) object.

USER_SPECIFIED Data validation failed as a result of a custom scripting error using the [Add](#) method of the **Errors** collection.

Note The [Type](#) property of the **Error** object can be used to determine the type of data validation error that has occurred.

Using the Error Object

The **Error** object provides a number of properties that can be used to programmatically interact with an InfoPath data validation error. The **Error** object is accessed through the [Errors](#) property of the [XDocument](#) object, which returns a reference to the **Errors** collection.

In the following example, the [Item](#) property of the **Errors** collection is used to set a reference to an **Error** object; then the [ShortErrorMessage](#) property of the **Error** object is used to display the error message in a message box:

```
var objError;
```

```
objError = XDocument.Errors(0);
```

```
XDocument.UI.Alert("Error message: " + objError.ShortErrorMessage
```

For more information about using the **Error** object, see [Handling errors](#).

ExternalApplication Object

ExternalApplication

Represents the Microsoft Office InfoPath 2003 application. The **ExternalApplication** object implements a limited set of methods that can be used for automating InfoPath by an external Component Object Model (COM)–based programming language.

Remarks

The **ExternalApplication** object can be used to perform a limited set of InfoPath operations such as creating, opening, or closing a [form](#); registering or unregistering a [form template](#); or simply quitting the application.

Using the ExternalApplication object

To use the **ExternalApplication** object, you must first create a reference to it using the ProgID of the InfoPath object model along with the name of the **ExternalApplication** object. The following example demonstrates creating a reference to the **ExternalApplication** object using the [Visual Basic for Applications \(VBA\)](#) programming language:

```
Dim objIP As Object
Set objIP = CreateObject("InfoPath.ExternalApplication")
```

Note This example uses late-binding for creating the reference to the **ExternalApplication** object; you can also use early-binding by setting a reference to the InfoPath object model in your programming environment.

After you have created a reference to the **ExternalApplication** object, you can then use the methods that it provides to interact with InfoPath. In the following example, written in VBA, the [Open](#) method of the **ExternalApplication** object is used to open a form based on the specified [Uniform Resource Identifier \(URI\)](#):

```
Public Sub OpenForm()

    Dim objIP As Object

    'Create the ExternalApplication object and open a specified form.
    Set objIP = CreateObject("InfoPath.ExternalApplication")
    objIP.Open ("C:\My Forms\Form1.xml")

    Set objIP = Nothing

End Sub
```

For more information about using the **ExternalApplication** object, see [Automating InfoPath](#).

HTMLTaskPane Object

HTMLTaskPane

Represents a [custom task pane](#) in a Microsoft Office InfoPath 2003 form that is associated with a window. The **HTMLTaskPane** object provides a number of properties and methods for working with the InfoPath custom task pane, and it provides properties and methods to the [TaskPane](#) object as an inherited object.

Remarks

The properties and methods that are available for an InfoPath task pane are determined by the type of task pane that you are working with. If the **Type** property of the **TaskPane** object returns 0, the task pane is a custom task pane and the properties and methods that are available are provided by the **HTMLTaskPane** object. If the **Type** property returns any other value, the task pane is a built-in task pane and the properties are provided by the **TaskPane** object directly.

The **Type** property is based on the [XdTaskPaneType](#) enumeration. These enumerated values are also used as arguments to the [Item](#) property of the [TaskPanes](#) collection for returning a reference to a specified type of task pane.

Note The properties and methods of the **HTMLTaskPane** object cannot be called during an [OnLoad](#) event because the view is not yet loaded when this event occurs, and task panes are associated with the view.

Using the HTMLTaskPane object

In the following example from the User Interface developer sample form, the **Item** property of the **TaskPanes** collection is used to set a global reference to the **TaskPane** object that represents the custom task pane. The code then calls a scripting function defined in the **HTML** code of the custom task pane using the **HTMLDocument** property of the **HTMLTaskPane** object, which is inherited by the **TaskPane** object.

```
function SetTaskPaneState()
{
    // Ensure View has loaded before trying to access the task pane.
    if (XDocument.View)
    {
        // Get a reference to the custom task pane. It is always the 0-th
        // task pane in the TaskPanes collection.
        if (gobjTaskPane == null)
            gobjTaskPane = XDocument.View.Window.TaskPanes.Item(0);

        // Ensure that the task pane is completely loaded.
        if (gobjTaskPane.HTMLDocument.readyState == "complete")
        {
            var strTaskPaneViewId = "TP_" + XDocument.View.Name.repla

            // Call a script function defined in the task pane HTML page.
            gobjTaskPane.HTMLDocument.parentWindow.SelectView(strTa
        }
    }
}
```


HTMLTaskPaneExternal Object

HTMLTaskPaneExternal

Represents a link to the Microsoft Office InfoPath 2003 object model. The **HTMLTaskPaneExternal** object is used to expose the InfoPath object model to the [Dynamic HTML \(DHTML\)](#) scripting code in a [custom task pane](#).

The **HTMLTaskPaneExternal** object provides a number of properties for accessing certain parts of the InfoPath object model, including the [XDocument](#) object and the [Window](#) object.

Remarks

The **HTMLTaskPaneExternal** object facilitates using the InfoPath object model within the scripting code that is part of a custom task pane. The object is exposed through the **external** property of the DHTML **window** object.

Using the HTMLTaskPaneExternal object

In the following example, the **HTMLTaskPaneExternal** object is used through the **external** property of the DHTML **window** object to set a reference to the **XDocument** object that is part of the InfoPath object model:

```
var objXDoc;
```

```
objXDoc = window.external.XDocument;  
objXDoc.View.SwitchView("View2");
```


HWSAdapter Object

Represents a connection to submit form information to a Microsoft Biztalk 2004 HWS (Human Workflow Services) server.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the HWSAdapter object

The **HWSAdapter** object contains the following properties:

| Property | Description |
|-------------------------------|---|
| Name | Returns the name of an HWSAdapter object. |
| QueryAllowed | Corresponds to the queryAllowed attribute in the form definition file (.xsf) . Always returns False for the HWSAdapter object. |
| SubmitAllowed | Corresponds to the submitAllowed attribute in the form definition file (.xsf). Always returns True for the HWSAdapter object. |

The **HWSAdapter** object contains the following methods:

| Method | Description |
|------------------------|---|
| Query | Because the HWSAdapter object is available for submitting data only, this method will always generate a run-time error when it is called on that object. |
| Submit | Executes the submit operation on the associated adapter. |

InfoPathControl Object

Defines methods used from the implementation of an ActiveX control for initializing, uninitializing, enabling, disabling, and saving the state of a control.

Remarks

The **InfoPathControl** and [InfoPathControlSite](#) objects and their methods and properties are designed to be used only from the implementation of an ActiveX control. These objects and their members are not supported in InfoPath form code. For more information on how to create ActiveX controls that work with InfoPath, see the [InfoPath Developer Center](#).

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form template that contains a view with an ActiveX control that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the InfoPathControl Object

The **InfoPathControl** object defines the following methods that must be implemented by the developer of an ActiveX control for use on InfoPath forms.

| Method | Description |
|----------------------------------|---|
| <u>Enable</u> | Defines a method that InfoPath calls when it needs to enable or disable an instance of the control in a view. |
| <u>Init</u> | Defines a method that performs any initialization routines required when an instance of the control is added to an InfoPath form. |
| <u>SaveState</u> | Defines a method that InfoPath calls when it needs to save the state of an instance of the control in a view. |
| <u>Uninit</u> | Defines a method that performs any clean up routines required before an instance of the control is removed from a view. |

InfoPathControlSite Object

Represents the object that InfoPath passes to an ActiveX control when it is initialized in an InfoPath view. The **InfoPathControlSite** object defines the [XDocument](#) property for accessing the [XDocument](#) object associated with a form, and the [Node](#) property for accessing the XML node to which the control is bound.

Remarks

The **InfoPathControlSite** object should not be implemented by the developer of an ActiveX control. It represents an interface that is implemented by InfoPath itself.

The [InfoPathControl](#) and **InfoPathControlSite** objects and their methods and properties are designed to be used only from the implementation of an ActiveX control. These objects and their members are not supported in InfoPath form code. For more information on how to create ActiveX controls that work with InfoPath, see the [InfoPath Developer Center](#).

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form template that contains a view with an ActiveX control that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

MailEnvelope Object

MailEnvelope

Represents a custom e-mail message in a Microsoft Office InfoPath 2003 form. The **MailEnvelope** object provides a number of properties that can be used to programmatically create a custom e-mail message within the default e-mail editor, and it attaches the currently open InfoPath form to the message.

Remarks

After the e-mail message is created, the default e-mail editor will display the e-mail message; users can then inspect and edit the e-mail message before sending it.

Note The **MailEnvelope** object cannot be used to send the e-mail messages it creates; users must manually send the e-mail messages.

You can also use the [ShowMailItem](#) method of the [UI](#) object to programmatically create e-mail messages.

Using the MailEnvelope object

The **MailEnvelope** object is accessed through the **MailEnvelope** property of the **Window** object.

In the following example from the Meeting Agenda sample form, the **MailEnvelope** property of the **Window** object is used to set a reference to the **MailEnvelope** object that is associated with the currently active window. The code then uses the **MailEnvelope** object to create a custom e-mail message.

```
function SendMeetingAgendaBtn::OnClick(oEvent)
{
    var rgRecipients = new Array();
    var xmlRecipients = getNodeList("/mtg:meetingAgenda/
        mtg:attendees/mtg:attendee/mtg:emailAddressPrimary");
    var xmlRecipient;

    while (xmlRecipient = xmlRecipients.nextNode())
        rgRecipients.push(xmlRecipient.text);

    try
    {
        var oEnvelope = Application.ActiveWindow.MailEnvelope;

        oEnvelope.Subject = getNode("/mtg:meetingAgenda/mtg:subject")
        oEnvelope.To = rgRecipients.join("; ");
        oEnvelope.Visible = true;
    }
    catch(ex)
    {
        XDocument.UI.Alert(ex.description);
    }
}
```


Math Object

Provides several math-related methods that can be used in Microsoft Office InfoPath 2003 [forms](#).

Remarks

Use the [Math](#) property of the [Util](#) object to return a **Math** object.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the Math object

The **Math** object contains the following methods:

| Function | Description |
|-----------------------------|--|
| <u>Avg</u> | Returns the average value of all the numerical elements in a node set. |
| <u>Eval</u> | Returns the set of results calculated when the expression is applied to each set of elements in the context. |
| <u>Max</u> | Returns the largest value of all the numerical elements in a node set. |
| <u>Min</u> | Returns the smallest value of all the numerical elements in a node set. |
| <u>Nz</u> | Replaces empty values in the specified node list with "0" (zero). |

MergeEvent Object

An event object that is used during an [OnMergeRequest](#) event. The **MergeEvent** object provides properties and methods that can be used during an **OnMergeRequest** event to programmatically interact with a form's underlying XML document and to determine merge properties such as the number of files being merged.

Remarks

During a single merge forms operation, multiple **OnMergeRequest** events will occur. One **OnMergeRequest** event will occur for each file being merged.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the MergeEvent object

The **MergeEvent** object is passed as a parameter to the **OnMergeRequest** event of an InfoPath form. The properties that it provides are available only during this event.

Example

In the following example, the [DOM](#) property and [ReturnStatus](#) property of the **MergeEvent** object and the [ImportDOM](#) method of the [XDocument](#) object are used to import (merge) a form from the **OnMergeRequest** event handler:

```
function XDocument::OnMergeRequest(eventObj)
{
    XDocument.ImportDOM(eventObj.DOM);

    eventObj.ReturnStatus = true;
}
```

SaveEvent Object

An event object that is used during an [OnSaveRequest](#) event. The **SaveEvent** object provides a number of properties and methods that can be used during a save operation from the **OnSaveRequest** event handler to programmatically interact with a form's underlying XML document, determine save properties, and perform the save operation.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the SaveEvents object

The **SaveEvent** object is passed as a parameter to the **OnSaveRequest** event of an InfoPath form. The properties that it provides are available only during this event. This object expires when the save event expires so it may not be stored and used outside of the scope of the save event.

Example

In the following example, the [ReturnStatus](#) property and the [PerformSaveOperation](#) method of the **SaveEvent** object are used to perform a standard InfoPath save operation:

```
function XDocument::OnSaveRequest(eventObj)
{
    eventObj.PerformSaveOperation();
    eventObj.ReturnStatus = true;
}
```


SharepointListAdapter Object

Represents a connection to a SharePoint list or document library. The **SharePointListAdapter** object represents the Office InfoPath 2003 [data adapter](#) for retrieving data from a SharePoint list or document library.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the SharepointListAdapter Object

For a [secondary data source](#), the **SharePointListAdapter** object is accessible through the **QueryAdapter** property of a data adapter object. Data adapter objects are accessible through the [DataAdapters](#) property of the [XDocument](#) object.

The **SharepointListAdapter** object contains the following properties:

| Property | Description |
|-------------------------------|--|
| Name | Returns the name of the SharepointListAdapter object. |
| QueryAllowed | A read-only property of type Boolean that corresponds to the queryAllowed attribute in the form definition file (.xsf) . Always returns True for the SharepointListAdapter object. |
| SiteUrl | A read-only property returning the Uniform Resource Locator (URL) of the SharePoint site that this adapter can query from. |
| SubmitAllowed | Corresponds to the submitAllowed attribute in the form definition file (.xsf). Always returns False for the SharepointListAdapter object. |

The **SharepointListAdapter** object contains the following methods:

| Method | Description |
|------------------------|---|
| Query | Reads data from the SharepointListAdapter object. |
| Submit | Because the SharepointListAdapter object is available for receiving data only, this method will always generate a run-time error when it is called on that object. |

Example

In the following example, a reference to the **SharePointListAdapter** object is accessed through the **QueryAdapter** property of a data adapter object by passing the name of the data adapter object to the **Item** property of the [DataAdapters](#) collection:

```
var objSPLAdapter;  
objSPLAdapter = XDocument.DataAdapters("Announcements").Query
```

After the reference has been set, you can use the methods of the **SharePointListAdapter** object as shown in the following example, which requeries the SharePoint list or library to update the **DOM** property of the data adapter object:

```
objSPLAdapter.Query();
```


Signature Object

Represents a digital signature that has been added to a [form](#) or set of signed data in a form.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the Signature object

Use the [Item](#) property of the [Signatures](#) collection to return a **Signature** object.

The **Signature** object contains the following properties, which can be used to return information about a digital signature:

| Property | Description |
|---------------------------------------|--|
| Comment | Returns the text comment that was added to the digital signature. |
| Status | Returns the status of the specified digital signature |
| SignatureBlockXmlNode | Returns the XML node corresponding the digital signature. |
| Certificate | Returns the Certificate object for the X.509 digital certificate that was used when signing a form or a set of signed data. |

The **Signature** object contains the following method:

| Method | Description |
|----------------------|---|
| Sign | Writes the XML digital signature block and computes the cryptographic hash for the signed data. |

SignedDataBlock Object

Represents a set of signed data in a Microsoft Office InfoPath 2003 [form](#). The **SignedDataBlock** object provides a number of properties and one method that can be used to programmatically interact with a set of signed data.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the SignedDataBlock object

The **SignedDataBlock** object contains the following read-only properties:

| Property | Description |
|--|--|
| XPath | Returns the XPath expression of the set of signed data represented by the SignedDataBlock object. |
| Name | Returns the name of the SignedDataBlock . |
| Caption | Returns the friendly name of the SignedDataBlock . |
| SignatureRelation | Returns the relation among multiple signatures on the SignedDataBlock . |
| Signatures | Returns a collection of the signatures of a SignedDataBlock . |
| XPathNamespaceDeclarations | Returns the namespace definitions for the SignedDataBlock . |
| SignatureContainer | Returns the root XML node of the sub-tree that contains the signature(s). |

The **SignedDataBlock** object contains the following method:

| Method | Description |
|----------------------|---|
| Sign | Invokes the Digital Signatures dialog box to add a digital signature to the SignedDataBlock section of the form. |

SignEvent Object

An event object that is used during the [OnSign](#) event.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the **SignEvent** object

Use the [SignedDataBlock](#) property of the **SignEventObject** object to determine which signed data block is triggering the **OnSign** event. The **OnSign** event is raised for a fully trusted [form template](#) only.

Example

In the following example, the **SignEvent** object is used to add a signature and timestamp to a **SignedDataBlock** object:

```
[InfoPathEventHandler(EventType=InfoPathEventType.OnSign)]
public void OnSign(SignEvent e)
{
    Signature signature = e.SignedDataBlock.Signatures.Create();
    signature.Sign();

    // Countersign the signature with a trusted timestamp.

    // Get the XML node storing the signature block.
    IXMLDOMNode oNodeSig = signature.SignatureBlockXmlNode;
    IXMLDOMNode oNodeSigValue = oNodeSig.selectSingleNode("Value");
    // Get time stamp from timestamp service (fictitious).
    MyTrustedTimeStampingService s = new MyTrustedTimeStampingService();
    string strVerifiedTimeStamp = s.AddTimeStamp(oNodeSigValue);

    //Add the value returned from the timestamping service to the
    //unsigned part of the signature block.
    IXMLDOMNode oNodeObj = oNodeSig.selectSingleNode("/*[name()='SignatureValue']");
    IXMLDOMNode oNode = oNodeObj.cloneNode(false);
    oNode.text = strVerifiedTimeStamp;
    oNodeObj.parentNode.appendChild(oNode);

    e.ReturnStatus = true;
}
```


Solution Object

Solution

Corresponds to a Microsoft Office InfoPath 2003 [form template](#). The **Solution** object implements properties for getting information about a form template, including its version number, the [Uniform Resource Locator \(URL\)](#) of its extracted [form files](#), the URL it was loaded from, and an XML [Document Object Model \(DOM\)](#) containing its [form definition \(.xsf\) file](#).

Remarks

The **Solution** object is accessed through the [Solution](#) property of the [XDocument](#) object.

Using the Solution object

In the following example, a reference is set to the **Solution** object, then the code gets the name of the person who authored the form from the .xsf using the **DOM** property of the **Solution** object. A test is then made to determine if there is an author value, and the results are displayed in a message box.

```
function SolutionInfo::OnClick(eventObj)
{
    var objSolution = XDocument.Solution;
    var strSolutionVersion = objSolution.Version;
    var objAuthorNode = objSolution.DOM
        .selectSingleNode("xsf:xDocumentClass/@author");
    var strAuthorText;

    if (objAuthorNode != null)
        strAuthorText = objAuthorNode.text;
    else
        strAuthorText = "Author not available.";

    XDocument.UI.Alert("Version: " + strSolutionVersion +
        "\nAuthor: " + strAuthorText);
}
```

For more information about using the **Solution** object, see [Accessing application data](#).

TaskPane Object



Represents a [task pane](#) in a Microsoft Office InfoPath 2003 form that is associated with a window. The **TaskPane** object provides a number of properties for working with the InfoPath built-in task panes, and the [HTMLTaskPane](#) object inherits those properties and methods for working with a [custom task pane](#).

Remarks

The properties and methods that are available for an InfoPath task pane are determined by the type of task pane that you are working with. If the [Type](#) property of the **TaskPane** object returns 0, the task pane is a custom task pane and the properties and methods that are available are provided by the **HTMLTaskPane** object. If the **Type** property returns any other value, the task pane is a built-in task pane and the properties are provided by the **TaskPane** object directly.

Note The **Type** property is based on the [XdTaskPaneType](#) enumeration. These enumerated values are also used as arguments to the [Item](#) property of the [TaskPanes](#) collection for returning a reference to a specified type of task pane.

Using the TaskPane object

In the following example from the User Interface developer sample form, the **Item** property of the **TaskPanes** collection is used to set a global reference to the **TaskPane** object that represents the custom task pane. The code then calls a scripting function defined in the **HTML** code of the custom task pane using the **HTMLDocument** property of the **HTMLTaskPane** object, which is inherited by the **TaskPane** object.

```
function SetTaskPaneState()
{
    // Ensure View has loaded before trying to access the task pane.
    if (XDocument.View)
    {
        // Get a reference to the custom task pane. It is always the 0-th
        // task pane in the TaskPanes collection.
        if (gobjTaskPane == null)
            gobjTaskPane = XDocument.View.Window.TaskPanes.Item(0);

        // Ensure that the task pane is completely loaded.
        if (gobjTaskPane.HTMLDocument.readyState == "complete")
        {
            var strTaskPaneViewId = "TP_" + XDocument.View.Name.repla

            // Call a script function defined in the task pane HTML page.
            gobjTaskPane.HTMLDocument.parentWindow.SelectView(strTa
        }
    }
}
```

UI Object



Represents various user interface components that can be used in a Microsoft Office InfoPath 2003 form. The **UI** (user interface) object provides a number of methods for displaying custom and built-in dialog boxes.

Remarks

The **UI** object is used to programmatically display various types of dialog boxes to users as they fill out a form. It is not used for modifying the InfoPath user interface.

Using the UI object

The **UI** object is accessed through the **UI** property of the [XDocument](#) object.

The **UI** object provides the [Alert](#) method, which is used to display a simple message box with some custom text, as shown in the following example:

```
XDocument.UI.Alert("Custom message text goes here.");
```

One of the built-in InfoPath dialog boxes that the **UI** object can display is the **Digital Signatures** dialog box; this dialog box can be displayed to the user by using the [ShowSignatureDialog](#) method, as follows:

```
XDocument.UI.ShowSignatureDialog();
```

Note The **ShowSignatureDialog** method can be used only in forms that have been enabled for digital signing. The method will return an error if used in a form that is not enabled for digital signing.

For more information about using the **UI** object, see [Displaying alerts and dialog boxes](#).

User Object

Provides methods that can be used to return information about the current user.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the User object

Use the [User](#) property of the [Application](#) object to return a **User** object.

The **User** object contains the following methods:

| Method | Description |
|--------------------------------|--|
| IsCurrentUser | Returns True if the current user matches the specified user name. |
| IsUserMemberOf | Returns True if current user is a member of the specified group. |

Util Object

Provides utility methods that can be used with Microsoft Office InfoPath 2003 [forms](#).

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Using the Util object

Use the [Util](#) property of the [XDocument](#) object to create a **Util** object.

The **Util** object contains the following properties:

| Property | Description |
|----------------------|---|
| Date | Use this property to return a Date object, which provides several date-related methods. |
| Math | Use this property to return a Math object, which provides several math-related methods. |

The **Util** object contains the following method:

| Method | Description |
|-----------------------|---|
| Match | Indicates whether a string matches a specified pattern. |

VersionUpgradeEvent Object

VersionUpgradeEvent

An event object that is used during a Microsoft Office InfoPath 2003 version upgrade event. The **VersionUpgradeEvent** object provides a number of properties that can be used within a version upgrade event to programmatically interact with a form's underlying XML document, determine the version numbers of the form and [form template](#), and provide a response indicating the success of the version upgrade process.

Remarks

The **VersionUpgradeEvent** object is passed as a parameter to the [OnVersionUpgrade](#) event of an InfoPath form. The properties that it provides are available only during this event.

Using the VersionUpgradeEvent object

In the following example, the [DocumentVersion](#) and [SolutionVersion](#) properties of the **VersionUpgradeEvent** object are used to display the version numbers of the form and form template:

```
function XDocument::OnVersionUpgrade(eventObj)
{
    XDocument.UI.Alert("The form version: " + eventObj.DocumentVer
        "\nThe form template version: " + eventObj.SolutionVersion);
    eventObj.ReturnStatus = true;
}
```

For more information about using the **VersionUpgradeEvent** object, see [Responding to form events](#).

View Object

View

Represents a view within a Microsoft Office InfoPath 2003 form. The **View** object provides a number of properties and methods that can be used to programmatically interact with an InfoPath view, including methods for selecting data contained in the view, switching from one view to another, synchronizing the view with a form's underlying XML document, and executing an InfoPath editing action.

Remarks

InfoPath forms can contain one or more views, and one view is always defined as the default view. When you work with a view using the **View** object, you are accessing the currently active view.

Note The InfoPath object model also provides the [ViewInfos](#) collection, which can be used to get information about all of the views implemented in a form.

Using the View object

The **View** object is accessed through the [View](#) property of the [XDocument](#) object. For example, the following code sets a reference to a form's currently active view:

```
var objView;
```

```
objView = XDocument.View;
```

To change the currently active view, you can use the [SwitchView](#) method of the **View** object as follows:

```
XDocument.View.SwitchView("View2");
```

To force an update of a view based on changes in a form's underlying XML document, you can use the [ForceUpdate](#) method of the **View** object, as shown here:

```
XDocument.View.ForceUpdate();
```

For more information about using the **View** object, see [Working with views](#).

ViewInfo Object



Contains descriptive information about a view within a Microsoft Office InfoPath 2003 form.

Remarks

The **ViewInfo** object provides properties that can be used to get the name of a view and determine whether a view is the default view of the form. The **ViewInfo** object is accessible through the [ViewInfos](#) property of the [XDocument](#) object.

Note To work with the view programmatically, use the [View](#) object.

Using the ViewInfo object

In the following example, the **IsDefault** property of the **ViewInfo** object is used to determine whether a view is the default view of the form. Then the code uses the **Name** property of the **ViewInfo** object to display the name of the default view in a message box.

```
var objViewInfos;

// Set a reference to the ViewInfos collection.
objViewInfos = XDocument.ViewInfos;

// Determine the default view and display a
// message box with its name.
for (i=0; i < objViewInfos.Count; i++)
{
    if (objViewInfos(i).IsDefault)
        XDocument.UI.Alert("The default view is: " + XDocument.ViewIr
}
```


WebServiceAdapter Object

WebServiceAdapter

Represents a connection to an XML Web service. The **WebServiceAdapter** object is a type of Microsoft Office InfoPath 2003 [data adapter](#) that contains all the information necessary for retrieving data from and submitting data to an external data source.

Remarks

The **WebServiceAdapter** object provides properties that can be used to get and set information about the data adapter's input and operation strings. It also provides a property for getting the [Uniform Resource Locator \(URL\)](#) of the Web Services Description Language (WSDL) file of the XML Web service.

Using the **WebServiceAdapter** object

For [secondary data sources](#), the **WebServiceAdapter** object is accessible through the [QueryAdapter](#) property of the [DataObject](#) object, and **DataObject** objects are accessible through the [DataObjects](#) property of the [XDocument](#) object.

If a Web service is used as the primary data source for a form, the **WebServiceAdapter** object is accessible through the [QueryAdapter](#) property of the **XDocument** object.

In the following example, a reference to the **WebServiceAdapter** object is set by using the [QueryAdapter](#) property of the **DataObject** object by passing the name of the **DataObject** object to the [Item](#) property of the **DataObjects** collection:

```
var objWSAdapter;
```

```
objWSAdapter = XDocument.DataObjects("GetCityList").QueryAdap
```

After the reference has been set, you can use the properties of the **WebServiceAdapter** object as shown in the following example, which sets a reference to the XML node returned by the [Input](#) property:

```
var objInputNode;
```

```
objInputNode = XDocument.DataObjects("GetCityList").QueryAdapte
```

Note that in this case, the [QueryAdapter](#) property was used to access the [Input](#) property of the **WebServiceAdapter** object.

For more information about using the **WebServiceAdapter** object, see [Accessing external data sources](#).

The **WebServiceAdapter** object contains the following properties:

| Property | Description |
|----------|-------------|
|----------|-------------|

| | |
|------------------------------|---|
| <u>ErrorsLocation</u> | A property of type IXMLDOMNode that sets or retrieves the errors node under which the WebServiceAdapter object will write the error details returned by the Web service. The initial value is null . |
| <u>Input</u> | Sets or retrieves a string that contains the source XML of the input element contained in the form definition file (.xsf) . |
| <u>Name</u> | A read-only property of type String that returns the name of the WebServiceAdapter object. |
| <u>Operation</u> | Sets or retrieves a string that contains the source XML of the operation element contained in the form definition (.xsf) file. |
| <u>OutputLocation</u> | A property of type IXMLDOMNode that sets or retrieves the output node under which the WebServiceAdapter object will copy the returned XML. |
| <u>QueryAllowed</u> | A read-only property of type Boolean that corresponds to the queryAllowed attribute in the form definition (.xsf) file. |
| <u>SubmitAllowed</u> | A read-only property of type Boolean that corresponds to the submitAllowed attribute in the form definition file (.xsf). |
| <u>Timeout</u> | A property of type Long that sets or retrieves the length of time, in seconds, for the WebServiceAdapter object to time-out on subsequent requests. The default value is 30 seconds. |
| <u>WSDLURL</u> | A read-only property that returns a string that contains the Uniform Resource Locator (URL) of the Web Services Description Language (WSDL) file for the Web service associated with the WebServiceAdapter object. |

The WebServiceAdapter object contains the following methods:

| Method | Property |
|---------------------------------------|--|
| <u>GenerateDataSetDiffGram</u> | Returns an ADO.Net DataSet containing an inline schema describing the data and the |

DiffGram of the DataSet. The DiffGram for the input dataDom is generated, using the sibling node **OriginalData** to compute the difference between the **OriginalData** and the input dataDom.

[Query](#)

Executes the **Query** method on the **WebServiceAdapter** object. Fails if the **QueryAllowed** property is **False**.

[Submit](#)

Executes the **Submit** method on the **WebServiceAdapter** object. Fails if the **SubmitAllowed** property is **False**.

Window Object

Windows | **Window**

Represents a window that is used in the Microsoft Office InfoPath 2003 application. **Window** objects represent the two types of windows that are used in the InfoPath application: the editing window that is used as the **form area** when a user fills out a form, and the designing window that is used as the **design mode** when a user designs a form.

Remarks

The **Window** object provides a number of properties and methods that can be used to programmatically interact with InfoPath windows, including the ability to activate or close a window and the ability to interact with the [task panes](#) and [command bars](#) that they contain. The **Window** object also provides a property for accessing the form's underlying XML document that is associated with the window.

Note Some properties of the **Window** object are only available when using the editing window type; they will return an error if used with the designing window type.

Using the Window object

The **Window** objects of an InfoPath form are accessed through the **Item** property of the **Windows** collection. The type of window can be determined by the **Type** property of the **Window** object.

You can access the currently open window directly by using the **ActiveWindow** property of the **Application** object, without going through the **Windows** collection. You can also access the **Window** object that is associated with a view by using the **Window** property of the **View** object.

In the following example, implemented as an OnClick **event handler** for a button control, the **ActiveWindow** property is used to set a reference to the current window. Then the code checks the window type; if it is the editing window type, it displays the number of task panes contained in the window in a message box.

```
function WindowObject::OnClick(eventObj)
{

    var objWindow;

    // Set a reference to the current window.
    objWindow = Application.ActiveWindow;

    // Check that the window is an editing window type.
    if (objWindow.Type == 0)
    {
        // Display the number of task panes in the window.
        XDocument.UI.Alert("Number of task panes: " +
            objWindow.TaskPanes.Count);
    }

    objWindow = null;
}
```

For more information about using the **Window** object, see [Working with form windows](#).

XDocument Object



Represents the underlying XML document of a Microsoft Office InfoPath 2003 form.

Remarks

The **XDocument** object is a key object in the InfoPath object model that provides properties, methods, and events that can be used to programmatically interact with and manipulate the source XML data of a form.

Note The **XDocument** object is embedded in the InfoPath script engine. While the **XDocument** object can be accessed using the [XDocuments](#) collection, in most cases you will access it directly, without going through the collection.

Using the XDocument object

The source XML data of a form takes the form of an XML [Document Object Model \(DOM\)](#), which is accessed through the [DOM](#) property of the **XDocument** object. The **XDocument** object also provides a number of properties that can be used to get information about the form and its underlying XML document. For example, the following code checks to see whether data in the form has been changed by using the [IsDirty](#) property:

```
if (XDocument.IsDirty)
    XDocument.UI.Alert("Form has been changed.");
else
    XDocument.UI.Alert("Form has not been changed.");
```

In addition to information about the form and its underlying XML document, the **XDocument** object provides a number of methods that can be used on the form, such as printing, saving, and submitting. It also provides a number of events that can be used to respond to various actions that occur at the form level, such as loading of a form, switching views, or a merge operation.

Accessing the XDocument object

The **XDocument** object can be accessed in a variety of places within the InfoPath object model. The following table summarizes the locations where the **XDocument** object is available.

| Name | Description |
|--|---|
| XDocuments collection | Accessed from the Application object. Provides the Item property for accessing the XDocument objects that it contains. |
| DataDOMEvent object | Provides an XDocument property for accessing the source XML data during an XML DOM change. |
| DocActionEvent object | Provides an XDocument property for accessing the source XML data during a button click in the form area. |
| DocEvent object | Provides an XDocument property for accessing the source XML data during a switch view or form merge operation. |
| DocReturnEvent object | Provides an XDocument property for accessing the source XML during the loading or submission of a form. |
| VersionUpgradeEvent object | Provides an XDocument property for accessing the source XML during the version upgrade operation. |
| Window object | Provides an XDocument property for accessing the XDocument object associated with the window. |

Note For more information about using the **XDocument** object, see [Accessing form data](#).

XMLFileAdapter Object

XMLFileAdapter

Represents a connection to an XML file. The **XMLFileAdapter** object is a type of Microsoft Office InfoPath 2003 [data adapter](#) that contains all the information necessary for retrieving and submitting data to an external data source.

Remarks

The **XMLFileAdapter** object provides the [FileURL](#) property, which can be used get or set the [Uniform Resource Locator \(URL\)](#) of the XML file that is being used an external data source.

Using the XMLFileAdapter object

For [secondary data sources](#), the **XMLFileAdapter** object is accessible through the [QueryAdapter](#) property of the [DataObject](#) object, and **DataObject** objects are accessible through the [DataObjects](#) property of the [XDocument](#) object.

In the following example, a reference to the **XMLFileAdapter** object is set by using the **QueryAdapter** property of the **DataObject** object by passing the name of the **DataObject** object to the [Item](#) property of the **DataObjects** collection:

```
var objXMLFileAdapter;
```

```
objXMLFileAdapter = XDocument.DataObjects("GetXMLCityList").(
```

For more information about using the **XMLFileAdapter** object, see [Accessing external data sources](#).

Active Property

A read-only property that returns a Boolean value that indicates whether the window associated with the [Window](#) object is the active window.

expression.**Active**

expression Required. An expression that returns a reference to the **Window** object.

Security Level

0: Can be accessed without restrictions.

Remarks

To designate a window as the active window, use the [Activate](#) method of the **Window** object.

Example

In the following example, the **Active** property of the **Window** object is used to determine whether the first window contained in the [Windows](#) collection is the active window:

```
if (Application.Windows(0).Active)
{
    XDocument.UI.Alert("The window is active.");
}
```

ActiveWindow Property

A read-only property that returns a reference to a [Window](#) object that represents the currently active window. The type of window returned is based on the [XdWindowType](#) enumeration.

expression.**ActiveWindow**

expression Required. An expression that returns the [Application](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

Using the **ActiveWindow** property, you can gain immediate access to the window that is currently being viewed by the user, and then use the properties and methods of the **Window** object that it returns.

Example

In the following example from the Meeting Agenda sample form, the **ActiveWindow** property is used to gain access to the [MailEnvelope](#) property of the **Window** object that returns a [MailEnvelope](#) object.

```
var oEnvelope = Application.ActiveWindow.MailEnvelope;
```

```
oEnvelope.Subject = getNode("/mtg:meetingAgenda/mtg:subject").text;
```

```
oEnvelope.To = rgRecipients.join("; ");
```

```
oEnvelope.Visible = true;
```

AttachmentFileName Property

Returns or sets a string that represents the file name to be used for the current form when the form is attached to the e-mail message of an [EmailAdapter](#) object.

expression.**AttachmentFileName**

expression Required. An expression that returns a reference to an **EmailAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The ".xml" file name extension will be appended to the string if it is not already included.

If the **AttachmentFileName** property is set to **null**, the form will not be attached to the e-mail message.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

This example shows how to use the **AttachmentFileName** property of an instance of the **EmailAdapter** object to change the name of the form when it is attached to the e-mail message before the **EmailAdapter** object is submitted:

```
objEmailAdapter.AttachmentFileName = strName + "-" + strDate + '
```

BCC Property (EmailAdapter Object)

Returns or sets a string that represents the BCC recipients of an e-mail message associated with an [EmailAdapter](#) object.

expression.**BCC**

expression Required. An expression that returns a reference to an **EmailAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The value must be a semicolon-delimited string that can be resolved into a list of valid e-mail addresses by the user's e-mail client.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

This example shows how to use the **BCC** property of an instance of the **EmailAdapter** object to change the BCC recipients before the **EmailAdapter** is submitted:

```
objEmailAdapter.BCC = oEmailAdapter.BCC + "; newUser@example.com";
```

BCC Property (Index)

The **BCC** property sets or retrieves a string containing the blind carbon copy (BCC) value for an e-mail message. This property is implemented in several Microsoft Office InfoPath 2003 object model objects. Click a **BCC** property link below to view the Help topic for a specific implementation of the **BCC** property.

[BCC](#) property as it applies to the [EmailAdapter](#) object.

[BCC](#) property as it applies to the [MailEnvelope](#) object.

BCC Property (MailEnvelope Object)

Sets or retrieves a string containing the blind carbon copy (BCC) value used in the [MailEnvelope](#) object that is associated with a [Window](#) object.

expression.**BCC**

expression Required. An expression that returns a reference to the **MailEnvelope** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The values set for the **BCC** property should be a string of valid e-mail addresses. You can specify multiple e-mail addresses by using ";" between each of them, as shown in the following example:

```
objEmail.BCC = "someone@example.com;someone@example.com"
```

Example

In the following example, the **BCC** property of the **MailEnvelope** object is used to set the BCC value of a custom e-mail message:

```
function CreateMailEnvelope::OnClick(eventObj)
{
    var objEmail;

    objEmail = Application.ActiveWindow.MailEnvelope;
    objEmail.To = "someone@example.com";
    objEmail.CC = "someone@example.com";
    objEmail.BCC = "someone@example.com";
    objEmail.Subject = "Test e-mail message";
    objEmail.Visible = true;
    objEmail = null;
}
```

Caption Property (Index)

This property is implemented in several Microsoft Office InfoPath 2003 object model objects. Click a **Caption** property link below to view the Help topic for a specific implementation of the **Caption** property.

[Caption](#) property as it applies to the [SignedDataBlock](#) object.

[Caption](#) property as it applies to the [Window](#) object.

Caption Property (SignedDataBlock Object)

A read-only property that returns the friendly name of the [SignedDataBlock](#) object.

expression.**Caption**

expression Required. An expression that returns a reference to the **SignedDataBlock** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Caption Property (Window Object)

A read/write property that returns or sets the caption text for the window represented by the [Window](#) object.

expression.**Caption**

expression Required. An expression that returns a reference to the **Window** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Remarks

When setting the caption text for a window, the caption will always be followed by "- Microsoft Office InfoPath 2003".

Example

In the following example, the **Caption** property is used to set the caption text of the active window.

```
var strCaption = "MyCaption";
```

```
Application.ActiveWindow.Caption = strCaption;
```

CC Property (EmailAdapter Object)

Returns or sets a string that represents the carbon copy (CC) recipients for the e-mail message associated with an [EmailAdapter](#) object.

expression.**CC**

expression Required. An expression that returns a reference to an **EmailAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The value must be a semicolon-delimited string that can be resolved into a list of valid e-mail addresses by the user's e-mail client.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

This example shows how to use the **CC** property of an instance of the **EmailAdapter** object to change the CC recipients before the EmailAdapter object is submitted:

```
objEmailAdapter.CC = oEmailAdapter.CC + "  
newUser@example.com";
```

CC Property (MailEnvelope Object)

Sets or retrieves a string containing the carbon copy (CC) value used in the [MailEnvelope](#) object that is associated with a [Window](#) object.

expression.**CC**

expression Required. An expression that returns a reference to the **MailEnvelope** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The values set for the **CC** property should be a string of valid e-mail addresses. You can specify multiple e-mail addresses by using ";" between each of them, as shown in the following example:

```
objEmail.CC = "someone@example.com;someone@example.com"
```

Example

In the following example, the **CC** property of the **MailEnvelope** object is used to set the CC value of a custom e-mail message:

```
function CreateMailEnvelope::OnClick(eventObj)
{
    var objEmail;

    objEmail = Application.ActiveWindow.MailEnvelope;
    objEmail.To = "someone@example.com";
    objEmail.CC = "someone@example.com";
    objEmail.BCC = "someone@example.com";
    objEmail.Subject = "Test e-mail message";
    objEmail.Visible = true;
    objEmail = null;
}
```

CC Property (Index)

The **CC** property sets or retrieves a string containing the carbon copy (CC) value for an e-mail message. This property is implemented in several Microsoft Office InfoPath 2003 object model objects. Click a **CC** property link below to view the Help topic for a specific implementation of the **CC** property.

[CC](#) property as it applies to the [EmailAdapter](#) object.

[CC](#) property as it applies to the [MailEnvelope](#) object.

Certificate Property

A read-only property that returns the [Certificate](#) object for the X.509 [digital certificate](#) that was used to sign a [form](#) or a set of signed data in a form.

expression.**Certificate**

expression Required. An expression that returns a [Signature](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Command Property

Sets or retrieves the SQL command string text for an [ADOAdapter](#) object.

expression.**Command**

expression Required. An expression that returns a reference to the **ADOAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Command** property of the **ADOAdapter** object contains the SQL command text that is used by the ADO [data adapter](#) to submit data to and retrieve data from an ActiveX Data Objects/OLEDB external data source.

Note The **ADOAdapter** object is limited to work only with Microsoft SQL Server and Microsoft Access databases.

Example

In the following example, the **Command** property of the **ADOAdapter** object is used to display the SQL command text of the ADO data adapter in a message box:

```
var objADOAdapter;
```

```
objADOAdapter = XDocument.DataObjects("CityList").QueryAdapte;  
XDocument.UI.Alert("SQL command text: " + objADOAdapter.Comr
```

CommandBars Property

A read-only property that returns a reference to the Microsoft Office [CommandBars](#) collection object contained in the window that is associated with the [Window](#) object.

expression.**CommandBars**

expression Required. An expression that returns a reference to the **Window** object.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

The **CommandBars** property can be used only by fully trusted forms. If used by a form that is not fully trusted, the **CommandBars** property will return a "permission denied" error.

Example

In the following example, the **CommandBars** property of the **Window** object is used to set a reference to the **CommandBars** collection object:

```
var objCommandBars;  
objCommandBars = Application.ActiveWindow.CommandBars;
```


Comment Property

A read-only property that returns the text comment that was added to the [digital signature](#) associated with a form or a section of a form.

expression.**Comment**

expression Required. An expression that returns a [Signature](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

ConditionName Property

A read-only property that returns a string value containing the name of the **Error** object.

expression.**ConditionName**

expression Required. An expression that returns a reference to the **Error** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **ConditionName** property is used only for custom errors; it is not used for schema validation–generated or system-generated errors. If the type of error is schema validation generated or system generated, the **ConditionName** property will return null.

Example

In the following example, the **ConditionName** property of the **Error** object is used to display the name of a custom error in a message box:

```
var objError;
```

```
objError = XDocument.Errors(0);
```

```
XDocument.UI.Alert("Error name: " + objError.ConditionName);
```


Connection Property

Sets or retrieves the connection string used for an [ADOAdapter](#) object.

expression.**Connection**

expression Required. An expression that returns a reference to the **ADOAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Connection** property of the **ADOAdapter** object contains the connection string that is used by the ADO [data adapter](#) to connect to an ActiveX Data Objects/OLEDB external data source.

Note The **ADOAdapter** object is limited to work only with Microsoft SQL Server and Microsoft Access databases.

Example

In the following example, the **Connection** property of the **ADOAdapter** object is used to display the connection string of the ADO data adapter in a message box:

```
var objADOAdapter;
```

```
objADOAdapter = XDocument.DataObjects("CityList").QueryAdapter;  
XDocument.UI.Alert("Connection string: " + objADOAdapter.Connec
```

Context Property

A read-only property that returns a reference to the XML Document Object Model (DOM) node that is the new context node provided by the [DocContextChangeEvent](#) object.

expression.**Context**

expression Required. An expression that returns a reference to the **DocContextChangeEvent** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Count Property (DataAdapters Collection)

A read-only property that returns the number of data adapters in the [DataAdaptersCollection](#) collection.

expression.**Count**

expression Required. An expression that returns a reference to a **DataAdaptersCollection** collection.

Security Level

0: Can be accessed without restrictions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Count Property (DataObjects Collection)

A read-only property that returns a count of the number of [DataObject](#) objects contained in the [DataObjects](#) collection.

expression.**Count**

expression Required. An expression that returns a reference to the **DataObjects** collection.

Security Level

0: Can be accessed without restrictions.

Remarks

The **Count** property returns a **long integer** value.

Example

In the following example, the **Count** property is used within a Microsoft **JScript for** loop to iterate through the collection of **DataObject** objects and display a message box indicating the name of each **DataObject** object associated with the form:

```
for (i=0; i < XDocument.DataObjects.Count; i++)  
{  
    XDocument.UI.Alert("DataObject name: " + XDocument.DataObjec  
}
```


Count Property (Errors Collection)

A read-only property that returns a count of the number of [Error](#) objects contained in the [Errors](#) collection.

expression.**Count**

expression Required. An expression that returns a reference to the **Errors** collection.

Security Level

0: Can be accessed without restrictions.

Remarks

The **Count** property returns a **long integer** value.

Example

In the following example, the **Count** property is used within a Microsoft [JScript for](#) loop to iterate through the collection of **Error** objects and display a message box indicating the short error message of each error:

```
for (i=0; i < XDocument.Errors.Count; i++)  
{  
    XDocument.UI.Alert("Error message: " + XDocument.Errors(i).ShortMessage);  
}
```

Count Property (Index)

The **Count** property returns a count of the number of objects contained in a collection. This property is implemented in several Microsoft Office InfoPath 2003 object model collections. Click a **Count** property link below to view the Help topic for a specific implementation of the **Count** property.

[Count](#) property as it applies to the [DataAdapters](#) collection.

[Count](#) property as it applies to the [DataObjects](#) collection.

[Count](#) property as it applies to the [Errors](#) collection.

[Count](#) property as it applies to the [MergeEvent](#) object.

[Count](#) property as it applies to the [Signatures](#) collection.

[Count](#) property as it applies to the [SignedDataBlocks](#) collection.

[Count](#) property as it applies to the [TaskPanels](#) collection.

[Count](#) property as it applies to the [ViewInfos](#) collection.

[Count](#) property as it applies to the [Windows](#) collection.

[Count](#) property as it applies to the [XDocuments](#) collection.

[Count](#) property as it applies to the [XMLNodes](#) collection.

Count Property (MergeEvent Object)

A read-only property that returns a count of the number of forms being merged in a merging operation.

expression.**Count**

expression Required. An expression that returns a reference to a [MergeEvent](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

The **Count** property can be used in combination with the [Index](#) property of the **MergeEvent** object to determine when the last form was merged.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

This example shows how to use the **Index** property along with the **Count** property of an instance of the **MergeEvent** object to determine if the current form is the last form being merged:

```
var fLast = eventObj.Index + 1 == eventObj.Count;
```

Count Property (Signatures Collection)

A read-only property that returns the number of [Signature](#) objects in the [Signatures](#) collection.

expression.**Count**

expression Required. An expression that returns a reference to a **Signatures** collection.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Count Property (SignedDataBlocks Collection)

A read-only property that returns the number of [SignedDataBlock](#) objects in the [form template](#).

expression.**Count**

expression Required. An expression that returns a [SignedDataBlocks](#) collection.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Count Property (TaskPanes Collection)

A read-only property that returns a count of the number of [TaskPane](#) objects contained in the [TaskPanes](#) collection.

expression.**Count**

expression Required. Returns a reference to the **TaskPanes** collection.

Security Level

0: Can be accessed without restrictions.

Remarks

The **Count** property returns a **long integer** value.

Example

In the following example, the **Count** property is used within a Microsoft **JScript for** loop to iterate through the collection of **TaskPane** objects. It checks the **Visible** property of each **TaskPane** object and if it is True, sets it to False.

```
var objTaskPanes;
```

```
objTaskPanes = XDocument.View.Window.TaskPanes;
```

```
for (i=0; i < objTaskPanes.Count; i++)  
{  
    if (objTaskPanes(i).Visible = true)  
        objTaskPanes(i).Visible = false;  
}
```


Count Property (ViewInfos Collection)

A read-only property that returns a count of the number of [ViewInfo](#) objects contained in the [ViewInfos](#) collection.

expression.**Count**

expression Required. An expression that returns a reference to the **ViewInfos** collection.

Security Level

0: Can be accessed without restrictions.

Remarks

The **Count** property returns a **long integer** value.

Example

In the following example, the **Count** property is used within a Microsoft [JScript for](#) loop to iterate through the collection of **ViewInfo** objects and display a message box indicating the name of each view implemented in the form:

```
for (i=0; i < XDocument.ViewInfos.Count; i++)  
{  
    XDocument.UI.Alert("View name: " + XDocument.ViewInfos(i).Name);  
}
```


Count Property (Windows Collection)

A read-only property that returns a count of the number of [Window](#) objects contained in the [Windows](#) collection.

expression.**Count**

expression Required. An expression that returns a reference to the **Windows** collection.

Security Level

0: Can be accessed without restrictions.

Remarks

The **Count** property returns a **long integer** value.

Example

In the following example, the **Count** property is used within a Microsoft **JScript for** loop to iterate through the collection of **Window** objects and display a message box indicating the window type value:

```
for (i=0; i < Application.Windows.Count; i++)  
{  
    XDocument.UI.Alert("Window type: " + Application.Windows(i).Ty  
}
```


Count Property (XDocuments Collection)

A read-only property that returns a count of the number of [XDocument](#) objects contained in the [XDocuments](#) collection.

expression.**Count**

expression Required. An expression that returns a reference to the **XDocuments** collection.

Security Level

0: Can be accessed without restrictions.

Remarks

The **Count** property returns a **long integer** value.

Example

In the following example, the **Count** property of the **XDocuments** collection is used within a **JScript for** loop to iterate through the collection of **XDocument** objects and display a message box indicating the **Uniform Resource Identifier (URI)** location for each currently open form:

```
for (i=0; i < Application.XDocuments.Count; i++)  
{  
    XDocument.UI.Alert("XDocument URI: " + Application.XDocumer  
}
```


Count Property (XMLNodes Collection)

A read-only property that returns a count of the number of XML [Document Object Model \(DOM\)](#) node objects contained in the [XMLNodes](#) collection.

expression.Count

expression Required. Returns a reference to the **XMLNodes** collection.

Security Level

0: Can be accessed without restrictions.

Remarks

The **Count** property returns a **long integer** value.

Example

In the following example, the **Count** property is used within a Microsoft **JScript for** loop to iterate through the collection of XML DOM node objects returned by the **GetSelectedNodes** method of the **View** object and display a message box indicating the name of each XML DOM node that is contained in the **XMLNodes** collection:

```
var objXMLNodes;
```

```
objXMLNodes = XDocument.View.GetSelectedNodes();
```

```
for (i=0; i < objXMLNodes.Count; i++)
```

```
{
```

```
    XDocument.UI.Alert("XML DOM node name: " + objXMLNodes(i
```

```
})
```

DataAdapters Property

A read-only property that returns a reference to the [DataAdapters](#) collection that is associated with a Microsoft Office InfoPath 2003 form.

expression.**DataAdapters**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **DataAdapters** property of the **XDocument** object is used to set a reference to a data adapter called "Main query":

```
var objDataAdapter;  
objDataAdapter = XDocument.DataAdapters("Main query");
```


DataObjects Property

A read-only property that returns a reference to the [DataObjects](#) collection that is associated with a Microsoft Office InfoPath 2003 form.

expression.**DataObjects**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **DataObjects** collection provides programmatic access to a form's [secondary data sources](#). Each secondary data source is contained in a [DataObject](#) object within the **DataObjects** collection.

Example

In the following example, the **DataObjects** property of the **XDocument** object is used to set a reference to the CityList secondary data source:

```
var objDataObject;  
objDataObject = XDocument.DataObjects("CityList");
```

Date Property

A read-only property that returns a reference to the [Date](#) object.

expression.**Date**

expression Required. An expression that returns a reference to the [Util](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

You can access all of the methods that the **Date** object provides by using the **Date** property.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

DetailedErrorMessage Property

Sets or retrieves the string value containing the detailed error message of an [Error](#) object.

expression.**DetailedErrorMessage**

expression Required. An expression that returns a reference to the **Error** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The detailed error message is the longer error message that users can choose to view when data validation fails in their forms.

Example

In the following example, the **DetailedErrorMessage** property of the **Error** object is used to display the detailed message of an error in a message box:

```
var objError;
```

```
objError = XDocument.Errors(0);
```

```
XDocument.UI.Alert("Error name: " + objError.DetailedErrorMessage");
```

DocumentVersion Property

A read-only property that returns a string containing the version number of a Microsoft Office InfoPath 2003 form.

expression.**DocumentVersion**

expression Required. An expression that returns a reference to the [VersionUpgradeEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

This property can be used only during the [OnVersionUpgrade](#) event.

Example

In the following example, the **DocumentVersion** property of the **VersionUpgradeEvent** object is used to display the version number of an InfoPath form in a message box:

```
function XDocument::OnVersionUpgrade(eventObj)
{
    XDocument.UI.Alert("The form version: " + eventObj.DocumentVe
        "\nThe form template version: " + eventObj.SolutionVersion);
    eventObj.ReturnStatus = true;
}
```


DOM Property (DataObject Object)

A read-only property that returns a reference to an XML [Document Object Model \(DOM\)](#) that is associated with a [DataObject](#) object.

expression.**DOM**

expression Required. An expression that returns a reference to the **DataObject** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **DOM** property allows you to programmatically access and manipulate the source XML of a [secondary data source](#) that is represented by the **DataObject** object. After you have set a reference to the XML DOM, which contains the source XML data of a secondary data source, you can use any of the properties and methods that are supported by the XML DOM.

Note To learn more about the XML DOM and all of the properties and methods that it supports, see the MSXML 5.0 SDK documentation in the [Microsoft Script Editor \(MSE\)](#) Help system.

Example

In the following example, the **DOM** property of the **DataObject** object is used to return all of the contents of a form's secondary data source using the **xml** property of the XML DOM:

```
var strXML;  
strXML = XDocument.DataObjects("CityList").DOM.xml;
```


DOM Property (Index)

The **DOM** property returns a reference to the [Document Object Model \(DOM\)](#). This property is implemented in several Microsoft Office InfoPath 2003 object model objects. Click a **DOM** property link below to view the Help topic for a specific implementation of the **DOM** property.

[DOM](#) property as it applies to the [DataObject](#) object.

[DOM](#) property as it applies to the [MergeEvent](#) object.

[DOM](#) property as it applies to the [Solution](#) object.

[DOM](#) property as it applies to the [XDocument](#) object.

DOM Property (MergeEvent Object)

A read-only property that returns a reference to a form's underlying XML document as an XML [Document Object Model \(DOM\)](#) construction for the current form involved in a merging operation.

expression.**DOM**

expression Required. An expression that returns a reference to a [MergeEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **DOM** property allows you to access and manipulate the source XML of a form programmatically. After you have set a reference to the XML DOM, which contains the source XML data of a form, you can use any of the properties and methods that are supported by the XML DOM.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Note To learn more about the XML DOM and the properties and methods that it supports, see the MSXML 5.0 SDK documentation in the [Microsoft Script Editor \(MSE\)](#) Help system.

Example

This example shows how to use the **DOM** property of an instance of the **MergeEvent** object to merge the data into the current form:

```
XDocument.ImportDOM(eventObj.DOM);
```


DOM Property (Solution Object)

A read-only property that returns a reference to an XML [Document Object Model \(DOM\)](#) that contains the source XML of the [form definition \(.xsf\)](#) file.

expression.**DOM**

expression Required. An expression that returns a reference to a [Solution](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **DOM** property of the **Solution** object allows you to programmatically access and manipulate the source XML of the .xsf file. After you have set a reference to the XML DOM, which contains the source XML data of the .xsf file, you can use any of the properties and methods that are supported by the XML DOM.

Note To learn more about the XML DOM and all of the properties and methods that it supports, see the MSXML 5.0 SDK documentation in the [Microsoft Script Editor \(MSE\)](#) Help system.

Example

In the following example, the **DOM** property of the **Solution** object is used to return all of the contents of a form's .xsf file using the **xml** property of the XML DOM:

```
var strXML;
```

```
strXML = XDocument.Solution.DOM.xml;
```


DOM Property (XDocument Object)

A read-only property that returns a reference to a form's underlying XML document in the form of an XML [Document Object Model \(DOM\)](#).

expression.**DOM**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

A key property of the **XDocument** object, the **DOM** property allows you to programmatically access and manipulate the source XML of a form. After you have set a reference to the XML DOM, which contains the source XML data of a form, you can use any of the properties and methods that are supported by the XML DOM.

Note To learn more about the XML DOM and all of the properties and methods that it supports, see the MSXML 5.0 SDK documentation in the [Microsoft Script Editor \(MSE\)](#) Help system.

Example

In the following example, the **DOM** property of the **XDocument** object is used to return all of the contents of a form's underlying XML document using the **xml** property of the XML DOM:

```
var strXML;  
strXML = XDocument.DOM.xml;
```

ErrorCode Property

Sets or retrieves the error code of an [Error](#) object.

expression.**ErrorCode**

expression Required. An expression that returns a reference to the **Error** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **ErrorCode** property is implemented as a **long integer**.

Example

In the following example, the **ErrorCode** property of the **Error** object is used to display the error code number of an error in a message box:

```
var objError;
```

```
objError = XDocument.Errors(0);
```

```
XDocument.UI.Alert("Error name: " + objError.ErrorCode);
```

Errors Property

A read-only property that returns a reference to the [Errors](#) collection that is associated with a Microsoft Office InfoPath 2003 form.

expression.**Errors**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Errors** collection is associated with a form's underlying XML document so that when an error occurs, it occurs within the XML document. After you set a reference to the **Errors** collection, you can access all of its properties and methods for managing the errors within an InfoPath form.

Example

In the following example, the **Errors** property of the **XDocument** object is used to return the count of the number of errors and then display that value in a message box:

```
var intErrors;
```

```
intErrors = XDocument.Errors.Count;
```

```
XDocument.UI.Alert("Total number of errors: " + intErrors);
```


ErrorsLocation Property

A read/write property that specifies the XML [Document Object Model \(DOM\)](#) node under which the adapter will copy returned errors as XML.

expression.**ErrorsLocation**

expression Required. An expression that returns a reference to a [WebServiceAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The initial value of the **ErrorsLocation** property is **null**.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

ExpirationDate Property

A read-only property that returns the expiration date of a [digital certificate](#) in the localized 'SHORTDATE' format.

expression.**ExpirationDate**

expression Required. An expression that returns a [Certificate](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Extension Property

A read-only property that returns a reference to the global scripting object, which exposes the functions and global variables contained in a Microsoft Office InfoPath 2003 form's primary scripting file.

expression.**Extension**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Extension** property facilitates using the functions and global variables implemented in an InfoPath form's scripting file. Usually, it is used from a [custom task pane](#), from a custom dialog box, or in the [XSL Transformation \(XSLT\)](#) of a view where direct access to the functions and variables may be needed.

For example, if you were to declare a global variable in your scripting file such as the following,

```
var constCity = "Redmond";
```

You could access that value in the code of your custom task pane or dialog box by using the **Extension** property, as shown here:

```
XDocument.Extension.constCity;
```

To use the **Extension** property within a custom task pane, you must first set a reference to the **XDocument** object by using the [Dynamic HTML \(DHTML\) external](#) property of the DHTML **window** object, as shown in this example:

```
objXDocument = window.external.XDocument;
```

Example

In the following example, the **Extension** property of the **XDocument** object is used to access a custom function:

```
objXDocument = window.external.XDocument;  
objXDocument.Extension.MyCustomFunction();
```

FileName Property (DAVAdapter Object)

Returns or sets a string that represents the file name that the current Microsoft Office InfoPath 2003 form will be given when the form is submitted by the [DAVAdapter](#) object.

expression.**FileName**

expression Required. An expression that returns a reference to the **DAVAdapter** object

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Remarks

The ".xml" file name extension will be appended to the string if it is not already included. If the **FileName** property is set to **null**, the form is given the name "Form.xml" when it is submitted.

Example

The following example shows how to use the **FileName** property of an instance of the **DAVAdapter** object to set the file name of the submitted form:

```
objDAVAdapter.FileName = strName + "-" + strDate + ".xml";
```

FileName Property (Index)

This property is implemented in several Microsoft Office InfoPath 2003 object model collections. Click a **FileName** property link below to view the Help topic for a specific implementation of the **FileName** property.

[FileName](#) property as it applies to the [DAVAdapter](#) object.

[FileName](#) property as it applies to the [SaveEvent](#) object.

FileName Property (SaveEvent Object)

A read-only property that returns a string that represents the file name to be used in the [OnSaveRequest](#) event.

expression.**FileName**

expression Required. An expression that returns a reference to a [SaveEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the [IsSaveAs](#) property of the **SaveEvent** object is true, and a save operation has not yet occurred, the **FileName** property returns an empty string. If the **IsSaveAs** property is true and a save operation has already occurred, the value returned by the **FileName** property is the file name under which the form was saved. If the **SaveEvent** object represents a save operation (that is, the **IsSaveAs** property of the **SaveEvent** object is false), the **FileName** property returns the same value as the [URI](#) property of the [XDocument](#) object.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **FileName** property of the **SaveEvent** object is used in a notification to the user before a save operation:

```
XDocument.UI.Alert("You are about to save the following file: " + eve
```


FileURL Property

Sets or retrieves the [Uniform Resource Locator \(URL\)](#) of the XML file that is associated with the [XMLFileAdapter](#) object.

expression.**FileURL**

expression Required. An expression that returns a reference to the **XMLFileAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **FileURL** property sets and retrieves a **string** value.

Example

In the following example, the **FileURL** property of the **XMLFileAdapter** object, accessed through the [QueryAdapter](#) property of the [XDocument](#) object, is used to display the URL of the XML file in a message box:

```
XDocument.UI.Alert("XML file URL: " + XDocument.QueryAdapter.
```

FolderURL Property

A property that returns or sets a string that represents the Uniform Resource Locator (URL) to which the form will be submitted by the [DAVAdapter](#) object.

expression.**FolderURL**

expression Required. An expression that returns a reference to the **DAVAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The [Submit](#) method of the **DAVAdapter** object will fail if the value of the **FolderURL** property does not begin with either the "http:" or the "https:" prefix. The **Submit** method will also fail if the site specified by the **FolderURL** property is not available, or if the URL is not in the same security domain as the form template.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

This example shows how to use the **FolderURL** property of an instance of the **DavAdapter** object to change the folder to which the form data will be submitted:

```
objDavAdapter.FolderURL = objDavAdapter.FolderURL + "data/";
```

Height Property

A read/write property of type **long integer** that specifies the height of the the window represented by the [Window](#) object, measured in points.

expression.**Height**

expression Required. An expression that returns a reference to the **Window** object.

Remarks

This property will return an error if it is set on a window that is minimized or maximized.

This property can't be set to a value that is larger than the value returned by the [UsableHeight](#) property.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

HTMLDocument Property

Returns a reference to an [HTML](#) document object of the Microsoft Office InfoPath 2003 [custom task pane](#).

expression.HTMLDocument

expression Required. An expression that returns a reference to the HTML document object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **HTMLDocument** property of the [HTMLTaskPane](#) object is one of the properties inherited by the [TaskPane](#) object when the type of the task pane is 0, which means that it is the custom task pane.

Using the **HTMLDocument** method, you can call scripting functions contained in the HTML code of the task pane from the primary scripting file of a form, and you can also directly manipulate the HTML code of the task pane using any of the properties and methods that the HTML document object provides.

Example

In the following example, the **HTMLDocument** property of the **HTMLTaskPane** object is used to set a reference to the HTML document object of the custom task pane. Then the code calls the `TaskPaneSwitchView` custom function that is defined in the HTML code of the custom task pane.

```
var objHTMLDoc;
```

```
objHTMLDoc = XDocument.View.Window.TaskPanels(0).HTMLDoc  
objHTMLDoc.parentWindow.TaskPaneSwitchView();
```


HTMLWindow Property

Returns a reference to an [HTML](#) window object of the Microsoft Office InfoPath 2003 [custom task pane](#).

expression.HTMLWindow

expression Required. An expression that returns a reference to the HTML window object.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

The **HTMLWindow** property of the [HTMLTaskPane](#) object is one of the properties inherited by the [TaskPane](#) object when the type of the task pane is 0, which means that it is the custom task pane.

Using the **HTMLWindow** property, you can call scripting functions contained in the HTML code of the task pane from the primary scripting file of a form, and you can also directly manipulate the HTML code of the task pane using any of the properties and methods that the HTML document object provides.

Note The **HTMLWindow** property provides the same functionality as the [HTMLDocument](#) property, but it is only available when using fully trusted forms. If the form is not fully trusted, you can use the **HTMLDocument** property.

Example

In the following example, the **HTMLWindow** property of the **HTMLTaskPane** object is used to set a reference to the HTML window object of the custom task pane of a fully trusted form. Then the code calls the `TaskPaneSwitchView` custom function that is defined in the HTML code of the custom task pane.

```
var objHTMLDoc;
```

```
objHTMLDoc = XDocument.View.Window.TaskPanels(0).HTMLWin  
objHTMLDoc.parentWindow.TaskPaneSwitchView();
```

Index Property

A read-only property that returns the 0-based index of the form that is currently being merged.

expression.**Index**

expression Required. An expression that returns a reference to a [MergeEvent](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

The value of the **Index** property of the **MergeEvent** object increases from 0 to **Count**-1 for each merge event that occurs when forms are merged.

When used in combination with the **Count** property of the **MergeEvent** object, the **Index** property can be used to determine when the last form has been merged.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

This example shows how to use the **Index** property along with the **Count** property of an instance of the **MergeEvent** object to determine if the current form is the last form to be merged:

```
var fLast = eventObj.Index + 1 == eventObj.Count;
```


Input Property

Sets or retrieves a string value that contains the source XML of the [input](#) element contained in the [form definition \(.xsf\) file](#).

expression.**Input**

expression Required. An expression that returns a reference to the [WebServiceAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **input** element of the .xsf file contains information about the parts of the input Simple Object Access Protocol (SOAP) message that are used to communicate with the Web service. Specific parts in the SOAP message are replaced by Microsoft Office InfoPath 2003 with data from within the form. It is used when a [secondary data source](#) is populated with data from a Web service, and InfoPath needs to pass arguments to the Web service to retrieve the data that it provides.

Example

In the following example, the **Input** property of the **WebServiceAdapter** object is used to display the input string of the Web service data adapter in a message box:

```
var objWSAdapter;
```

```
objWSAdapter = XDocument.DataObjects("WebCityList").QueryAdapt  
XDocument.UI.Alert("Input string: " + objWSAdapter.Input);
```

Intro Property

Returns or sets a string that represents the introduction in the body of the e-mail message for an [EmailAdapter](#) object.

expression.Intro

expression Required. An expression that returns a reference to an **EmailAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

This example shows how to use the **Intro** property of an instance of the **EmailAdapter** object to change the introduction of the body of the e-mail message before the **EmailAdapter** is submitted:

```
objEmailAdapter.Intro = "Here is my status report for the week.";
```

IsCancelled Property

A read/write property that provides additional information for use in the [OnSaveRequest](#) event in conjunction with the [ReturnStatus](#) property.

expression.**IsCancelled**

expression Required. An expression that returns a reference to a [SaveEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

When closing InfoPath, the user is prompted to save the document if the [IsDirty](#) property of the [XDocument](#) object is **true**. If the **IsCancelled** property is **true**, InfoPath will be prevented from closing if the save operation fails (that is, the [ReturnStatus](#) property is **false**).

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **IsCancelled** property of the **SaveEvent** object is used to ensure that the document does not close if the save operation was cancelled:

```
eventObj.IsCancelled = eventObj.PerformSaveOperation();  
if (eventObj.IsCancelled)  
    return;  
eventObj.ReturnStatus = true;
```

IsDefault Property

Sets or retrieves a Boolean value that indicates whether the view is defined as the default view in a Microsoft Office InfoPath 2003 form.

expression.**IsDefault**

expression Required. An expression that returns a reference to the [ViewInfo](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **IsDefault** property of the **ViewInfo** object can be used to determine whether a view is the default view, and it can be used to programmatically change the default view before the first view is loaded.

Example

In the following example, the **IsDefault** property is used to make the second view contained in the [ViewInfos](#) collection the default view:

```
XDocument.ViewInfos(1).IsDefault = true;
```

IsDirty Property

A read-only property that returns a Boolean value that indicates whether the data in a Microsoft Office InfoPath 2003 form has been modified since it was last saved.

expression.**IsDirty**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the **IsDirty** property is **True**, data in the form's underlying XML document has been changed since it was last saved. If **False**, no changes have occurred.

Note Changes that occur during the [OnLoad](#) event will not result in the **IsDirty** property being set to True.

Example

In the following example, the **IsDirty** property of the **XDocument** object is used to determine whether data in the form has been changed:

```
if (XDocument.IsDirty)
    XDocument.UI.Alert("Data has been changed.");
else
    XDocument.UI.Alert("Data has not been changed.");
```


IsDOMReadOnly Property

A read-only property that returns a Boolean value that indicates whether the data in the underlying XML document of a Microsoft Office InfoPath 2003 form has been placed in a read-only state.

expression.**IsDOMReadOnly**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the **IsDOMReadOnly** property is **True**, data in the form's underlying XML document is in a read-only state and cannot be changed. If **False**, the data in the form's underlying XML document can be changed.

To determine whether the form has been placed in a read-only state, use the [IsReadOnly](#) property of the **XDocument** object.

The data in a form's XML document will be placed in a read-only state in the following scenarios:

The form is digitally signed

The form is in Reduced Functionality Mode (RFM) mode

During the [OnBeforeChange](#) event

During the [OnValidate](#) event

During an undo or redo operation

Example

In the following example, the **IsDOMReadOnly** property of the **XDocument** object is used to determine whether data in the form's underlying XML document has been placed in a read-only state. If it has, a return statement is used to exit the [event handler](#).

```
function msoxd__item::OnAfterChange(objEvent)
{

    // Determine whether the XML DOM is read-only.
    if (XDocument.IsDOMReadOnly)
        return;

    // Continue normal processing...

}
```

IsNew Property

A read-only property that returns a Boolean value that indicates whether a newly created Microsoft Office InfoPath 2003 form has been saved.

expression.**IsNew**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the **IsNew** property is **True**, data in the form's underlying XML document has not been saved since the form was initially created. If **False**, the data in the new form's underlying XML document has been saved.

Example

In the following example, the **IsNew** property of the **XDocument** object is used to determine whether the data in a new form has been saved:

```
if (XDocument.IsNew)
    XDocument.UI.Alert("Please save your form.");
else
    return;
```

IsReadOnly Property

A read-only property that returns a Boolean value that indicates whether a Microsoft Office InfoPath 2003 form is in read-only mode.

expression.**IsReadOnly**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the **IsReadOnly** property is **True**, the form has been placed in a read-only state. Changes can still be made to the form, but it cannot be saved using a save operation, it must be saved using a save-as operation.

To determine whether the form's underlying XML document has been placed in a read-only state, use the [IsDOMReadOnly](#) property of the **XDocument** object.

Example

In the following example, the **IsReadOnly** property of the **XDocument** object is used to determine whether the form is in a read-only state:

```
if (XDocument.IsReadOnly)
    XDocument.UI.Alert("The form cannot be modified.");
else
    return;
```

IsRecovered Property

A read-only property that returns a **Boolean** value that indicates whether a Microsoft Office InfoPath 2003 form was last saved by an AutoRecover save operation.

expression.**IsRecovered**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

The following example shows how to use the **IsRecovered** property to indicate, when the form is opened, whether the form was last saved by an AutoRecover save operation:

```
function XDocument::OnLoad(eventObj)
{
    XDocument.UI.Alert("Last saved by an AutoRecover save operation")
}
```

IsSaveAs Property

A read-only property that returns a **Boolean** value that indicates whether the [PerformSaveOperation](#) method of the [SaveEvent](#) object will be performed as a "save" operation or as a "save as" operation.

expression.**IsSaveAs**

expression Required. An expression that returns a reference to a **SaveEvent** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **IsSaveAs** property returns **true** when the event was triggered by a call to the [SaveAs](#) method, or by a call to the [Save](#) method when the document is new, or by a call to either method when the document is read-only. Otherwise, the **IsSaveAs** property returns **False**.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **IsSaveAs** property of the **SaveEvent** object is used to determine if the user will be presented with a **Save As** dialog box; otherwise, it will inform the user that the form will simply be saved.

```
        if (!eventObj.IsSaveAs)
    {
        XDocument.UI.Alert("About to save this form.");
    }
    eventObj.PerformSaveOperation();
```


IsSigned Property

A read-only property that returns a Boolean value that indicates whether a Microsoft Office InfoPath 2003 form has been digitally signed using [digital signatures](#).

expression.**IsSigned**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the **IsSigned** property is **True**, the form has been digitally signed. If **False**, the form has not been digitally signed.

InfoPath uses [XML Signatures](#) to digitally sign forms.

Note If a form has been digitally signed, its underlying XML document is placed in a read-only state.

Example

In the following example, the **IsSigned** property of the **XDocument** object is used to determine whether a form has been digitally signed:

```
if (XDocument.IsSigned)
    XDocument.UI.Alert("This form contains digital signatures.");
else
    return;
```


IssuedBy Property

A read-only property that returns the issuer of the [digital certificate](#).

expression.**IssuedBy**

expression Required. An expression that returns a [Certificate](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

IssuedTo Property

A read-only property that returns a string representing to whom the [digital certificate](#) is issued.

expression.**IssuedTo**

expression Required. An expression that returns a [Certificate](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

IsUndoRedo Property (DataDOMEvent Object)

A read-only property that returns a Boolean value indicating whether the [data validation](#) event occurs during an undo operation or a redo operation.

expression.**IsUndoRedo**

expression Required. Returns a reference to the [DataDOMEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

When an undo or a redo operation takes place, the form's underlying XML document is placed in a read-only state and cannot be modified. This can sometimes occur during an [OnAfterChange](#) event ; when it does, the **IsUndoRedo** property is used to bypass the script-based data validation that it contains.

Example

In the following example from the Data Validation developer sample form, the **IsUndoRedo** property is used to determine whether the event is occurring during an undo or redo operation. If it is, the event handler is exited using the `return;` statement.

```
function msoxd__itemB_quantityListB::OnAfterChange(eventObj)
{
    if (eventObj.IsUndoRedo)
    {
        // An undo or redo operation has occurred and the DOM is read-only
        return;
    }
    XDocument.DOM.selectSingleNode("/sampleData/listB/total").text =
        GetTotal("/sampleData/listB/itemB/quantityListB");
}
```

IsUndoRedo Property (DocContextChangeEvent Object)

A read-only property that returns a **Boolean** value indicating whether the context change event occurred in response to undoing operation or redoing an operation.

expression.IsUndoRedo

expression Required. Returns a reference to a [DocContextChangeEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

IsUndoRedo Property (Index)

This property is implemented in several Microsoft Office InfoPath 2003 object model collections. Click an **IsUndoRedo** property link below to view the Help topic for a specific implementation of the **IsUndoRedo** property.

[IsUndoRedo](#) property as it applies to the [DataDOMEvent](#) object.

[IsUndoRedo](#) property as it applies to the [DocContextChangeEvent](#) object.

Item Property (DataAdapters Collection)

A read-only property that returns a reference to the specified data adapter object from the [DataAdaptersCollection](#) collection, based on position or name.

```
expression.Item(ByVal varIndex As Variant) As Object
```

expression Required. An expression that returns a reference to the **DataAdapters** collection.

varIndex Required **Variant**. An expression that specifies the position of a member of the **DataAdapters** collection. If the argument is a [numeric expression](#), it must be a number from 0 to the value of the collection's [Count](#) property minus 1. If the argument is a [string expression](#), it must be the name of a member of the collection.

returns A reference to a data adapter object.

Security Level

0: Can be accessed without restrictions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **Item** property of the **DataAdapters** collection is used to return a reference to a data adapter in the collection:

```
var objDataAdapter;  
objDataAdapter = XDocument.DataAdapters.Item(0);
```

Because the **Item** property is the default property of the **DataAdapters** collection, it can also be used as follows:

```
var objDataAdapter;  
objDataAdapter = XDocument.DataAdapters(0);
```

You can also use the name of a data adapter as the argument to the **Item** method, as shown in the following example:

```
var objDataAdapter;  
objDataAdapter = XDocument.DataAdapters("MyDataAdapter");
```


Item Property (DataObjects Collection)

A read-only property that returns a reference to the specified [DataObject](#) object from the [DataObjects](#) collection.

```
expression.Item(ByVal varIndex As Variant) As DataObject
```

expression Required. An expression that returns a reference to the **DataObjects** collection.

varIndex Required **Variant**. An expression that specifies the position of a member of the **DataObjects** collection. If a [numeric expression](#), the argument must be a number from 0 to the value of the collection's [Count](#) property minus 1. If a [string expression](#), the argument must be the name of a member of the collection.

returns A reference to a **DataObject** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the value provided for the *varIndex* argument does not match any existing member of the collection, an error occurs.

After you have set a reference to the **DataObject** object that the **Item** property returns, you can access any of its properties or methods.

Example

In the following example, the **Item** property of the **DataObjects** collection is used to return a reference to a **DataObject** object:

```
var objDataObject;  
objDataObject = XDocument.DataObjects.Item(0);
```

Because the **Item** property is the default property of the **DataObjects** collection, it can also be used as follows:

```
var objDataObject;  
objDataObject = XDocument.DataObjects(0);
```

You can also use the name of the **DataObject** object as the argument to the **Item** method, as shown in the following example:

```
var objDataObject;  
objDataObject = XDocument.DataObjects("MyDataObject");
```

Item Property (Errors Collection)

A read-only property that returns a reference to the specified [Error](#) object from the [Errors](#) collection.

```
expression.Item(ByVal index As Long) As Error
```

expression Required. An expression that returns a reference to the **Errors** collection.

index Required **Long Integer**. An expression that specifies the position of a member of the **Errors** collection. The argument must be a number from 0 to the value of the collection's [Count](#) property minus 1.

returns A reference to an **Error** object.

Security Level

0: Can be accessed without restrictions.

Remarks

If the value provided for the *index* argument does not match any existing member of the collection, an error occurs.

After you have set a reference to the **Error** object that the **Item** property returns, you can access any of its properties or methods.

Example

In the following example, the **Item** property of the **Errors** collection is used to return a reference to an **Error** object:

```
var objError;  
objError = XDocument.Errors.Item(0);
```

Because the **Item** property is the default property of the **Errors** collection, it can also be used as follows:

```
var objError;  
objError = XDocument.Errors(0);
```

Item Property (Index)

The **Item** property returns a reference to an object contained in a collection. This property is implemented in several Microsoft Office InfoPath 2003 object model collections. Click an **Item** property link below to view the Help topic for a specific implementation of the **Item** property.

[Item](#) property as it applies to the [DataAdapters](#) collection.

[Item](#) property as it applies to the [DataObjects](#) collection.

[Item](#) property as it applies to the [Errors](#) collection.

[Item](#) property as it applies to the [Signatures](#) collection.

[Item](#) property as it applies to the [SignedDataBlocks](#) collection.

[Item](#) property as it applies to the [TaskPanels](#) collection.

[Item](#) property as it applies to the [ViewInfos](#) collection.

[Item](#) property as it applies to the [Windows](#) collection.

[Item](#) property as it applies to the [XDocuments](#) collection.

[Item](#) property as it applies to the [XMLNodes](#) collection.

Item Property (Signatures Collection)

A read-only property that returns a reference to the specified [digital signature](#) from the [Signatures](#) collection.

```
expression.Item(ByVal varIndex As Variant) As Signature
```

expression Required. An expression that returns a reference to the **Signatures** collection.

varIndex Required **Variant**. A numeric expression that specifies the position of a member of the **Signatures** collection. The argument must be a number from 0 to the value of the collection's **Count** property minus 1.

returns A reference to a digital signature.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Item Property (SignedDataBlocks Collection)

A read-only property that returns a reference to the specified [SignedDataBlock](#) object from the [SignedDataBlocks](#) collection, based on position or name.

```
expression.Item(ByVal varIndex As Variant) As SignedDataBlock
```

expression Required. An expression that returns a [SignedDataBlocks](#) collection.

varIndex Required **Variant**. An expression that specifies the position of a member of the SignedDataBlocks collection. If this argument is a numeric expression, it must be a number from 0 to the value of the collection's **Count** property minus 1. If this argument is a string expression, it must be the name of a member of the collection.

returns A reference to a SignedDataBlock object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Item Property (TaskPanes Collection)

A read-only property that returns a reference to the specified [TaskPane](#) object from the [TaskPanes](#) collection.

```
expression.Item(ByVal varIndex As Variant) As TaskPane
```

expression Required. Returns a reference to the **TaskPanes** collection.

varIndex Required **Variant**. A [numeric expression](#) that specifies the type of task pane. Based on the [XdTaskPaneType](#) enumeration.

returns A reference to a **TaskPane** object.

Security Level

0: Can be accessed without restrictions.

Remarks

If the value provided for the *varIndex* argument does not match any existing member of the collection, an error occurs.

After you have set a reference to the **TaskPane** object that the **Item** property returns, you can access any of its properties and methods.

Example

In the following example, the **Item** property of the **TaskPanels** collection is used to return a reference to a specified **TaskPane** object:

```
var objTaskPane;
```

```
objTaskPane = XDocument.View.Window.TaskPanels.Item(4);
```

Because the **Item** property is the default property of the **TaskPanels** collection, it can also be used as follows:

```
var objTaskPane;
```

```
objTaskPane = XDocument.View.Window.TaskPanels(4);
```


Item Property (ViewInfos Collection)

A read-only property that returns a reference to the specified [ViewInfo](#) object from the [ViewInfos](#) collection.

```
expression.Item(ByVal varIndex As Variant) As ViewInfo
```

expression Required. An expression that returns a reference to the **ViewInfos** collection.

varIndex Required **Variant**. An expression that specifies the position of a member of the **ViewInfos** collection. If a [numeric expression](#), the argument must be a number from 0 to the value of the collection's [Count](#) property minus 1. If a [string expression](#), the argument must be the name of a member of the collection.

returns A reference to a **ViewInfo** object.

Security Level

0: Can be accessed without restrictions.

Remarks

If the value provided for the *varIndex* argument does not match any existing member of the collection, an error occurs.

After you have set a reference to the **ViewInfo** object that the **Item** property returns, you can access any of its properties.

Example

In the following example, the **Item** property of the **ViewInfos** collection is used to return a reference to a **ViewInfo** object:

```
var objViewInfo;  
objViewInfo = XDocument.ViewInfos.Item(0);
```

Because the **Item** property is the default property of the **ViewInfos** collection, it can also be used as follows:

```
var objViewInfo;  
objViewInfo = XDocument.ViewInfos(0);
```

You can also use the name of the **ViewInfo** object, which is the name of a view, as the argument to the **Item** method, as shown in the following example:

```
var objViewInfo;  
objViewInfo = XDocument.ViewInfos("MyView");
```

Item Property (Windows Collection)

A read-only property that returns a reference to the specified [Window](#) object from the [Windows](#) collection.

```
expression.Item(ByVal varIndex As Variant) As Window
```

expression Required. An expression that returns a reference to the **Windows** collection.

varIndex Required **Variant**. An expression that specifies the position of a member of the **Windows** collection. The argument must be a number from 0 to the value of the collection's count property minus 1.

returns A reference to a **Window** object.

Security Level

0: Can be accessed without restrictions.

Remarks

If the value provided for the ***varIndex*** argument does not match any existing member of the collection, an error occurs.

After you have set a reference to the **Window** object that the **Item** property returns, you can access any of its properties and methods.

Example

In the following example, the **Item** property of the **Windows** collection is used to return a reference to a **Window** object:

```
var objWindow;  
objWindow = Application.Windows.Item(0);
```

Because the **Item** property is the default property of the **Windows** collection, it can also be used as follows:

```
var objWindow;  
objWindow = Application.Windows(0);
```


Item Property (XDocuments Collection)

A read-only property that returns a reference to the specified [XDocument](#) object from the [XDocuments](#) collection.

```
expression.Item(ByVal varIndex As Variant) As XDocument
```

expression Required. An expression that returns a reference to the **XDocuments** collection.

varIndex Required **Variant**. An expression that specifies the position of a member of the **XDocuments** collection. If a [numeric expression](#), the argument must be a number from 0 to the value of the collection's [Count](#) property minus 1. If a [string expression](#), the argument must be the [Uniform Resource Locator \(URL\)](#) path of a member of the collection.

returns A reference to an **XDocument** object.

Security Level

0: Can be accessed without restrictions.

Remarks

If the value provided for the *varIndex* argument does not match any existing member of the collection, an error occurs.

After you have set a reference to the **XDocument** object that the **Item** property returns, you can access any of its properties or methods.

Example

In the following example, the **Item** property of the **XDocuments** collection is used to return a reference to an **XDocument** object:

```
var objXDoc;  
objXDoc = Application.XDocuments.Item(0);
```

Because the **Item** property is the default property of the **XDocuments** collection, it can also be used as follows:

```
var objXDoc;  
objXDoc = Application.XDocuments(0);
```

You can also use the name of the **XDocument** as the argument to the **Item** method, as shown in the following example:

```
var objXDoc;  
objXDoc = Application.XDocuments("MyForm");
```


Item Property (XMLNodes Collection)

A read-only property that returns a reference to the specified XML Document Object Model (DOM) node from the [XMLNodes](#) collection.

```
expression.Item(ByVal varIndex As Variant) As IXMLDOMNode
```

expression Required. Returns a reference to the **XMLNodes** collection.

varIndex Required **Variant**. A [numeric expression](#) that specifies the position of a member of the **XMLNodes** collection. The argument must be a number from 0 to the value of the collection's count property minus 1.

returns A reference to an XML DOM node object.

Security Level

0: Can be accessed without restrictions.

Remarks

If the value provided for the ***varIndex*** argument does not match any existing member of the collection, an error occurs.

After you have set a reference to the XML DOM node object that the **Item** property returns, you can access any of its properties and methods.

Note To learn more about the XML DOM and all of the properties and methods that it supports, see the MSXML 5.0 SDK documentation in the [Microsoft Script Editor \(MSE\)](#) Help system.

Example

In the following example, the **Item** property of the **XMLNodes** collection is used to return a reference to an XML DOM node object:

```
var objXMLNodes;  
var objXMLNode;
```

```
objXMLNodes = XDocument.View.GetContextNodes();  
objXMLNode = objXMLNodes.Item(0);
```

Because the **Item** property is the default property of the **XMLNodes** collection, it can also be used as follows:

```
var objXMLNodes;  
var objXMLNode;
```

```
objXMLNodes = XDocument.View.GetContextNodes();  
objXMLNode = objXMLNodes(0);
```

Language Property

Sets or retrieves a Microsoft Office InfoPath 2003 form's default language code.

expression.**Language**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The language settings for an InfoPath form can also be accessed using the [LanguageSettings](#) property of the [Application](#) object.

Example

In the following example, the **Language** property of the **XDocument** object is used to display a form's current language setting in a message box:

```
XDocument.UI.Alert("The current language is: " + XDocument.Langu
```


LanguageSettings Property

A read-only property that returns a reference to the Microsoft Office [LanguageSettings](#) object.

expression.**LanguageSettings**

expression An expression that returns a reference to the [Application](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

The **LanguageSettings** property is available only when using scripting code.

After you establish a reference to the **LanguageSettings** object, you can access all the properties and methods of the object.

Example

The following example uses the [LanguageID](#) property of the **LanguageSettings** object to return the LCID value (a four-digit number) for the language that is currently being used for the Office help system:

```
Application.LanguageSettings.LanguageID(3);
```

Note Because Microsoft Office InfoPath 2003 uses scripting code instead of [Visual Basic for Applications \(VBA\)](#), you cannot use the names of enumerated values; you must use the numerical values of the enumerations, as in the previous example (`msoLanguageIDHelp = 3`).

Left Property

A read/write property of type **long integer** that specifies the horizontal position of the window represented by the [Window](#) object, measured in points.

expression.**Left**

expression Required. An expression that returns a reference to a **Window** object.

Remarks

This property will return an error if it is set on a window that is minimized or maximized.

Setting this property to a position that is off the screen, will actually cause the window to be displayed on the screen.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

MachineOnlineState Property

A read-only property of type [XdMachineOnlineState](#) that returns the current connection state of the client computer.

expression.**MachineOnlineState**

expression Required. An expression that returns a reference to the [Application](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

MailEnvelope Property

A read-only property that returns a reference to the [MailEnvelope](#) object associated with the window that is represented by the [Window](#) object.

expression.**MailEnvelope**

expression Required. An expression that returns a reference to the **Window** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **MailEnvelope** property can be used only with the editing window types; if used with a designing window type, it will return an error. It will also return an error if no form is open in the [form area](#).

Example

In the following example, the **MailEnvelope** property of the **Window** object is used to set a reference to the **MailEnvelope** object, which is then used to create and display a custom e-mail message:

```
var objEmail;
```

```
objEmail = Application.ActiveWindow.MailEnvelope;
```

```
objEmail.Subject = "Test e-mail message";
```

```
objEmail.Visible = true;
```


MatchExpression Property

A read-only property that returns a string value containing the [XPath](#) expression for the XML [Document Object Model \(DOM\)](#) node for which the event is currently being processed.

expression.MatchExpression

expression Required. Returns a reference to the [DataDOMEvent](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

The XPath expression that the **MatchExpression** property contains points to an XML DOM node in the form's underlying XML document. This is the node that is currently being validated by the data validation [event handler](#).

Example

In the following example, the **MatchExpression** property of the **DataDOMEEvent** object is used to display the XPath expression of the XML DOM node that is currently being validated:

```
XDocument.UI.Alert("Match expression: " + eventObj.MatchExpress
```

Math Property

A read-only property that returns a reference to the [Math](#) object.

expression.**Math**

expression Required. An expression that returns a reference to the [Util](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

Using the **Math** property, you can access all of the methods that the **Math** object provides.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Name Property (ADOAdapter Object)

A read-only property that returns the name of an [ADOAdapter](#) object.

expression.Name

expression Required. An expression that returns a reference to an **ADOAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Name Property (Application Object)

A read-only property that returns a string containing the name of the Microsoft Office InfoPath 2003 application.

expression.Name

expression Required. An expression that returns a reference to the [Application](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

The name of the application does not contain the version number. To obtain the version number of an application, use the [Version](#) property of the **Application** object.

Example

In the following example, the [Alert](#) method of the [UI](#) object, accessed through the [XDocument](#) object, is used to display a message box that indicates the name of the application:

```
XDocument.UI.Alert("Application name: " + Application.Name);
```


Name Property (DataObject Object)

A read-only property that returns a string value indicating the name of the associated [DataObject](#) object.

expression.Name

expression Required. An expression that returns a reference to the **DataObject** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The name of the **DataObject** object is the same as the name of the [secondary data source](#) that it represents. The name of the **DataObject** object can also be used as the argument to the [Item](#) property of the [DataObjects](#) collection.

Example

In the following example, the **Name** property of the **DataObject** object is used to to display the name of the **DataObject** object in a message box:

```
var objDataObject;
```

```
objDataObject = XDocument.DataObjects(0);
```

```
XDocument.UI.Alert("DataObject name: " + objDataObject.Name);
```

Name Property (DAVAdapter Object)

A read-only property that returns the name of a [DAVAdapter](#) object.

expression.Name

expression Required. An expression that returns a reference to a **DAVAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Name Property (EmailAdapter Object)

A read-only property that returns the name of an [EmailAdapter](#) object.

expression.Name

expression Required. An expression that returns a reference to an **EmailAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Name Property (HWSAdapter Object)

A read-only property that returns the name of an [HWSAdapter](#) object.

expression.Name

expression Required. An expression that returns a reference to a **HWSAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Name Property (Index)

The **Name** property returns a string that specifies the name of an object. This property is implemented in several Microsoft Office InfoPath 2003 object model objects. Click a **Name** property link below to view the Help topic for a specific implementation of the **Name** property.

[Name](#) property as it applies to the [ADOAdapter](#) object.

[Name](#) property as it applies to the [Application](#) object.

[Name](#) property as it applies to the [DataObject](#) object.

[Name](#) property as it applies to the [DAVAdapter](#) object.

[Name](#) property as it applies to the [EmailAdapter](#) object.

[Name](#) property as it applies to the [HWSAdapter](#) object.

[Name](#) property as it applies to the [SharepointListAdapter](#) object.

[Name](#) property as it applies to the [SignedDataBlock](#) object.

[Name](#) property as it applies to the [View](#) object.

[Name](#) property as it applies to the [ViewInfo](#) object.

[Name](#) property as it applies to the [WebServiceAdapter](#) object.

[Name](#) property as it applies to the [XMLFileAdapter](#) object.

Name Property (SharePointListAdapter Object)

A read-only property that returns the name of a [SharepointListAdapter](#) object.

expression.Name

expression Required. An expression that returns a reference to a **SharePointListAdapter** object

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Name Property (SignedDataBlock Object)

A read-only property that returns the name of a [SignedDataBlock](#) object.

expression.Name

expression Required. An expression that returns a reference to a **SignedDataBlock** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Name Property (View Object)

A read-only property that returns a string containing the name of the view that is represented by the [View](#) object.

expression.Name

expression Required. An expression that returns a reference to the **View** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

To determine whether a view is the default view, use the [ViewInfo](#) object.

Example

In the following example from the Events developer sample form, the **Name** property of the **View** object is used to determine which view the user has switched to. If the view is the Archive Customer view, a note value is added to the form's underlying XML document:

```
function XDocument::OnSwitchView(eventObj)
{
    var oDate = new Date();

    if (XDocument.View.Name == "Archive Customer")
    {
        var oNotesNode = XDocument.DOM
            .selectSingleNode("/Customers/CustomerInfo/Notes");
        var oDivNode = XDocument.DOM
            .createNode(1, "div", "http://www.w3.org/1999/xhtml");

        oDivNode.text = "Note recorded " + oDate.toString();
        oNotesNode.appendChild(oDivNode);
    }
}
```

Name Property (ViewInfo Object)

A read-only property that returns a string value indicating the name of the view that is associated with the [ViewInfo](#) object.

expression.Name

expression Required. An expression that returns a reference to the **ViewInfo** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The name of the **ViewInfo** object is the same as the name of the view that it represents. The name of the **ViewInfo** object can also be used as the argument to the [Item](#) property of the [ViewInfos](#) collection.

Example

In the following example, the **Name** property of the **ViewInfo** object is used to to display the name of a view in a message box:

```
var objViewInfo;
```

```
objViewInfo = XDocument.ViewInfos(0);
```

```
XDocument.UI.Alert("View name: " + objViewInfo.Name);
```

Name Property (WebServiceAdapter Object)

A read-only property that returns the name of a [WebServiceAdapter](#) object.

expression.Name

expression Required. An expression that returns a reference to a **WebServiceAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Name Property (XMLFileAdapter Object)

A read-only property that returns the name of an [XMLFileAdapter](#) object.

expression.Name

expression Required. An expression that returns a reference to an **XMLFileAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

NewValue Property

A read-only property that returns a string value indicating the value of an XML [Document Object Model \(DOM\)](#) node that is being updated or inserted during a [data validation](#) event.

expression.**NewValue**

expression Required. Returns a reference to the [DataDOMEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **NewValue** property contains the value of the XML DOM node that will replace the existing value. To get the original value of the XML DOM node, use the [OldValue](#) property of the **DataDOMEvent** object.

Example

In the following example from the ADO (ActiveX Data Objects) developer sample form, the **NewValue** property of the **DataDOMEvent** object is used to determine whether the new value of the XML DOM node is an empty string. If it is not, the code sets other fields to be empty strings.

```
function msoxd__Employees_EmployeeID_attr::OnAfterChange(eventObj)
{
    if (eventObj.IsUndoRedo)
    {
        // An undo or redo operation has occurred and the DOM is read-only
        return;
    }

    if (eventObj.NewValue == "")
        return;

    if (XDocument.DOM.selectSingleNode
        ('/dfs:myFields/dfs:queryFields/q:Employees/@FirstName').text !=
        XDocument.DOM.selectSingleNode
        ('/dfs:myFields/dfs:queryFields/q:Employees/@FirstName').text)
        return;

    if (XDocument.DOM.selectSingleNode
        ('/dfs:myFields/dfs:queryFields/q:Employees/@LastName').text !=
        XDocument.DOM.selectSingleNode
        ('/dfs:myFields/dfs:queryFields/q:Employees/@LastName').text)
        return;
}
```


Node Property (Error Object)

A read-only property that returns a reference to the XML [Document Object Model \(DOM\)](#) node of a form's underlying XML document that is associated with an [Error](#) object.

expression.**Node**

expression Required. An expression that returns a reference to the **Error** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Using the **Node** property, you can access all of the properties and methods that the XML DOM node object provides.

Example

In the following example, the **Node** property of the **Error** object is used to display the XML node of an error in a message box:

```
var objError;
```

```
objError = XDocument.Errors(0);
```

```
XDocument.UI.Alert("Error name: " + objError.Node.xml);
```


Node Property (Index)

The **Node** property returns a reference to the XML [Document Object Model \(DOM\)](#) node of a form's underlying XML document that is associated with a particular object. This property is implemented in several Microsoft Office InfoPath 2003 object model objects. Click a **Node** property link below to view the Help topic for a specific implementation of the **Node** property.

[Node](#) property as it applies to the [Error](#) object.

[Node](#) property as it applies to the [InfoPathControlSite](#) object.

Node Property (InfoPathControlSite Object)

Retrieves a reference to the XML node to which the ActiveX control is bound.

expression.Node

expression Required. An expression that returns a reference to the [InfoPathControlSite](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The [InfoPathControl](#) and **InfoPathControlSite** objects and their methods and properties are designed to be used only from the implementation of an ActiveX control. These objects and their members are not supported in InfoPath form code. For more information on how to create ActiveX controls that work with InfoPath, see the [InfoPath Developer Center](#).

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form template that contains a view with an ActiveX control that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

OldValue Property

A read-only property that returns a string value indicating the original value of an XML [Document Object Model \(DOM\)](#) node that is being updated or deleted during a [data validation](#) event.

expression.**OldValue**

expression Required. Returns a reference to the [DataDOMEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **OldValue** property contains the original value of the XML DOM node that will be replaced with a new value or deleted. To get the new value of the XML DOM node, use the [NewValue](#) property of the **DataDOMEvent** object.

Example

In the following example, the **OldValue** property of the **DataDOMEvent** object is used to display the original value of an XML DOM node, along with its new value:

```
XDocument.UI.Alert("Original value: " + eventObj.OldValue +  
    "\nNew value: " + eventObj.NewValue);
```


Operation Property (DataDOMEvent Object)

A read-only property that returns a string value indicating the type of action that is applied to an XML [Document Object Model \(DOM\)](#) node during a [data validation](#) event.

expression.**Operation**

expression Required. Returns a reference to the [DataDOMEvent](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

The values that the **Operation** property returns include Insert, Update, and Delete.

Example

In the following example from the Structural Editing developer sample form, the **Operation** property of the **DataDOMEvent** object is used to check the type of action that is occurring; if it is a delete action, the code calls a custom function:

```
function msoxd__item::OnAfterChange(eventObj)
{
    if (!eventObj.IsUndoRedo && eventObj.Operation == "Delete"
        && eventObj.Site.nodeName == "item" &&
        eventObj.Source.nodeName == "item")
    {
        Calculate();
    }
}
```

Operation Property (Index)

The **Operation** property either specifies or returns a string that represents a Web service command string, or returns the name of an event action. This property is implemented in several Microsoft Office InfoPath 2003 object model objects. Click an **Operation** property link below to view the Help topic for a specific implementation of the **Operation** property.

[Operation](#) property as it applies to the [DataDOMEvent](#) object.

[Operation](#) property as it applies to the [WebServiceAdapter](#) object.

Operation Property (WebServiceAdapter Object)

Sets or retrieves a string value that contains the source XML of the [operation](#) element contained in the [form definition \(.xsf\) file](#).

expression.**Operation**

expression Required. An expression that returns a reference to the [WebServiceAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **operation** element of the .xsf file contains information about the Web service, including its name, the method used for retrieving and submitting data, and its [Uniform Resource Locator \(URL\)](#).

Example

In the following example, the **Operation** property of the **WebServiceAdapter** object is used to display the operation string of the Web service data adapter in a message box:

```
var objWSAdapter;
```

```
objWSAdapter = XDocument.DataObjects("WebCityList").QueryAdapter  
XDocument.UI.Alert("Operation string: " + objWSAdapter.Operation)
```


OutputLocation Property

A read/write property that specifies the XML [Document Object Model \(DOM\)](#) node under which the adapter will copy the XML returned by the XML Web service.

expression.**OutputLocation**

expression Required. An expression that returns a reference to a [WebServiceAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the value of the **OutputLocation** property is not set on a query adapter, it will correspond to the **dataFields** element in the data source associated with the data connection.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

PackageURL Property

A read-only property that returns a string containing the [Uniform Resource Locator \(URL\)](#) of the cache folder that contains a Microsoft Office InfoPath 2003 form's extracted [form files](#).

expression.**PackageURL**

expression Required. An expression that returns a reference to a [Solution](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **PackageURL** property is set at run time, and the URL it contains points to the folder where the form template's files are cached.

Example

In the following example, the **PackageURL** property of the **Solution** object is used to display the folder's URL in a message box:

```
XDocument.UI.Alert("PackageURL: " + XDocument.Solution.Packag
```


Parent Property

A read-only property that returns a reference to the XML [Document Object Model \(DOM\)](#) node of the parent of the XML DOM node being changed during a [data validation](#) event.

expression.**Parent**

expression Required. Returns a reference to the [DataDOMEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

After you have set a reference to the XML DOM node that the **Parent** property returns, you can use any of the properties and methods that are supported by the XML DOM. This can be especially useful during delete operations, because the **Parent** property maps to the location of the XML DOM node that was removed.

Note To learn more about the XML DOM and all of the properties and methods that it supports, see the MSXML 5.0 SDK documentation in the [Microsoft Script Editor \(MSE\)](#) Help system.

Example

In the following partial example from the Data Validation developer sample form, the **Parent** property of the **DataDOMEvent** object is used to check the name of the parent node; if it matches certain criteria, an error message is displayed:

```
function msoxd__shippingDates::OnBeforeChange(eventObj)
{
    var objOrderDate = new Date(XDocument.DOM.selectSingleNode
        ('/sampleData/shippingDates/orderDate')
        .text.replace(/(.*)-(.*)-(.*)/, "$2-$3-$1"));
    var objShipDate = new Date(XDocument.DOM.selectSingleNode
        ('/sampleData/shippingDates/shipDate')
        .text.replace(/(.*)-(.*)-(.*)/, "$2-$3-$1"));
    ...

    if (objShipDate.toString() != "NaN" && objOrderDate.toString() ==
    {
        eventObj.ReturnMessage = "The Ship Date is invalid without an or

        if (eventObj.Parent.nodeName == "orderDate")
            eventObj.ReturnMessage += " You must delete the Ship Date " +
            "before deleting the Order Date.";

        eventObj.ReturnStatus = false;
        return;
    }
    ...
}
```


QueryAdapter Property (DataObject Object)

A read-only property that returns a reference to the [data adapter](#) object that is used for a [secondary data source](#).

expression.**QueryAdapter**

expression Required. An expression that returns a reference to the [DataObject](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

After you have set a reference to the data adapter object that the **QueryAdapter** property returns, you can use the properties and methods that the particular data adapter object contains.

Microsoft Office InfoPath 2003 supports three types of data adapters:

1. **ActiveX Data Objects** Represented by the [ADOAdapter](#) object.
2. **Web services** Represented by the [WebServiceAdapter](#) object.
3. **XML file** Represented by the [XMLFileAdapter](#) object.

Example

In the following example, the **QueryAdapter** property of the **DataObject** object is used to return a reference to the data adapter that is associated with the **DataObject** object, which, in this case, is an **ADOAdapter** data adapter object. The code then uses the **Command** property of the **ADOAdapter** object to display the SQL command text in a message box:

```
var objDataAdapter;
```

```
objDataAdapter = XDocument.DataObjects("CityList").QueryAdapte  
XDocument.UI.Alert("SQL command text: " + objDataAdapter.Comm
```

QueryAdapter Property (Index)

The **QueryAdapter** property returns a reference to a type of data adapter object such as the [ADOAdapter](#), [WebServiceAdapter](#), or [XMLFileAdapter](#) object. This property is implemented in a number of Microsoft Office InfoPath 2003 object model objects. Click a **QueryAdapter** property link below to view the Help topic for a specific implementation of the **QueryAdapter** property.

[QueryAdapter](#) property as it applies to the [DataObject](#) object.

[QueryAdapter](#) property as it applies to the [XDocument](#) object.

QueryAdapter Property (XDocument Object)

A read-only property that returns a reference to the [data adapter](#) object that is associated with a Microsoft Office InfoPath 2003 form.

expression.**QueryAdapter**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Data adapter objects provide different properties and methods that retrieve and submit data to external data sources; the data adapter that is associated with a form is dependent on the type of data source that was used when the form was initially created.

The **QueryAdapter** property allows you to access an InfoPath form's primary data source. To access the data adapter objects used for a form's [secondary data sources](#), use the [DataObjects](#) property of the **XDocument** object.

Microsoft Office InfoPath 2003 supports three types of data adapters:

1. **ActiveX Data Objects** Represented by the [ADOAdapter](#) object.
2. **Web services** Represented by the [WebServiceAdapter](#) object.
3. **XML file** Represented by the [XMLFileAdapter](#) object.

Note The **XMLFileAdapter** object cannot be used with the **QueryAdapter** property of the **XDocument** object, it is only used for secondary data sources. To access a form's underlying XML document, use the [DOM](#) property of the **XDocument** object.

Example

In the following example, the **QueryAdapter** property of the **XDocument** object is used to set a reference to the [ADOAdapter](#) data adapter object; then the [Command](#) property of the **ADOAdapter** object is used to display the SQL command text in a message box:

```
var objADOAdapter;
```

```
objADOAdapter = XDocument.QueryAdapter;
```

```
XDocument.UI.Alert("SQL command text: " + objADOAdapter.Comn
```


QueryAllowed Property (ADOAdapter Object)

A read-only property of type **Boolean** that always returns **True**, corresponding to the [queryAllowed](#) attribute in the [form definition file \(.xsf\)](#).

expression.**QueryAllowed**

expression An expression that returns an [ADOAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

QueryAllowed Property (DAVAdapter Object)

A read-only property of type **Boolean** that always returns **False**, corresponding to the **queryAllowed** attribute in the [form definition file \(.xsf\)](#).

expression.**QueryAllowed**

expression An expression that returns a [DAVAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

QueryAllowed Property (EmailAdapter Object)

A read-only property of type **Boolean** that always returns **False**, corresponding to the **queryAllowed** attribute in the [form definition file \(.xsf\)](#).

expression.**QueryAllowed**

expression An expression that returns an [EmailAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

QueryAllowed Property (HWSAdapter Object)

A read-only property that returns a **Boolean** value that is always **False**, corresponding to the **queryAllowed** attribute in the [form definition file \(.xsf\)](#).

expression.**QueryAllowed**

expression An expression that returns an [HWSAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

QueryAllowed Property (Index)

The **QueryAllowed** property returns a value that corresponds to the **queryAllowed** attribute in the [form definition file \(.xsf\)](#). This property is implemented in several Microsoft Office InfoPath 2003 object model objects. Click a **QueryAllowed** property link below to view the Help topic for a specific implementation of the **QueryAllowed** property.

[QueryAllowed](#) property as it applies to the [ADOAdapter](#) object.

[QueryAllowed](#) property as it applies to the [DAVAdapter](#) object.

[QueryAllowed](#) property as it applies to the [EmailAdapter](#) object.

[QueryAllowed](#) property as it applies to the [HWSAdapter](#) object.

[QueryAllowed](#) property as it applies to the [SharepointListAdapter](#) object.

[QueryAllowed](#) property as it applies to the [WebServiceAdapter](#) object.

[QueryAllowed](#) property as it applies to the [XMLFileAdapter](#) object.

QueryAllowed Property (SharePointListAdapter Object)

A read-only property of type **Boolean** that always returns **True**, corresponding to the **queryAllowed** attribute in the [form definition file \(.xsf\)](#).

expression.**QueryAllowed**

expression An expression that returns a [SharepointListAdapterObject](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

QueryAllowed Property (WebServiceAdapter Object)

A read-only property of type **Boolean** that corresponds to the [queryAllowed](#) attribute in the [form definition file \(.xsf\)](#). The default value is **False**.

expression.**QueryAllowed**

expression An expression that returns a [WebServiceAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

QueryAllowed Property (XMLFileAdapter Object)

A read-only property of type **Boolean** that always returns **True**, corresponding to the **queryAllowed** attribute in the [form definition file \(.xsf\)](#).

expression.**QueryAllowed**

expression An expression that returns an [XMLFileAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

ReturnMessage Property

Sets or retrieves a string value indicating the error message that will be returned if the [data validation](#) event is not successful.

expression.ReturnMessage

expression Required. Returns a reference to the [DataDOMEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Used in conjunction with the [ReturnStatus](#) property of the **DataDOMEEvent** object, the **ReturnStatus** property displays a message box to the user with the specified text message.

Example

In the following example, the **ReturnMessage** property of the **DataDOMEvent** object is used to display a message to the user if the data validation for the XML DOM node fails:

```
function msoxd__id_attr::OnBeforeChange(eventObj)
{
    if (eventObj.NewValue == "")
    {
        eventObj.ReturnMessage = "You must supply a value for this field";
        eventObj.ReturnStatus = false;
        return;
    }
}
```


ReturnStatus Property (DataDOMEvent Object)

Sets or retrieves a Boolean value indicating the return status of the [data validation](#) event.

expression.ReturnStatus

expression Required. Returns a reference to the [DataDOMEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the **ReturnStatus** property is set to False, the changes to the XML [Document Object Model \(DOM\)](#) node are not accepted and the data validation event fails. If set to True, no data validation error has occurred and the data validation event is successful.

Example

In the following example, the **ReturnStatus** property of the **DataDOMEEvent** object is used to indicate that the data validation event was not successful. The code also uses the [ReturnMessage](#) property of the **DataDOMEEvent** object to display a message to the user.

```
function msoxd__id_attr::OnBeforeChange(eventObj)
{
    if (eventObj.NewValue == "")
    {
        eventObj.ReturnMessage = "You must supply a value for this field.
        eventObj.ReturnStatus = false;
        return;
    }
}
```

ReturnStatus Property (DocActionEvent Object)

Sets or retrieves a Boolean value indicating the return status of the [OnClick](#) event.

expression.ReturnStatus

expression Required. Returns a reference to the [DocActionEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the **ReturnStatus** property is set to False, the **OnClick** event fails. If set to True, the **OnClick** event is successful. The default value is True.

Note Setting the **ReturnStatus** property to false will cause a message box to be displayed which indicates what has failed.

Example

In the following example, the **ReturnStatus** property of the **DocActionEvent** object is used to indicate that the **OnClick** event was not successful:

```
function DocActionEventCancel::OnClick(eventObj)
{
    // Cancel the event.
    eventObj.ReturnStatus = false;
}
```

ReturnStatus Property (DocReturnEvent Object)

Sets or retrieves a Boolean value indicating the return status of the [OnLoad](#) and [OnSubmitRequest](#) events.

expression.ReturnStatus

expression Required. Returns a reference to the [DocReturnEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the **ReturnStatus** property is set to **False**, the **OnLoad** and **OnSubmitRequest** events fail. If set to **True**, the **OnLoad** and **OnSubmitRequest** events are successful. The default value for the **OnLoad** event is **True**, and the default value for the **OnSubmitRequest** event is **False**.

Example

In the following example, the **ReturnStatus** property of the **DocReturnEvent** object is used to indicate that the **OnLoad** event was not successful:

```
function XDocument::OnLoad(eventObj)
{
    // Cancel the event.
    eventObj.ReturnStatus = false;
}
```

ReturnStatus Property (Index)

The **ReturnStatus** property returns a **Boolean** value indicating whether the changes that occurred during the event are accepted or rejected. This property is implemented in several Microsoft Office InfoPath 2003 object model objects. Click a **ReturnStatus** property link below to view the Help topic for a specific implementation of the **ReturnStatus** property.

[ReturnStatus](#) property as it applies to the [DataDOMEvent](#) object.

[ReturnStatus](#) property as it applies to the [DocActionEvent](#) object.

[ReturnStatus](#) property as it applies to the [DocReturnEvent](#) object.

[ReturnStatus](#) property as it applies to the [MergeEvent](#) object.

[ReturnStatus](#) property as it applies to the [SaveEvent](#) object.

[ReturnStatus](#) property as it applies to the [VersionUpgradeEvent](#) object.

ReturnStatus Property (MergeEvent Object)

A read-write property that sets or retrieves a **Boolean** value indicating the return status of the [OnMergeRequest](#) event.

expression.ReturnStatus

expression Required. An expression that returns a reference to a [MergeEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The default value of the **ReturnStatus** property of the **MergeEvent** object is **False**. If this property is not set to **True**, the event handler for the **OnMergeRequest** event will fail.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

ReturnStatus Property (SaveEvent Object)

Sets or retrieves a **Boolean** value indicating the return status of the [OnSaveRequest](#) event.

expression.ReturnStatus

expression Required. An expression that returns a reference to a [SaveEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

ReturnStatus Property (SignEvent Object)

Sets or retrieves a **Boolean** value indicating the return status of the **OnSign** event.

expression.ReturnStatus

expression Required. An expression that returns a reference to a **SignEvent** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

ReturnStatus Property (VersionUpgradeEvent Object)

Sets or retrieves a Boolean value indicating the return status of the [OnVersionUpgrade](#) event.

expression.ReturnStatus

expression Required. Returns a reference to the [VersionUpgradeEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the **ReturnStatus** property is set to False, the **OnVersionUpgrade** event fails and the form is not opened. If set to True, the **OnVersionUpgrade** event is successful.

Example

In the following example, the **ReturnStatus** property of the **VersionUpgradeEvent** object is used to indicate that the **OnVersionUpgrade** event was not successful:

```
function XDocument::OnVersionUpgrade(eventObj)
{
    // Cancel the event.
    eventObj.ReturnStatus = false;
}
```

Role Property

Sets or retrieves the user's current role.

expression.**Role**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **Role** property is used to determine the current user's role.

```
var strCurrentRole = XDocument.Role;
```

RollBack Property (MergeEvent object)

A read/write property that provides additional information to the [OnMergeRequest](#) event along with the [ReturnStatus](#) property flag.

expression.RollBack

expression Required. An expression that returns a reference to a [MergeEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the **ReturnStatus** property of the **MergeEvent** object is set to **True**, this property is ignored.

If the **ReturnStatus** property is **False** and the **RollBack** property is **True**, the entire merging operation will be cancelled and rolled back to the state before the operation was initiated. If the **RollBack** property is **False**, merging the current form will fail, but the merging operation will continue with the next form.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **RollBack** property of the **MergeEvent** object is used to rollback the entire operation if a merge fails.

```
        try
    {
        XDocument.ImportDOM(eventObj.DOM)
    }
    catch (ex)
    {
        eventObj.ReturnStatus = false;
        eventObj.RollBack = true;
    }
```

ShortErrorMessage Property

Sets or retrieves the string value containing the short error message of an [Error](#) object.

expression.**ShortErrorMessage**

expression Required. An expression that returns a reference to the **Error** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The short error message is the ToolTip that users view when data validation fails in their forms.

Example

In the following example, the **ShortErrorMessage** property of the **Error** object is used to display the short message of an error in a message box:

```
var objError;
```

```
objError = XDocument.Errors(0);
```

```
XDocument.UI.Alert("Error name: " + objError.ShortErrorMessage);
```


SignatureBlockXmlNode Property

A read-only property that returns the XML node corresponding a [digital signature](#).

expression.**SignatureBlockXmlNode**

expression Required. An expression that returns a [Signature](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

SignatureContainer Property

A read-only property that returns the root XML node of the subtree containing the signatures in the [SignedDataBlock](#) object.

expression.**SignatureContainer**

expression Required. An expression that returns a reference to the **SignedDataBlock** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

SignatureRelation Property

A read-only property that returns the relation among multiple signatures of the [SignedDataBlock](#) object, expressed as [XdSignatureRelation](#) enumerated constants.

expression.**SignatureRelation**

expression Required. An expression that returns a reference to the **SignedDataBlock** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Signatures Property

A read-only property that returns a reference to the [Signatures](#) collection associated with the [SignedDataBlock](#) object.

expression.**Signatures**

expression Required. An expression that returns a reference to the **SignedDataBlock** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

SignedDataBlock Property

A read-only property that returns the signed data block that triggers the [OnSign](#) event.

expression.**SignedDataBlock**

expression Required. An expression that returns a [SignEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

SignedDataBlocks Property

A read-only property that returns a reference to the [SignedDataBlocks](#) collection that is associated with an [XDocument](#) object.

expression.**SignedDataBlocks**

expression Required. An expression that returns a reference to an **XDocument** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **SignedDataBlocks** property is used to obtain a reference to the **SignedDataBlocks** collection.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Site Property

A read-only property that returns a reference to the XML [Document Object Model \(DOM\)](#) node where the [data validation](#) event is currently being processed.

expression.**Site**

expression Required. Returns a reference to the [DataDOMEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

After you have set a reference to the XML DOM node that the **Site** property returns, you can use any of the properties and methods that are supported by the XML DOM.

Note To learn more about the XML DOM and all of the properties and methods that it supports, see the MSXML 5.0 SDK documentation in the [Microsoft Script Editor \(MSE\)](#) Help system.

Example

In the following example from the Data Validation developer sample form, the **Site** property of the **DataDOMEEvent** object is used to check the value of the XML DOM node; if it matches certain criteria, an error is created:

```
function msoxd__itemB_quantityListB::OnValidate(eventObj)
{
    if (parseInt(eventObj.Site.nodeTypeValue, 10) > 50)
        eventObj.ReportError(eventObj.Site, "Invalid quantity. " +
            "The total number of each type of block cannot exceed 50.", false);

    if (parseInt(eventObj.Site.nodeTypeValue, 10) < 0)
        eventObj.ReportError(eventObj.Site, "Invalid quantity. " +
            "The total number of each type of block cannot be less than 0.", false);
}
```


SiteUrl Property

A read-only property returning the [Uniform Resource Locator \(URL\)](#) of the Windows SharePoint Services site that the [SharepointListAdapter](#) will query.

expression.**SiteUrl**

expression Required. An expression that returns a reference to a **SharePointListAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

The following example shows how to use the **SiteUrl** property of a **SharePointListAdapter** object to store the URL location in a local variable:

```
var strSiteURL = XDocument.DataAdapters["Announcements"].SiteU
```


Solution Property

A read-only property that returns a reference to the [Solution](#) object that is associated with a Microsoft Office InfoPath 2003 form.

expression.**Solution**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Solution** object provides access to information about a form's associated [form definition \(.xsf\) file](#), including access to an XML [Document Object Model \(DOM\)](#) that contains all of the source XML of the .xsf file.

Example

In the following example, the **Solution** property of the **XDocument** object is used to load a variable with the XML contents of the .xsf file:

```
var strXSF;  
strXSF = XDocument.Solution.DOM.xml;
```


SolutionVersion Property

A read-only property that returns a string containing the version number of a Microsoft Office InfoPath 2003 [form template](#).

expression.**SolutionVersion**

expression Required. An expression that returns a reference to the [VersionUpgradeEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

This property can be used only during the [OnVersionUpgrade](#) event.

Example

In the following example, the **SolutionVersion** property of the **VersionUpgradeEvent** object is used to display the version number of an InfoPath form template in a message box:

```
function XDocument::OnVersionUpgrade(eventObj)
{
    XDocument.UI.Alert("The form version: " + eventObj.DocumentVer
        "\nThe form template version: " + eventObj.SolutionVersion);
    eventObj.ReturnStatus = true;
}
```


Source Property (DataDOMEvent Object)

A read-only property that returns a reference to the XML [Document Object Model \(DOM\)](#) where the [data validation](#) event is occurring.

expression.**Source**

expression Required. Returns a reference to the [DataDOMEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

After you have set a reference to the XML DOM node that the **Source** property returns, you can use any of the properties and methods that are supported by the XML DOM.

Note To learn more about the XML DOM and all of the properties and methods that it supports, see the MSXML 5.0 SDK documentation in the [Microsoft Script Editor \(MSE\)](#) Help system.

Example

In the following example from the Structural Editing developer sample form, the **Source** property of the **DataDOMEvent** object is used to return a reference to the XML DOM node that caused the initial change. If the node name matches certain criteria, a custom function is called.

```
function msoxd__quantity::OnAfterChange(eventObj)
{
    if (!eventObj.IsUndoRedo && eventObj.Source.nodeName != "item
        Calculate(eventObj.Site.parentNode);
}
```


Source Property (DocActionEvent Object)

A read-only property that returns a reference to the inner-most XML [Document Object Model \(DOM\)](#) node of a form's underlying XML document.

expression.**Source**

expression Required. Returns a reference to the [DocActionEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

After you have set a reference to the XML DOM node that the **Source** property returns, you can use any of the properties and methods that are supported by the XML DOM.

Note To learn more about the XML DOM and all of the properties and methods that it supports, see the MSXML 5.0 SDK documentation in the [Microsoft Script Editor \(MSE\)](#) Help system.

Example

In the following example, the **Source** property of the **DocActionEvent** object is used to display the source XML data of the XML DOM node in a message box:

```
function DocActionEventSource::OnClick(eventObj)
{
    XDocument.UI.Alert("Source: " + eventObj.Source.xml);
}
```


Source Property (Index)

The **Source** property returns a reference to the XML [Document Object Model \(DOM\)](#) node containing the source XML of the data being changed during an event. This property is implemented in a number of Microsoft Office InfoPath 2003 object model objects. Click a **Source** property link below to view the Help topic for a specific implementation of the **Source** property.

[Source](#) property as it applies to the [DataDOMEvent](#) object.

[Source](#) property as it applies to the [DocActionEvent](#) object.

Status Property (Certificate Object)

A read-only property that returns the status of the [digital certificate](#). The status that is returned is one of the [XdcertificateStatus](#) enumerated constants.

expression.**Status**

expression Required. An expression that returns a [Certificate](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Status Property (Index)

This property is implemented in several Microsoft Office InfoPath 2003 object model collections. Click a **Status** property link below to view the Help topic for a specific implementation of the **Status** property.

[Status](#) property as it applies to the [Certificate](#) object.

[Status](#) property as it applies to the [Signature](#) object.

Status Property (Signature Object)

A read-only property that returns the status of the specified [digital signature](#). The status that is returned is based on the [XdSignatureStatus](#) enumeration

expression.**Status**

expression Required. An expression that returns a [Signature](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Status** property only verifies whether the hash of the digital signature is valid. It does not verify the chain of trust of the digital certificate, nor does it verify that the image of the view captured at the time the signature was added matches the current view of the signed form.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Subject Property (EmailAdapter Object)

Returns or sets a string that represents the subject of the e-mail message associated with the specified [EmailAdapter](#) object.

expression.**Subject**

expression Required. An expression that returns a reference to an **EmailAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

This example shows how to use the **Subject** property of an instance of the **EmailAdapter** object to change the subject of the e-mail message before the **EmailAdapter** object is submitted:

```
objEmailAdapter.Subject = "Weekly Status Report";
```

Subject Property (MailEnvelope Object)

Sets or retrieves a string containing the subject value used in the [MailEnvelope](#) object that is associated with a [Window](#) object.

expression.**Subject**

expression Required. An expression that returns a reference to the **MailEnvelope** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **MailEnvelope** object does not support the programmatic creation of the body of an e-mail message. Users enter the body text after the e-mail message is displayed in the default e-mail editor.

Example

In the following example, the **Subject** property of the **MailEnvelope** object is used to set the subject value of a custom e-mail message:

```
function CreateMailEnvelope::OnClick(eventObj)
{
    var objEmail;

    objEmail = Application.ActiveWindow.MailEnvelope;
    objEmail.To = "someone@example.com";
    objEmail.CC = "someone@example.com";
    objEmail.BCC = "someone@example.com";
    objEmail.Subject = "Test e-mail message";
    objEmail.Visible = true;
    objEmail = null;
}
```


SubmitAllowed Property (ADOAdapter Object)

A read-only property that returns a **Boolean** value corresponding to the **submitAllowed** attribute in the [form definition file \(.xsf\)](#).

expression.**SubmitAllowed**

expression Required. An expression that returns a reference to an [ADOAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The default value of the **SubmitAllowed** property is **False**, as is the default value for the **submitAllowed** attribute. If the **submitAllowed** attribute is set to **True**, the ADO connection supports submitting data as well as querying.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

SubmitAllowed Property (DAVAdapter Object)

A read-only property that returns a **Boolean** value that always returns **False**, corresponding to the **submitAllowed** attribute in the [form definition file \(.xsf\)](#).

expression.**SubmitAllowed**

expression Required. An expression that returns a reference to a [DAVAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

SubmitAllowed Property (EmailAdapter Object)

A read-only property that returns a **Boolean** value that always returns **True**, corresponding to the **submitAllowed** attribute in the [form definition file \(.xsf\)](#).

expression.**SubmitAllowed**

expression Required. An expression that returns a reference to an [EmailAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

SubmitAllowed Property (HWSAdapter Object)

A read-only property that returns a **Boolean** value that is always **True**, corresponding to the **submitAllowed** attribute in the [form definition file \(.xsf\)](#).

expression.**SubmitAllowed**

expression Required. An expression that returns a reference to a [HWSAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

SubmitAllowed Property (Index)

The **SubmitAllowed** property returns a value that corresponds to the **submitAllowed** attribute in the form definition file (.xsf). This property is implemented in several Microsoft Office InfoPath 2003 object model objects. Click a **SubmitAllowed** property link below to view the Help topic for a specific implementation of the **SubmitAllowed** property.

[SubmitAllowed](#) property as it applies to the [ADOAdapter](#) object.

[SubmitAllowed](#) property as it applies to the [DAVAdapter](#) object.

[SubmitAllowed](#) property as it applies to the [EmailAdapter](#) object.

[SubmitAllowed](#) property as it applies to the [HWSAdapter](#) object.

[SubmitAllowed](#) property as it applies to the [SharepointListAdapter](#) object.

[SubmitAllowed](#) property as it applies to the [WebServiceAdapter](#) object.

[SubmitAllowed](#) property as it applies to the [XMLFileAdapter](#) object.

SubmitAllowed Property (SharePointListAdapter Object)

A read-only property that returns a **Boolean** value corresponding to the **submitAllowed** attribute in the [form definition file \(.xsf\)](#). Always returns **False** for the **SharePointListAdapter** object.

expression.**SubmitAllowed**

expression Required. An expression that returns a reference to a [SharepointListAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

SubmitAllowed Property (WebServiceAdapter Object)

A read-only property that returns a **Boolean** value corresponding to the **submitAllowed** attribute in the [form definition file \(.xsf\)](#). The default value is **False**.

expression.**SubmitAllowed**

expression Required. An expression that returns a reference to a [WebServiceAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

SubmitAllowed Property (XMLFileAdapter Object)

A read-only property that returns a **Boolean** value that always returns **False**, corresponding to the **submitAllowed** attribute in the [form definition file \(.xsf\)](#).

expression.**SubmitAllowed**

expression Required. An expression that returns a reference to an [XMLFileAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

TaskPanels Property

A read-only property that returns a reference to the [TaskPanels](#) collection that is associated with the [Window](#) object.

expression.**TaskPanels**

expression Required. An expression that returns a reference to the **Window** object.

Security Level

0: Can be accessed without restrictions.

Remarks

The **TaskPanes** property can be used only with the editing window types; if used with a designing window type, it will return an error.

Example

In the following example, the **TaskPanes** property of the **Window** object is used to set a reference to the **TaskPanes** collection:

```
var objTaskPanes;
```

```
objTaskPanes = Application.ActiveWindow.TaskPanes;
```


Timeout Property (ADOAdapter Object)

Sets or retrieves the **long integer** timeout value for an [ADOAdapter](#) object.

expression.**Timeout**

expression Required. An expression that returns a reference to the **ADOAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Timeout** property of the **ADOAdapter** object contains the timeout value that is used by the ADO [data adapter](#) to regulate the time used to submit and retrieve data from an ActiveX Data Objects/OLEDB external data source.

Note The **ADOAdapter** object is limited to work only with Microsoft SQL Server and Microsoft Access databases.

Example

In the following example, the **Timeout** property of the **ADOAdapter** object is used to display the timeout value of the ADO data adapter in a message box:

```
var objADOAdapter;
```

```
objADOAdapter = XDocument.DataObjects("CityList").QueryAdapte;  
XDocument.UI.Alert("SQL command text: " + objADOAdapter.Timeo
```


Timeout Property (Index)

The **Timeout** property sets or retrieves the time-out value for some types of [data adapters](#). This property is implemented in several Microsoft Office InfoPath 2003 object model objects. Click a **Timeout** property link below to view the Help topic for a specific implementation of the **Timeout** property.

[Timeout](#) property as it applies to the [ADOAdapter](#) object.

[Timeout](#) property as it applies to the [WebServiceAdapter](#) object.

Timeout Property (WebServiceAdapter Object)

Sets or retrieves the **long integer** time-out value in seconds for a [WebServiceAdapter](#) object.

expression.Timeout

expression Required. An expression that returns a reference to a **WebServiceAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The default value of the **Timeout** property of the **WebServiceAdapter** object is 30 seconds.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

To Property (EmailAdapter Object)

Returns or sets a string that represents the recipients for the e-mail message associated with a specified [EmailAdapter](#) object.

expression.To

expression Required. An expression that returns a reference to an **EmailAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The value must be a semicolon-delimited string that can be resolved into a list of valid e-mail addresses by the user's e-mail client.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

This example shows how to use the **To** property of an instance of the **EmailAdapter** object to change the recipients of an e-mail message before the **EmailAdapter** object is submitted:

```
objEmailAdapter.To = objEmailAdapter.To + "; newUser@example.cc
```

To Property (Index)

The **To** property returns or sets the recipients of an e-mail message. This property is implemented in several Microsoft Office InfoPath 2003 object model objects. Click a **To** property link below to view the Help topic for a specific implementation of the **To** property.

[To](#) property as it applies to the [EmailAdapter](#) object.

[To](#) property as it applies to the [MailEnvelope](#) object.

To Property (MailEnvelope Object)

Sets or retrieves a string containing the send-to value used in the [MailEnvelope](#) object that is associated with a [Window](#) object.

expression.To

expression Required. An expression that returns a reference to the **MailEnvelope** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The values set for the **To** property should be a string of valid e-mail addresses. You can specify multiple e-mail addresses by using ";" between each of them, as shown in the following example:

```
objEmail.To = "someone@example.com;someone@example.com"
```

Example

In the following example, the **To** property of the **MailEnvelope** object is used to set the send-to value of a custom e-mail message:

```
function CreateMailEnvelope::OnClick(eventObj)
{
    var objEmail;

    objEmail = Application.ActiveWindow.MailEnvelope;
    objEmail.To = "someone@example.com";
    objEmail.CC = "someone@example.com";
    objEmail.BCC = "someone@example.com";
    objEmail.Subject = "Test e-mail message";
    objEmail.Visible = true;
    objEmail = null;
}
```

Top Property

A read/write property of type **long integer** that specifies the vertical position of the window represented by the [Window](#) object, measured in points.

expression.**Top**

expression Required. An expression that returns a reference to a **Window** object.

Remarks

The **Top** property will return an error if it is set on a window that is minimized or maximized.

Setting the **Top** property to a position that is off the screen, will cause the entire window to be displayed on the screen.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Type Property (DocContextChangeEvent Object)

A read-only property that returns a string containing the type of context change event that occurred when the [OnContextChange](#) event was triggered.

expression.Type

expression Required. An expression that returns a reference to the [DocContextChangeEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

As described in the [OnContextChange](#) event topic, the **Type** property returns only the value "ContextNode" for context changes in Microsoft InfoPath 2003 Service Pack 1. Nevertheless, if code in an event handler performs actions that depend on current functionality, that code should still be designed to check the value of the **Type** property, because future versions of InfoPath may use different values for different context changes.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Type Property (Error Object)

A read-only property that returns a string value containing the type of an **Error** object.

expression.**Type**

expression Required. An expression that returns a reference to the **Error** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

There are three types of errors:

SCHEMA_VALIDATION Data validation failed as a result of an [XML Schema](#)-defined constraint.

SYSTEM_GENERATED Data validation failed as a result of constraints defined in the [form definition \(.xsf\) file](#) or as a result of scripting code calling the [ReportError](#) method of the [DataDOMEvent](#) object.

USER_SPECIFIED Data validation failed as a result of a custom scripting error using the [Add](#) method of the **Errors** collection.

Example

In the following example, the **Type** property of the **Error** object is used to display the type of an error in a message box:

```
var objError;
```

```
objError = XDocument.Errors(0);
```

```
XDocument.UI.Alert("Error name: " + objError.Type);
```


Type Property (HTMLTaskPane Object)

A read-only property that returns a value indicating the type of [task pane](#) represented by the [HTMLTaskPane](#) object, which is always a [custom task pane](#).

expression.Type

expression Required. An expression that returns a reference to the [HTMLTaskPane](#) object. Based on the [XdTaskPaneType](#) enumeration.

Security Level

0: Can be accessed without restrictions.

Remarks

The **Type** property of the **HTMLTaskPane** object is one of the properties inherited from the [TaskPane](#) object when the type of the task pane is 0, which means that it is the custom task pane.

Note The **Type** property is based on the **XdTaskPaneType** enumeration. These enumerated values are also used as arguments to the [Item](#) property of the [TaskPanes](#) collection for returning a reference to a specified type of task pane.

Example

In the following example, the **Item** property of the **TaskPanes** collection is used to set a reference to the **HTMLTaskPane** object that represents the custom task pane. Then the [Visible](#) property of the **HTMLTaskPane** object is used to make the custom task pane appear in the Microsoft Office InfoPath 2003 user interface. The type value of the custom task pane is the same as the value passed to the **Item** property.

```
var objTaskPane;
```

```
objTaskPane = XDocument.View.Window.TaskPanes(0);
```

```
objTaskPane.Visible = true;
```

Type Property (Index)

The **Type** property returns a value indicating the type of the object. This property is implemented in several Microsoft Office InfoPath 2003 object model objects. Click a **Type** property link below to view the Help topic for a specific implementation of the **Type** property.

[Type](#) property as it applies to the [DocContextChangeEvent](#) object.

[Type](#) property as it applies to the [Error](#) object.

[Type](#) property as it applies to the [HTMLTaskPane](#) object.

[Type](#) property as it applies to the [TaskPane](#) object.

[Type](#) property as it applies to the [Window](#) object.

Type Property (TaskPane Object)

A read-only property that returns a value indicating the type of [task pane](#) represented by the [TaskPane](#) object.

expression.Type

expression Required. An expression that returns a reference to the [TaskPane](#) object. Based on the [XdTaskPaneType](#) enumeration.

Security Level

0: Can be accessed without restrictions.

Remarks

If the **Type** property of the **TaskPane** object returns 0, the task pane is a custom task pane. If the **Type** property returns any other value, the task pane is a built-in task pane.

Note The **Type** property is based on the **XdTaskPaneType** enumeration. These enumerated values are also used as arguments to the **Item** property of the **TaskPanes** collection for returning a reference to a specified type of task pane.

Example

In the following example, the **Item** property of the **TaskPanes** collection is used to set a reference to the **TaskPane** object that represents the built-in Help task pane. Then the [Visible](#) property of the **TaskPane** object is used to make the Help task pane appear in the Microsoft Office InfoPath 2003 user interface.

```
var objTaskPane;
```

```
objTaskPane = XDocument.View.Window.TaskPanes.Item(4);  
objTaskPane.Visible = true;
```


Type Property (Window Object)

A read-only property that returns a long integer value that indicates the type of window that is represented by the [Window](#) object. The value returned is based on the [XdWindowType](#) enumeration.

expression.**Type**

expression Required. An expression that returns a reference to the **Window** object.

Security Level

0: Can be accessed without restrictions.

Remarks

The **Window** objects represent the two types of windows that are used in the InfoPath application: the editing window that is used as the [form area](#) when a user fills out a form, and the designing window that is used as the [design mode](#) when a user designs a form.

Example

In the following example, the **Type** property of the **Window** object is used to determine the type of window that is the currently active window:

```
if (Application.ActiveWindow.Type == 0)
    XDocument.UI.Alert("The active window is an editing window.");
else
    XDocument.UI.Alert("The active window is a designing window.");
```

UI Property

Returns a reference to the Microsoft Office InfoPath 2003 [UI](#) object.

expression.**UI**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **UI** (user interface) object provides a number of methods that can be used to display custom and built-in dialog boxes.

Example

In the following example, the **UI** property of the **XDocument** object is used to display a simple message box using the [Alert](#) method:

```
XDocument.UI.Alert("Here is the message text.");
```


URI Property (Index)

The **URI** property returns a string value that specifies a [Uniform Resource Identifier \(URI\)](#) location. This property is implemented in a number of Microsoft Office InfoPath 2003 object model objects. Click a **URI** property link below to view the Help topic for a specific implementation of the **URI** property.

[URI](#) property as it applies to the [Solution](#) object.

[URI](#) property as it applies to the [XDocument](#) object.

URI Property (Solution Object)

A read-only property that returns a string value containing the [Uniform Resource Identifier \(URI\)](#) of a Microsoft Office InfoPath 2003 [form template](#).

expression.**URI**

expression Required. An expression that returns a reference to a [Solution](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The value of the **URI** property of the **Solution** object can take the form of a [Uniform Resource Locator \(URL\)](#) or [Uniform Resource Name \(URN\)](#), depending on the location from which the form was opened.

Example

In the following example, the **URI** property of the **Solution** object is used to display a form template's URI in a message box:

```
XDocument.UI.Alert("URI: " + XDocument.Solution.URI);
```


URI Property (XDocument Object)

A read-only property that returns a string value containing the [Uniform Resource Identifier \(URI\)](#) of a Microsoft Office InfoPath 2003 form.

expression.**URI**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **URI** property can be used as the name of a form when accessed through the [XDocuments](#) collection, as shown in the following example:

```
Application.XDocuments(XDocument.URI);
```

Example

In the following example, the **URI** property of the **XDocument** object is used to display a form's URI in a message box:

```
XDocument.UI.Alert("URI: " + XDocument.URI);
```

UsableHeight Property

A read-only property of type **long integer** that returns the available screen height (the maximum number of points to which you can set the height of an InfoPath document window.)

expression.**UsableHeight**

expression Required. An expression that returns a reference to an [Application](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

UsableWidth Property

A read-only property of type **long integer** that returns the available screen width (the maximum number of points to which you can set the width of an InfoPath document window.)

expression.UsableWidth

expression Required. An expression that returns a reference to an [Application](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

User Property

A read-only property that returns a reference to the [User](#) object.

expression.**User**

expression Required. An expression that returns a reference to an [Application](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Util Property

Read-only property that returns a reference to the [Util](#) object.

expression.**Util**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Version Property (Application Object)

A read-only property that returns a string containing the Microsoft Office InfoPath 2003 version number.

expression.**Version**

expression Required. An expression that returns a reference to the [Application](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

The version number of the application does not contain the name. To obtain the name of an application, use the [Name](#) property of the **Application** object.

To obtain the version number of a [form template](#), use the [Version](#) property of the [Solution](#) object.

Example

In the following example, the **Version** property of the **Application** object is used to display the application's version number in a message box:

```
XDocument.UI.Alert("Application version: " + Application.Version);
```

Version Property (Index)

The **Version** property returns a string value that specifies a version number. This property is implemented in a number of Microsoft Office InfoPath 2003 object model objects. Click a **Version** property link below to view the Help topic for a specific implementation of the **Version** property.

[Version](#) property as it applies to the [Application](#) object.

[Version](#) property as it applies to the [Solution](#) object.

Version Property (Solution Object)

A read-only property that returns a string containing the version number of a Microsoft Office InfoPath 2003 [form template](#).

expression.**Version**

expression Required. An expression that returns a reference to the [Solution](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

To obtain the version number of the InfoPath application, use the [Version](#) property of the [Application](#) object.

Example

In the following example, the **Version** property of the **Solution** object is used to display a form template's version number in a message box:

```
XDocument.UI.Alert("Form template version: " + XDocument.Solutio
```

View Property

A read-only property that returns a reference to the [View](#) object associated with a Microsoft Office InfoPath 2003 form.

expression.**View**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **View** object that the **View** property accesses represents the view that is currently active in an InfoPath form. After you have set a reference to the **View** object, you can access any of its properties and methods to programmatically interact with the view.

Example

In the following example, the **View** property of the **XDocument** object is used to set a reference to the **View** object; then, using the [Name](#) property of the **View** object, it displays the name of the view in a message box:

```
var objView;
```

```
objView = XDocument.View;
```

```
XDocument.UI.Alert("View name: " + objView.Name);
```

ViewInfos Property

A read-only property that returns a reference to the [ViewInfos](#) collection associated with a Microsoft Office InfoPath 2003 form.

expression.**ViewInfos**

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **ViewInfos** collection contains a collection of [ViewInfo](#) objects that contain information about each of the views implemented in an InfoPath form.

Example

In the following example, the **ViewInfos** property of the **XDocument** object is used to set a reference to the **ViewInfos** collection; then, using the **Count** property of the **ViewInfos** collection, it loops through the collection to determine the default view using the **IsDefault** property of the **ViewInfo** object. When the default view is found, the code displays the name of the view in a message box.

```
var objViewInfos;
```

```
objViewInfos = XDocument.ViewInfos;  
for (i=0; i < objViewInfos.Count; i++)  
{  
    if (objViewInfos(i).IsDefault)  
        XDocument.UI.Alert("The default view is: " + objViewInfos(i).Nam  
}
```


Visible Property (HTMLTaskPane Object)

Sets or retrieves a Boolean value indicating that the [task pane](#) represented by the [HTMLTaskPane](#) object, which is always a [custom task pane](#), is visible in the Microsoft Office InfoPath 2003 user interface.

expression.**Visible**

expression Required. An expression that returns a reference to the **HTMLTaskPane** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Visible** property of the **HTMLTaskPane** object is one of the properties inherited from the [TaskPane](#) object when the type of the task pane is 0, which means that it is the custom task pane.

Setting the **Visible** property to True causes the task pane to appear in the user interface, and setting it to False causes it to be removed from the user interface.

Example

In the following example, the [Item](#) property of the [TaskPanels](#) collection is used to set a reference to the **HTMLTaskPane** object that represents the custom task pane. Then the **Visible** property of the **HTMLTaskPane** object is used to make the custom task pane appear in the InfoPath user interface.

```
var objTaskPane;
```

```
objTaskPane = XDocument.View.Window.TaskPanels(0);  
objTaskPane.Visible = true;
```

Visible Property (Index)

The **Visible** property specifies or returns a **Boolean** value indicating whether the user interface component represented by the object is visible in the user interface. This property is implemented in several Microsoft Office InfoPath 2003 object model objects. Click a **Visible** property link below to view the Help topic for a specific implementation of the **Visible** property.

[Visible](#) property as it applies to the [HTMLTaskPane](#) object.

[Visible](#) property as it applies to the [MailEnvelope](#) object.

[Visible](#) property as it applies to the [TaskPane](#) object.

Visible Property (MailEnvelope Object)

Sets or retrieves a Boolean value that indicates the visibility of the custom e-mail message created with the [MailEnvelope](#) object that is associated with a [Window](#) object.

expression.**Visible**

expression Required. An expression that returns a reference to the **MailEnvelope** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

When the **Visible** property of the **MailEnvelope** object is set to True, the custom e-mail message will be displayed using the default e-mail editor. If there is no default e-mail editor configured, the **Visible** property will return an error.

Example

In the following example, the **Visible** property of the **MailEnvelope** object is used to display a custom e-mail message in the default e-mail editor:

```
function CreateMailEnvelope::OnClick(eventObj)
{
    var objEmail;

    objEmail = Application.ActiveWindow.MailEnvelope;
    objEmail.To = "someone@example.com";
    objEmail.CC = "someone@example.com";
    objEmail.BCC = "someone@example.com";
    objEmail.Subject = "Test e-mail message";
    objEmail.Visible = true;
    objEmail = null;
}
```


Visible Property (TaskPane Object)

Sets or retrieves a Boolean value indicating that the [task pane](#) represented by the [TaskPane](#) object is visible in the Microsoft Office InfoPath 2003 user interface.

expression.**Visible**

expression Required. An expression that returns a reference to the **TaskPane** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Setting the **Visible** property to True causes the task pane to appear in the user interface, and setting it to False causes it to be removed from the user interface.

Example

In the following example, the [Item](#) property of the **TaskPanes** collection is used to set a reference to the **TaskPane** object that represents the built-in Help task pane. Then the **Visible** property of the [TaskPanes](#) object is used to make the Help task pane appear in the InfoPath user interface.

```
var objTaskPane;
```

```
objTaskPane = XDocument.View.Window.TaskPanes(4);  
objTaskPane.Visible = true;
```

Width Property

A read/write property of type **long integer** that specifies the width of the window represented by the [Window](#) object, measured in points.

expression.**Width**

expression Required. An expression that returns a reference to a **Window** object.

Remarks

The **Width** property will return an error if it is set on a window that is minimized or maximized.

The **Width** property can't be set to a value that is larger than the value returned by the [UsableWidth](#) property.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Window Property (HTMLTaskPaneExternal Object)

A read-only property that returns a reference to the [Window](#) object associated with a [custom task pane](#).

expression.**Window**

expression Required. An expression that returns a reference to the [HTMLTaskPaneExternal](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

The **Window** object returned by the **Window** property represents the currently active Microsoft Office InfoPath 2003 window that is associated with the custom task pane.

Example

In the following example, the **HTMLTaskPaneExternal** object is used through the **external** property of the [Dynamic HTML \(DHTML\) window](#) object to set a reference to the **Window** object that is part of the InfoPath object model:

```
var objWindow;
```

```
objWindow = window.external.Window;  
objWindow.MailEnvelope.Visible = true;
```

Window Property (Index)

The **Window** property returns a reference to a [Window](#) object. This property is implemented in a number of Microsoft Office InfoPath 2003 object model objects. Click a **Window** property link below to view the Help topic for a specific implementation of the **Window** property.

[Window](#) property as it applies to the [HTMLTaskPaneExternal](#) object.

[Window](#) property as it applies to the [View](#) object.

Window Property (View Object)

A read-only property that returns a reference to the window object associated with a view.

expression.**Window**

expression Required. An expression that returns a reference to the [View](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The [Window](#) object returned by the **Window** property represents the currently active Microsoft Office InfoPath 2003 window. It can also be accessed through the [Windows](#) collection.

Example

In the following example, the **Window** property of the **View** object is used to set a reference to the first [task pane](#) contained in the [TaskPanes](#) collection:

```
var objTaskPane;
```

```
objTaskPane = XDocument.View.Window.TaskPanes(0);
```

Windows Property

A read-only property that returns a reference to the [Windows](#) collection.

expression.**Windows**

expression Required. An expression that returns a reference to the [Application](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

After you have set a reference to the **Windows** collection, you can use its properties to access each of the [Window](#) objects that it contains.

Note The **Windows** collection can be used only to get the count of the **Window** objects that it contains or to return a reference to a **Window** object. It cannot be used to add or remove **Window** objects.

Example

In the following example, the **Windows** property is used to access the **Count** property of the **Windows** collection and display the value in a message box:

```
XDocument.UI.Alert("Count of Windows: " + Application.Windows.C
```

WindowState Property

A read/write property of type [XdWindowState](#) that returns or sets the state of the window represented by the [Window](#) object.

expression.**WindowState**

expression Required. An expression that returns a reference to a **Window** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

WSDLURL Property

A read-only property that returns a string value containing the [Uniform Resource Locator \(URL\)](#) of the Web Services Description Language (WSDL) file for the Web service associated with the [WebServiceAdapter](#) object.

expression.**WSDLURL**

expression Required. An expression that returns a reference to the **WebServiceAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The WSDL file is an XML document that defines the format of messages an XML Web service understands. The service description serves as an agreement that defines the behavior of an XML Web service and instructs potential clients in how to interact with it. The behavior of an XML Web service is determined by messaging patterns that the service defines and supports. These patterns conceptually dictate what the service consumer can expect to happen when a properly formatted message is submitted to the XML Web service.

Example

In the following example, the **WSDLURL** property of the **WebServiceAdapter** object is used to display the URL of the WSDL file that is used for the Web service:

```
var objWSAdapter;
```

```
objWSAdapter = XDocument.DataObjects("WebCityList").QueryAdap  
XDocument.UI.Alert("WSDL file URL: " + objWSAdapter.WSDLUR
```


XDocument Property (DataDOMEvent Object)

A read-only property that returns a reference to the [XDocument](#) object that is associated with the [DataDOMEvent](#) object during a [data validation](#) event.

expression.**XDocument**

expression Required. Returns a reference to the **DataDOMEvent** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

After you have set a reference to the **XDocument** object that the **XDocument** property of the **DataDOMEEvent** object returns, you can use any of the properties and methods that it provides.

Example

In the following example, the **XDocument** property of the **DataDOMEvent** object is used to set a reference to the **XDocument** object:

```
var objXDoc;
```

```
objXDoc = eventObj.XDocument;
```

XDocument Property (DocActionEvent Object)

A read-only property that returns a reference to the [XDocument](#) object that is associated with the [DocActionEvent](#) object during an [OnClick](#) event.

expression.XDocument

expression Required. Returns a reference to the **DocActionEvent** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

After you have set a reference to the **XDocument** object that the **XDocument** property of the **DocActionEvent** object returns, you can use any of the properties and methods that it provides.

Example

In the following example, the **XDocument** property of the **DocActionEvent** object is used to set a reference to the **XDocument** object:

```
var objXDoc;
```

```
objXDoc = eventObj.XDocument;
```

XDocument Property (DocContextChangeEvent Object)

A read-only property that returns a reference to the [XDocument](#) object that is associated with the [DocContextChangeEvent](#) object in an [OnContextChange](#) event.

expression.XDocument

expression Required. Returns a reference to a **DocContextChangeEvent** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

XDocument Property (DocEvent Object)

A read-only property that returns a reference to the [XDocument](#) object that is associated with the [DocEvent](#) object during an [OnSwitchView](#) or [OnAfterChange](#) event.

expression.XDocument

expression Required. Returns a reference to the **DocEvent** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

After you have set a reference to the **XDocument** object that the **XDocument** property of the **DocEvent** object returns, you can use any of the properties and methods that it provides.

Example

In the following example, the **XDocument** property of the **DocEvent** object is used to set a reference to the **XDocument** object:

```
var objXDoc;
```

```
objXDoc = eventObj.XDocument;
```

XDocument Property (DocReturnEvent Object)

A read-only property that returns a reference to the [XDocument](#) object that is associated with the [DocReturnEvent](#) object during an [OnLoad](#) or [OnSubmitRequest](#) event.

expression.XDocument

expression Required. Returns a reference to the **DocReturnEvent** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

After you have set a reference to the **XDocument** object that the **XDocument** property of the **DocReturnEvent** object returns, you can use any of the properties and methods that it provides.

Example

In the following example, the **XDocument** property of the **DocReturnEvent** object is used to set a reference to the **XDocument** object:

```
var objXDoc;
```

```
objXDoc = eventObj.XDocument;
```


XDocument Property (HTMLTaskPaneExternal Object)

A read-only property that returns a reference to the [XDocument](#) object associated with a [custom task pane](#).

expression.XDocument

expression Required. An expression that returns a reference to the [HTMLTaskPaneExternal](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

The **XDocument** object returned by the **XDocument** property represents the form's underlying XML document that is associated with the custom task pane.

Example

In the following example, the **HTMLTaskPaneExternal** object is used through the **external** property of the [Dynamic HTML \(DHTML\) window](#) object to set a reference to the **XDocument** object that is part of the InfoPath object model:

```
var objXDoc;
```

```
objXDoc = window.external.XDocument;  
objXDoc.View.SwitchView("View2");
```

XDocument Property (Index)

The **XDocument** property returns a reference to an [XDocument](#) object. This property is implemented in several Microsoft Office InfoPath 2003 object model objects. Click an **XDocument** property link below to view the Help topic for a specific implementation of the **XDocument** property.

[XDocument](#) property as it applies to the [DataDOMEvent](#) object.

[XDocument](#) property as it applies to the [DocActionEvent](#) object.

[XDocument](#) property as it applies to the [DocContextChangeEvent](#) object.

[XDocument](#) property as it applies to the [DocEvent](#) object.

[XDocument](#) property as it applies to the [DocReturnEvent](#) object.

[XDocument](#) property as it applies to the [HTMLTaskPaneExternal](#) object.

[XDocument](#) property as it applies to the [MergeEvent](#) object.

[XDocument](#) property as it applies to the [SaveEvent](#) object.

[XDocument](#) property as it applies to the [VersionUpgradeEvent](#) object.

[XDocument](#) property as it applies to the [Window](#) object.

XDocument Property (InfoPathControlSite Object)

Retrieves a reference to the [XDocument](#) object associated with the view that contains the control.

expression.XDocument

expression Required. An expression that returns a reference to the [InfoPathControl](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **InfoPathControl** and [InfoPathControlSite](#) objects and their methods and properties are designed to be used only from the implementation of an ActiveX control. These objects and their members are not supported in InfoPath form code. For more information on how to create ActiveX controls that work with InfoPath, see the [InfoPath Developer Center](#).

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form template that contains a view with an ActiveX control that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

XDocument Property (MergeEvent Object)

A read-only property that returns a reference to the [XDocument](#) object that is associated with the [MergeEvent](#) object in an [OnMergeRequest](#) event.

expression.**XDocument**

expression Required. Returns a reference to a **MergeEvent** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

XDocument Property (SignEvent Object)

Retrieves a reference to the [XDocument](#) object associated with the [OnSign](#) event.

expression.XDocument

expression Required. An expression that returns a reference to a [SignEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

XDocument Property (VersionUpgradeEvent Object)

A read-only property that returns a reference to the [XDocument](#) object that is associated with the [VersionUpgradeEvent](#) object during an [OnVersionUpgrade](#) event.

expression.XDocument

expression Required. Returns a reference to the **VersionUpgradeEvent** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

After you have set a reference to the **XDocument** object that the **XDocument** property of the **VersionUpgradeEvent** object returns, you can use any of the properties and methods that it provides.

Example

In the following example, the **XDocument** property of the **VersionUpgradeEvent** object is used to set a reference to the **XDocument** object:

```
var objXDoc;
```

```
objXDoc = eventObj.XDocument;
```


XDocument Property (Window Object)

A read-only property that returns a reference to the [XDocument](#) object that is associated with the window that is represented by the [Window](#) object.

expression.**XDocument**

expression Required. An expression that returns a reference to the **Window** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **XDocument** property can be used only with the editing window types; if used with a designing window type, it will return an error. It will also return an error if no form is open in the [form area](#).

Example

In the following example, the **XDocument** property of the **Window** object is used to set a reference to the form's underlying XML document that is associated with the currently active window:

```
var objXDoc;
```

```
objXDoc = Application.ActiveWindow.XDocument;
```

XDocuments Property

A read-only property that returns a reference to the [XDocuments](#) collection.

expression.XDocuments

expression Required. An expression that returns a reference to the [Application](#) object.

Security Level

0: Can be accessed without restrictions.

Remarks

After you have set a reference to the **XDocuments** collection, you can use its properties to access each of the [XDocument](#) objects that it contains.

Example

In the following example, the **XDocuments** property is used to access the **Count** property of the **XDocuments** collection and display the value in a message box:

```
XDocument.UI.Alert("Count of XDocuments: " + Application.XDocu
```


XPath Property

A read-only property that returns the [XPath](#) expression of a **SignedDataBlock** object.

expression.XPath

expression Required. An expression that returns a reference to a **SignedDataBlock** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

XPathNamespaceDeclarations Property

A read-only property containing the namespace declarations for the [XPath](#) expression returned by the [XPath](#) property of a [SignedDataBlock](#) object.

expression.XPathNamespaceDeclarations

expression Required. An expression that returns a reference to a [SignedDataBlock](#) object

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Activate Method

Activates the window that is represented by the [Window](#) object.

```
expression.Activate()
```

expression Required. An expression that returns a reference to the **Window** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Activate** method can be used only with the editing window types; if used with a designing window type, it will return an error.

To determine whether a window is the active window, use the [Active](#) property of the **Window** object.

Example

In the following example, the **Activate** method of the **Window** object is used to activate a window that is associated with the view. Note the check of the window type.

```
var objWindow;  
  
// Set a reference to a view's associated window.  
objWindow = XDocument.View.Window;  
  
if (objWindow.Type == 0)  
{  
    // Make the window the active window.  
    objWindow.Activate();  
}  
  
objWindow = null;
```


Add Method

Adds an [Error](#) object to the [Errors](#) collection and returns a reference to the new **Error** object.

```
expression.Add(ByVal varNode As Variant, ByVal bstrConditionName As String, ByVal bstrShortErrorMessage As String, [ByVal bstrDetailedErrorMessage As String], [ByVal IErrorCode As Long], [ByVal bstrType As String = "modeless"]) As Error
```

expression Required. An expression that returns a reference to the **Errors** collection.

varNode Required **Variant**. The XML node that the error will be associated with.

bstrConditionName Required **String**. The name of the error.

bstrShortErrorMessage Required **String**. The short message for the error.

bstrDetailedErrorMessage Optional **String**. The detailed message for the error.

IErrorCode Optional **Long Integer**. Default value is 0. The error code of the error.

bstrType Optional **String**. Default value is "modeless". The type of error processing. The other supported value is "modal".

returns A reference to the newly created **Error** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Add** method is used to create custom error messages in a Microsoft Office InfoPath 2003 form. There are two types of errors that can be created using the **Add** method:

modeless The user is notified of the error with an [inline alert](#) and can choose to return to the previous value with an undo operation.

modal The user is notified of the error with a [dialog box alert](#). After clicking **OK** in the dialog box alert, the error will appear as an inline alert and the user can choose to return to the previous value with an undo operation.

Note Custom errors can also be created using the [ReportError](#) method of the [DataDOMEvent](#) object.

Example

In the following example, the **Add** method of the **Errors** collection is used to create a custom error message:

```
var objErrors;  
var objError;
```

```
objErrors = XDocument.Errors;  
objError = objErrors.Add(MyXMLNode, "ValidationError", "The data
```

Alert Method

Displays a message box with a custom text message in a Microsoft Office InfoPath 2003 form.

expression.Alert(ByVal ***bstrAlertString*** As String)

expression Required. An expression that returns a reference to the [UI](#) object.

bstrAlertString Required **String**. The text to be displayed.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Alert** method displays a simple message box that uses an information icon and an **OK** button. Only the text in the message box can be customized.

Note Carriage returns can be inserted into the text of the custom message by using the standard `\n` characters.

Example

In the following example, the **Alert** method of the **UI** object is used to display a message box:

```
XDocument.UI.Alert("Custom message text goes here.");
```

Avg Method

Returns a **Variant** that is the average value of all of the numerical elements in a node set.

```
expression.Avg(ByVal pxmllistInput As IXMLDOMNodeList) As Variant
```

expression Required. An expression that returns a reference to the [Math](#) object.

pxmllistInput Required **IXMLDOMNodeList**. The node set that contains the values to be averaged.

returns A **Variant** that represents the average value of all the numerical elements in a node set.

Security

0: Can be accessed without restrictions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the variable `averageAge` is set to the average value of all of the numerical elements in the `my:ages` node set.

```
var ages = XDocument.DOM.selectNodes("//my:ages");  
var averageAge = XDocument.Util.Math.Avg(ages);
```

BuildSQLFromXMLNodes Method

Returns a string containing an SQL command text fragment using the specified XML node.

```
expression.BuildSQLFromXMLNodes(ByRef pXmlNode As  
IXMLDOMNode) As String
```

expression Required. An expression that returns a reference to the [ADOAdapter](#) object.

pXmlNode Required **Object**. The XML node to be converted to an SQL fragment.

returns **String**.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The fragment of SQL that the **BuildSQLFromXMLNodes** method generates is an SQL WHERE clause in the form of field = value. The XML node that you use for the *pXmlNode* argument should be a descendant of the dfs:queryFields node; when you have the SQL command text fragment, you can add it to the existing SQL command string of the **ADOAdapter** object using the [Command](#) property.

Example

In the following example, the **BuildSQLFromXMLNodes** method of the **ADOAdapter** object is used to create an SQL command text fragment based on a specified XML node. This example is based on a form that uses the Orders table in the Microsoft SQL Server Northwind sample database.

```
function QueryGreaterThan()
{
    var objQueryFieldNode;
    var strWhereClause;
    var strOldCommand;
    var objQueryFieldAttributes;
    var objCurQueryFieldAttribute;

    // Build the WHERE clause from the QueryFields in the form's
    // underlying XML DOM.
    objQueryFieldNode = XDocument.DOM
        .selectSingleNode("dfs:myFields/dfs:queryFields/q:Orders");
    strWhereClause = XDocument.QueryAdapter
        .BuildSQLFromXMLNodes(objQueryFieldNode);

    // Replace the '=' signs with '>=', and append the clause to
    // the SQL command text.
    strWhereClause = strWhereClause.replace(/=/, ">=");
    strOldCommand = XDocument.QueryAdapter.Command;

    if (strWhereClause != "")
    {
        XDocument.QueryAdapter.Command = strOldCommand +
            " where " + strWhereClause;
    }
}
```

```
// Clear the QueryFields so the WHERE clause isn't
// automatically generated.
objQueryFieldAttributes = objQueryFieldNode.attributes;
while (objCurQueryFieldAttribute = objQueryFieldAttributes.nextNode()
{
    objCurQueryFieldAttribute.text = "";
}

// Perform the query.
try
{
    XDocument.Query();
}
catch (e)
{
    XDocument.UI.Alert("Failed to query.\n\n" + e.message);
}

// Reset the command so that subsequent queries are based on
// the correct SQL command text string.
XDocument.QueryAdapter.Command = strOldCommand;

// Clean up.
objQueryFieldNode = null;
strWhereClause = null;
strOldCommand = null;
objQueryFieldAttributes = null;
objCurQueryFieldAttribute = null;
}
```


CacheSolution Method (Application Object)

Examines the [form template](#) in the cache and, if necessary, updates it from the published location of the form template.

```
expression.CacheSolution(ByVal bstrSolutionURI As String)
```

expression Required. An expression that returns a reference to an [Application](#) object.

bstrSolutionURI Required **String**. The string that specifies the [Uniform Resource Identifier \(URI\)](#) of the form template. This parameter can be specified as a [form definition \(.xsf\) file](#) or a form template (.xsn) file.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

If the form template that currently exists in the cache matches the form template from the published location, no caching takes place. If the computer is offline and the form is already in the cache, the cache is kept and no update will occur.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following [Visual Basic for Applications \(VBA\)](#) example, the **CacheSolution** method of the **Application** object is used to cache a form template:

```
Public Sub CacheFormTemplate()

    Dim I As Integer
    Dim objApp As Object
    Dim aryForms(2) As String

    ' Create a reference to the Application object.
    Set objApp = CreateObject("InfoPath.Application")

    ' Populate the array with form template locations.
    aryForms(0) = "\\MyServer\MyForms\MyForm.xsn"
    aryForms(1) = "\\MyServer\MyForms\manifest.xsf"

    ' Loop through the array and cache the form templates.
    For I = 0 To UBound(aryForms) - 1
        objApp.CacheSolution(aryForms(I))
    Next I

End Sub
```


CacheSolution Method (ExternalApplication Object)

Examines the [form template](#) in the cache and, if necessary, updates it from the published location of the form template.

```
expression.CacheSolution(ByVal bstrSolutionURI As String)
```

expression Required. An expression that returns a reference to the [ExternalApplication](#) object.

bstrSolutionURI Required **String**. The string value that specifies the [Uniform Resource Identifier \(URI\)](#) of the form template. This parameter can be specified as a [form definition \(.xsf\) file](#) or a form template (.xsn) file

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

If the form template that currently exists in the cache matches the form template from the published location, no caching takes place. If the computer is offline and the form is already in the cache, the cache is kept and no update will occur.

Example

In the following example, which is written in the [Visual Basic for Applications \(VBA\)](#) programming language, the **CacheSolution** method of the **ExternalApplication** object is used to cache a form template:

```
Public Sub CacheFormTemplate()

    Dim I As Integer
    Dim objExternalApp As Object
    Dim aryForms(2) As String

    'Create a reference to the ExternalApplication object.
    Set objExternalApp = CreateObject("InfoPath.ExternalApplication")

    'Populate the array with form template locations.
    aryForms(0) = "\\MyServer\MyForms\MyForm.xsn"
    aryForms(1) = "\\MyServer\MyForms\manifest.xsf"

    'Loop through the array and cache the form templates.
    For I = 0 To UBound(aryForms) - 1
        objExternalApp.CacheSolution(aryForms(I))
    Next I

End Sub
```


CacheSolution Method (Index)

The **CacheSolution** method examines the [form template](#) in the cache and, if necessary, updates it from the published location of the form template. This method is implemented in several Microsoft Office InfoPath 2003 object model collections. Click a **CacheSolution** method link below to view the Help topic for a specific implementation of the **CacheSolution** method.

[CacheSolution](#) method as it applies to the [Application](#) object.

[CacheSolution](#) method as it applies to the [ExternalApplication](#) object.

Close Method (ExternalApplication Object)

Closes the specified Microsoft Office InfoPath 2003 form.

```
expression.Close(ByVal bstrDocumentURI As String)
```

expression Required. An expression that returns a reference to the [ExternalApplication](#) object.

bstrDocumentURI Required **String**. The string value that specifies the [Uniform Resource Identifier \(URI\)](#) of a form.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

The **Close** method closes the currently open form without quitting the InfoPath application. When using the **Close** method, the form is closed unconditionally, meaning that any changes made to the data in the form are not saved.

Example

In the following example, which is written in the [Visual Basic for Applications \(VBA\)](#) programming language, the **Close** method of the **ExternalApplication** object is used to close the currently open form:

```
Public Sub AutomateInfoPathForm()

    Dim objIP As Object

    'Create a reference to the ExternalApplication object.
    Set objIP = CreateObject("InfoPath.ExternalApplication")

    'Open an InfoPath form.
    objIP.Open ("C:\My Forms\Form1.xml")
    MsgBox ("The InfoPath form has been opened.")

    'Close the InfoPath form.
    objIP.Close ("C:\My Forms\Form1.xml")
    MsgBox ("The InfoPath form has been closed.")

    'Quit the InfoPath application.
    objIP.Quit
    MsgBox ("The InfoPath application has been closed.")

    Set objIP = Nothing

End Sub
```

Close Method (Index)

The **Close** method closes the item associated with the specified collection or object. This method is implemented in several Microsoft Office InfoPath 2003 object model collections and objects. Click a **Close** method link below to view the Help topic for a specific implementation of the **Close** method.

[Close](#) method as it applies to the [ExternalApplication](#) object.

[Close](#) method as it applies to the [Window](#) object.

[Close](#) method as it applies to the [XDocuments](#) collection.

Close Method (Window Object)

Closes the window that is represented by the [Window](#) object.

```
expression.Close([ByVal bForce As Boolean])
```

expression Required. An expression that returns a reference to the **Window** object.

bForce Optional **Boolean**. Default value is False. Determines whether open documents will be saved.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Close** method will close the associated window and the forms that it contains. If the **bForce** parameter is set to True, all forms will be closed without saving, even if they contain changes since they were last saved. If set to False, users will be prompted to save their changes.

The **Close** method can be used only with the editing window types; if used with a designing window type, it will return an error. In addition, the **Close** method can only be used with the OnSubmitRequest and OnClick [event handlers](#). If used with any other type of event handler, it will return an error.

Note If the window being closed is the only window open in Microsoft Office InfoPath 2003, the InfoPath application will also be closed.

Example

In the following example, the **Close** method of the **Window** object is used to close the currently active window, forcing a save if changes in the form have occurred:

```
Application.ActiveWindow.Close(false);
```


Close Method (XDocuments Collection)

Closes the specified Microsoft Office InfoPath 2003 form.

```
expression.Close(ByVal varIndex As Variant)
```

expression Required. An expression that returns a reference to the [XDocuments](#) collection.

varIndex Required **Variant**. The string value that specifies the [Uniform Resource Identifier \(URI\)](#) of a form, a long integer value that specifies the positional index of an [XDocument](#) object within the **XDocuments** collection, or a reference to an **XDocument** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Close** method closes the currently open form without quitting the InfoPath application. The form is closed unconditionally, meaning that any changes made to the data in the form are not saved.

Example

In the following example, the **Close** method of the **XDocuments** collection is used to close the currently open form using the positional index of an **XDocument** object contained in the collection:

```
Application.XDocuments.Close(0);
```

Alternatively, you can pass a reference to an **XDocument** object:

```
var objXDoc;
```

```
objXDoc = Application.XDocuments(0);  
Application.XDocuments.Close(objXDoc);
```

Confirm Method

Displays a message box with buttons for input from a user. The value that is returned is one of the [XdConfirmChoice](#) enumerated constants.

```
expression.Confirm(prompt As String, buttons As XdConfirmButtons)  
As XdConfirmChoice
```

expression Required. An expression that returns a reference to the [UI](#) object

bstrPrompt Required **String**. The text message to be displayed.

IButtons Required [XdConfirmButtons](#). Specifies the number and type of buttons to display. You may specify any of the values of the **XdConfirmButtons** enumeration.

Security

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Remarks

Example

In the following example, the **Confirm** method of the **UI** object is used to display a dialog box with **Yes** and **No** buttons:

```
XDocument.UI.Confirm("Do you wish to continue?", 4);
```

Create Method

Creates a new [Signature](#) object. This method can be called only from the [OnSign](#) event handler.

```
expression.Create()
```

expression Required. An expression that returns a [Signatures](#) collection.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

The digital signature is not actually written to the file until the [Sign](#) method is called for the newly created **Signature** object.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

CreateDOM Method

Creates a new instance of the XML [Document Object Model \(DOM\)](#) in memory.

expression.**CreateDOM()** As XMLDOMDocument

expression Required. An expression that returns a reference to an [XDocument](#) object.

returns A reference to the XML DOM.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Using the **CreateDOM** method to create an instance of the XML DOM is equivalent to using the following method of creating a Microsoft XML Core Services (MSXML) 5.0 **DOMDocument** object:

```
var objDoc = new ActiveXObject("Msxml2.DOMDocument.5.0");
```

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **CreateDOM** method of the **XDocument** object is used to create an instance of XML DOM and assign it to a variable.

```
var objDOM = XDocument.CreateDOM();
```


Delete Method

Deletes the specified [Error](#) object from the [Errors](#) collection.

```
expression.Delete(ByVal varNode As Variant, ByVal  
bstrConditionName As String)
```

expression Required. An expression that returns a reference to the **Errors** collection.

varNode Required **Variant**. The XML [Document Object Model \(DOM\)](#) node associated with the error.

bstrConditionName Required **String**. The name of the error.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Delete** method deletes all the **Error** objects in the **Errors** collection that are associated with the specified XML node and that have the same name. To delete all of the **Error** objects contained in the **Errors** collection, use the [DeleteAll](#) method.

Note The **Delete** method will only delete errors that were created using the [Add](#) method of the **Errors** collection. It cannot be used to delete errors that occur because of schema or data validation constraints, or errors that were created using the [ReportError](#) method of the [DataDOMEvent](#) object.

Example

In the following example, the **Delete** method of the **Errors** collection is used to delete all the errors based on their associated XML node and name:

```
XDocument.Errors.Delete(MyXMLNode, "ValidationError");
```

DeleteAll Method

Deletes all of the [Error](#) objects contained in the [Errors](#) collection.

```
expression.DeleteAll()
```

expression Required. An expression that returns a reference to the **Errors** collection.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

To delete a specific **Error** object from the **Errors** collection, use the [Delete](#) method.

Note Unlike the **Delete** method that will only delete errors that were created using the [Add](#) method of the **Errors** collection, the **DeleteAll** method will delete all errors in the **Errors** collection, regardless of how they were created.

Example

In the following example, the **DeleteAll** method of the **Errors** collection is used to delete all the errors that it contains:

```
XDocument.Errors.DeleteAll();
```

DisableAutoUpdate Method

Disables automatic synchronization between a form's underlying XML document and the associated [View](#) object.

```
expression.DisableAutoUpdate()
```

expression Required. An expression that returns a reference to the **View** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The views in a Microsoft Office InfoPath 2003 form are automatically synchronized with the data that is contained in a form's underlying XML document. You can override this by using the **DisableAutoUpdate** method. You may need to do this for performance reasons, such as when you are programatically making many changes to a form's underlying XML document and you do not want the view to be refreshed until the changes are complete.

Automatic synchronization can be enabled using the [EnableAutoUpdate](#) method of the **View** object.

Example

In the following example, the **DisableAutoUpdate** method of the **View** object is used to disable synchronization between a form's underlying XML document and the view that it is associated with:

```
XDocument.View.DisableAutoUpdate();
```

Enable Method

Defines a method that must be provided by the developer for InfoPath to call when it needs to enable or disable an instance of the control in a view.

expression.**Enable**(*vfEnabled* As Boolean)

expression Required. An expression that returns a reference to the [InfoPathControl](#) object.

vfEnabled Required. A **Boolean** value that specifies whether the control is enabled.

Remarks

InfoPath will call the **Enable** method to enable or disable the control, such as when the view is being refreshed or closed, when the document is signed and should not be edited, or when rules are applied to the form that must disable the control.

The **InfoPathControl** and [InfoPathControlSite](#) objects and their methods and properties are designed to be used only from the implementation of an ActiveX control. These objects and their members are not supported in InfoPath form code. For more information on how to create ActiveX controls that work with InfoPath, see the [InfoPath Developer Center](#).

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form template that contains a view with an ActiveX control that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

EnableAutoUpdate Method

Enables automatic synchronization between a form's underlying XML document and the associated [View](#) object.

expression.**EnableAutoUpdate**()

expression Required. An expression that returns a reference to the **View** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The views in a Microsoft Office InfoPath 2003 form are automatically synchronized with the data that is contained in a form's underlying XML document. However, this can be overridden using the [DisableAutoUpdate](#) method. To re-enable synchronization, use the **EnableAutoUpdate** method.

Example

In the following example, the **EnableAutoUpdate** method of the **View** object is used to enable synchronization between a form's underlying XML document and the view that it is associated with:

```
XDocument.View.EnableAutoUpdate();
```

Eval Method

Returns a **Variant** containing the set of results calculated when the expression is applied to each set of elements in the context.

```
expression.Eval(ByVal nodeList As IXMLDOMNodeList, ByVal  
bstrExpression as String) As Variant
```

expression Required. An expression that returns a reference to the [Math](#) object.

nodeList Required **IXMLDOMNodeList**. The node that sets the context for the expression.

bstrExpression Required **String**. The expression to be applied to each set of nodes in the specified context.

Security Level

0: Can be accessed without restrictions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the variable `maxCost` is set to the largest cost value, where cost is calculated by multiplying price by quantity.

```
var nodes = XDocument.DOM.selectNodes("/my:items/my:item");  
var maxCost = XDocument.Util.Math.Max(XDocument.Util.Math.Eva
```


ExecuteAction Method

Executes a Microsoft Office InfoPath 2003 editing command against a form's underlying XML document, based on the data selected in the view that is associated with the [View](#) object.

```
expression.ExecuteAction(ByVal bstrAction As String, [ByVal  
varXmlToEdit As Variant])
```

expression Required. An expression that returns a reference to the **View** object.

bstrAction Required **String**. The name of the editing action to perform.

varXmlToEdit Optional **Variant**. The name of the field or group to which to apply the editing action. This is equivalent to the value of the [name](#) attribute in the [xmlToEdit](#) element of the [form definition \(.xsf\) file](#).

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **ExecuteAction** method is used to programmatically perform built-in InfoPath editing actions against a form's underlying XML document, based on the selected context in a view.

The action that is executed will be the same action that would be used when clicking on an equivalent menu or toolbar button; namely one for which the button element in the .xsf file has corresponding [xmlToEdit](#) and [action](#) attributes. As with using a button, the action will be based on current selection: it will act on the selected context (and in the case where the selection would lead the button to be disabled, then the **ExecuteAction** method will have no effect).

Note It is possible with scripting code to first set the selection context by using the [SelectNodes](#) or [SelectText](#) methods of the **View** object, then calling the **ExecuteAction** method to act on that context.

The **ExecuteAction** method will return an error for the following reasons:

The ***bstrAction*** parameter does not contain a valid editing component name.

The ***varXmlToEdit*** parameter does not match an editing component that is defined in the view.

The ***varXmlToEdit*** parameter is required for a specific editing action.

The editing action is not applicable to the selected context.

▶ [Valid parameter combinations](#)

Note In some cases, calling the **ExecuteAction** method from the [OnClick](#) event for a button in a view may cause an error. This is because the selected context changes to the button when the button is clicked. In this case, it is better to use a button (or link) on a custom task pane, toolbar, or menu to call the **ExecuteAction** method.

Example

In the following example, the **ExecuteAction** method of the **View** object is used to delete selected data and place it on the clipboard:

```
XDocument.View.ExecuteAction("Cut");
```

In the following example, the **ExecuteAction** method of the **View** object is used to insert data using the xCollection editing component, based on the selected context:

```
XDocument.View.ExecuteAction("xCollection::insert", "group1_1");
```

Export Method

Exports the view to a file of the specified format.

```
expression.Export(ByVal bstrURL As String, ByVal bstrFormat As String)
```

expression Required. An expression that returns a reference to the [View](#) object.

bstrURL Required **String**. The directory location that the exported view file will be written to.

bstrFormat Required **String**. The type of file format to export to. Only the "MHT" value is supported.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

If used in a form that is not fully trusted, the **Export** method will return a "permission denied" error.

Example

In the following example, the **Export** method of the **View** object is used to export the current view:

```
XDocument.View.Export("C:\\MyView", "MHT");
```


ForceUpdate Method

Forces synchronization between a form's underlying XML document and the associated [View](#) object.

expression.**ForceUpdate**()

expression Required. An expression that returns a reference to the **View** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The views in a Microsoft Office InfoPath 2003 form are automatically synchronized with the data that is contained in a form's underlying XML document. However, you can force synchronization to occur using the **ForceUpdate** method. This is also useful when data in a [secondary data source](#) has changed and needs to be refreshed in the view.

Automatic synchronization can be disabled using the [DisableAutoUpdate](#) method and enabled using the [EnableAutoUpdate](#) method.

Example

In the following example, the **ForceUpdate** method of the **View** object is used to force synchronization between a form's underlying XML document and the view that it is associated with:

```
XDocument.View.ForceUpdate();
```


FormatString Method

Formats the specified string or XML node according to the specified category and options parameters.

```
expression.FormatString(ByVal varInput, ByVal bstrCategory As String, ByVal bstrOptions As String) As String
```

expression Required. An expression that returns a reference to the [Application](#) object.

varInput Required **String** or **XML node**. The string value or XML node to be formatted.

bstrCategory Required **String**. The string value that specifies the category used for formatting. Values include **number**, **percentage**, **currency**, **date**, **time**, and **datetime**.

bstrOptions Required **String**. The string value that specifies the options used for formatting. Takes the form of a case-sensitive string in the format "optionName:value".

returns **String**.

Security Level

0: Can be accessed without restrictions.

Remarks

The **FormatString** method can be used anywhere in your scripting code in which you need to change the format of a specified string. It must be qualified with the **Application** object name, and its return value must be assigned to a variable or used as an expression that requires a string value.

The following sections list the values that may be used for the ***bstrCategory*** and ***bstrOptions*** parameters:

- ▶ [List of categories](#)
- ▶ [List of options](#)

Example

In the following example, the **FormatString** method of the **Application** object is used to format the specified date string into a long date format:

```
Application.FormatString("2003-01-08", "date", "dateFormat:Long D
```


GenerateDataSetDiffGram Method

Returns an XML DataSet, containing an inline schema describing the data and the DataSet's DiffGram.

```
expression.GenerateDataSetDiffGram(ByVal pNode As  
IXMLDocumentObject) As IXMLDocumentObject
```

expression Required. An expression that returns a reference to a [WebServiceAdapter](#) object.

pNode Required **IXMLDocumentObject**. An XML [Document Object Model \(DOM\)](#) node that contains the XML data of the DataSet whose DiffGram will be created.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The DiffGram for the input *pNode* is generated using the sibling node `originalData` to compute the difference between the `originalData` and the input *pNode*.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

GetContextNodes Method

Returns a reference to an [XMLNodes](#) collection that is populated with XML [Document Object Model \(DOM\)](#) nodes based on the current context. It consists of the sequence of XML DOM nodes that are mapped from the view, corresponding to the current [XSL Transformation \(XSLT\)](#) node, starting at the current selection and walking up through the view ancestors to the BODY tag.

```
expression.GetContextNodes([ByVal varNode As Variant], [ByVal varViewContext As Variant]) As XMLNodes
```

expression Required. An expression that returns a reference to the [View](#) object.

varNode Optional **Variant**. An XML DOM node.

varViewContext Optional **Variant**. The ID of the control that is used for the context.

returns A reference to the **XMLNodes** collection.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If no parameters are used, the context nodes are based on the current selection. If parameters are used, then the context nodes returned are those that would be returned based on the selection that would be obtained from calling the [SelectNodes](#) method of the **View** object.

Example

In the following partial example from the Structural Editing developer sample form, the **GetContextNodes** method of the **View** object is used to return a collection of XML DOM nodes based on the current context. Then the code loops through the collection of XML DOM nodes looking for a particular node. When it is found, the code calls the custom `ApplyDiscountToItem` function to update the data that the node contains.

```
objContextNodes = XDocument.View.GetContextNodes();

// Scan the list of context nodes for an item node and if one is found
// apply the discount to it.
for (var i = 0; i < objContextNodes.Count; i++)
{
    if (objContextNodes.item(i).nodeName == "item")
    {
        ApplyDiscountToItem(objContextNodes.item(i), intPercentage);
        blnAppliedDiscount = true;
        break;
    }
}
```


GetDataVariable Method

Returns a string containing the value of the specified variable, which is a predefined variable stored as a [processing instruction](#) attribute in the form's underlying XML document.

expression.**GetDataVariable**(ByVal *IVariableNumber* As Long) As String

expression Required. An expression that returns a reference to an [XDocument](#) object.

IVariableNumber Required **Long**. The number of the variable.

returns **String**.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the variable is not defined or is empty, the **GetDataVariable** method will return an empty string. To set a variable, use the [SetDataVariable](#) method of the **XDocument** object.

Note InfoPath only supports using the initialView variable, which is the variable used to specify the initial view displayed when a form is opened. The number of this variable is always 1, and its value must be the name of a view within the form.

Example

In the following example, the **GetDataVariable** method of the **XDocument** object is used to return the value of the first variable:

```
var strVariable1;  
strVariable1 = XDocument.GetDataVariable(1);
```


GetDOM Method

Returns a reference to the XML [Document Object Model \(DOM\)](#) of the specified [DataObject](#) object associated with the [XDocument](#) object.

```
expression.GetDOM(ByVal bstrName As String) As XMLDOMDocument
```

expression Required. An expression that returns a reference to an [XDocument](#) object.

bstrName Required **String**. The name of a [DataObject](#) object.

returns A reference to an XML DOM document object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

After you have a reference to the XML DOM that the **GetDOM** method returns, you can use any of the properties and methods that the XML DOM supports to manipulate the data that the DOM contains.

Note For more information about the XML DOM, see the MSXML 5.0 SDK documentation in the [Microsoft Script Editor \(MSE\)](#) Help system.

Example

In the following example, the **GetDOM** method of the **XDocument** object is used to set a reference to the XML DOM that it returns, which in this case is the **DataObject** object named CityDropDownList:

```
var objXml;
```

```
objXml = XDocument.GetDOM("CityDropDownList");
```

GetNamedNodeProperty Method

Returns the value of a named property for the specified XML node, which must be a nonattribute node in the main data source.

```
expression.GetNamedNodeProperty(ByVal varMainDOMNode As Variant, ByVal bstrPropertyName As String, ByVal bstrDefaultValue As String) As String
```

expression Required. An expression that returns a reference to an [XDocument](#) object.

varMainDOMNode Required **Variant**. An XML node corresponding to a nonattribute node in the main data source, for which a named property is to be set.

bstrPropertyName Required **String**. Specifies the name of the property whose value is to be returned.

bstrDefaultValue Required **String**. Specifies the default value to be returned if the property has not been set.

returns A string corresponding to the current value of the named property for the specified XML node in the main data source. If the specified property has not been set for this XML node, the specified default string is returned.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Named properties allow users to associate strings with user-defined properties of XML element nodes in the main data source. The value of a named property can be set by using the [SetNamedNodeProperty](#) method. Use the **GetNamedNodeProperty** method to read the value of a named property.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

The following example demonstrates setting and getting the value of a named property (with the name "cost") of an XML node (called "item"):

```
var objXMLNode = XDocument.DOM.selectSingleNode("/items/item")
var strTest = XDocument.GetNamedNodeProperty(objXMLNode, 'cost')

// The value of the "cost" named property is set to 100.
XDocument.SetNamedNodeProperty(objXMLNode, 'cost', '100');

strTest = XDocument.GetNamedNodeProperty(objXMLNode, 'cost',
```

In the following XSL example, the "cost" named property of the item node is displayed:

```
<xsl:value-of select="xdXDocument:GetNamedNodeProperty(item,
```


GetSelectedNodes Method

Returns a reference to an [XMLNodes](#) collection that is populated with XML [Document Object Model \(DOM\)](#) nodes based on the current selection of items in a view.

expression.**GetSelectedNodes()** As XMLNodes

expression Required. An expression that returns a reference to the [View](#) object.

returns A reference to the **XMLNodes** collection.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If no items are selected in a view, or if only text is selected, then the **GetSelectedNodes** method returns an empty collection.

Example

In the following example, the **GetSelectedNodes** method of the **View** object is used to set a reference to a collection of XML DOM nodes based on what is currently selected in the view. Then the code determines whether nodes were returned and, if they were, displays information about the first node found in the **XMLNodes** collection in a message box.

```
objXMLNodes = XDocument.View.GetSelectedNodes();
```

```
if (objXMLNodes.Count > 0)
```

```
{
```

```
    XDocument.UI.Alert(objXMLNodes(0).nodeName + "\n\n" + objXMLNodes(0).nodeValue);
```

```
}
```

ImportDOM Method

Imports the specified XML data into the current form.

```
expression.ImportDOM(ByVal pxDoc As IXMLDocumentObject)
```

expression Required. An expression that returns a reference to an [XDocument](#) object.

pxDoc Required **IXMLDocument**. The XML data that is to be imported (merged) into the currently open form.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Using the object model to import a form programmatically is equivalent to performing a merge operation using the **Merge Forms** command on the **File** menu in InfoPath.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **ImportDOM** method of the **XDocument** object is used to import a form from the [OnMergeRequest](#) event handler:

```
XDocument::OnMergeRequest(eventObj)
{
    XDocument.ImportDOM(eventObj.DOM);
    eventObj.ReturnStatus = true;
}
```


ImportFile Method

Imports the specified form into the current form.

```
expression.ImportFile(ByVal bstrFileURI As String)
```

expression Required. An expression that returns a reference to an [XDocument](#) object.

bstrFileURI Required **String**. The [Uniform Resource Identifier \(URI\)](#) of the form that is to be imported (merged) into the currently open form.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Using the object model to programmatically import a form is equivalent to performing a merge operation in the user interface.

Example

In the following example, the **ImportFile** method of the **XDocument** object is used to import a form:

```
XDocument.ImportFile("C:\SomeOtherForm.xml");
```

Init Method

Defines a method that must be provided by the developer to perform any initialization routines required when an instance of the control is added to an InfoPath form.

expression.**Init**(*pControlSite* As InfoPathControlSite)

expression Required. An expression that returns a reference to the [InfoPathControl](#) object.

pControlSite Required. An instance of the [InfoPathControlSite](#) object.

Remarks

InfoPath calls the **Init** method when a user adds an instance of the control to a view. InfoPath passes an instance of the **InfoPathControlSite** object to the **Init** method when the control is initialized. The **InfoPathControlSite** object provides the [Node](#) property that provides access to the XML DOM node to which the control is bound, and the [XDocument](#) property for accessing the [XDocument](#) object associated with a form, which in turn provides access to the full InfoPath object model.

The **InfoPathControl** and **InfoPathControlSite** objects and their methods and properties are designed to be used only from the implementation of an ActiveX control. These objects and their members are not supported in InfoPath form code. For more information on how to create ActiveX controls that work with InfoPath, see the [InfoPath Developer Center](#).

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form template that contains a view with an ActiveX control that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

IsCurrentUser Method

Returns **True** if the current user matches the specified user name.

```
expression.IsCurrentUser(ByVal bstrUsername As String) As Boolean
```

expression Required. An expression that returns a reference to the [User](#) object.

bstrUsername Required **String**. The user name in the format of "domain\username."

returns A **Boolean** value indicating whether the specified user name matches the name of the current user.

Security Level

0: Can be accessed without restrictions.

Remarks

Even though the **IsCurrentUser** method is marked as security level 0, it is not always accessible. When a call is made to the **IsCurrentUser** method, InfoPath first performs a security check to determine if access to this method is allowed. The security check confirms whether the calling code is trusted or not and it determines the location of the calling code.

If the calling code is trusted (as it is when, for example, the **IsCurrentUser** method is called from an installed or signed InfoPath [form template](#), or from trusted external code, such as an executable file on the local computer), InfoPath will allow full access to the **IsCurrentUser** method.

If the calling code is not trusted (as in the circumstance of a call coming from a domain-based InfoPath form template or from script executing in Microsoft Internet Explorer), InfoPath first checks where the call is being made from. If the call is from code that is not trusted in an InfoPath form template at a URL such as "http://www.contoso.com/example.xsn", then the call is from the Internet zone. InfoPath denies access to the **IsCurrentUser** method for all calls that are not trusted coming from the Internet zone. If the call is from a URL such as "http://contoso/example.xsn", then the call is from the Intranet zone. For a call that is not trusted from the Intranet zone, InfoPath checks whether the Internet Explorer user authentication settings allow automatic logon (in the Intranet zone only or for every logon). If Internet Explorer is configured for automatic logon, then InfoPath allows full access to the **IsCurrentUser** method. Otherwise, access to the **IsCurrentUser** method is denied.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **IsCurrentUser** method of the **User** object is used to determine if the current user equals "UserDomain\NancyDavilio".

```
var fUserMatched;  
fUserMatched = Application.User.IsCurrentUser("UserDomain\Nanc
```


IsDestinationReachable Method

Returns a **Boolean** value indicating whether the specified [Uniform Resource Locator \(URL\)](#), [universal naming convention \(UNC\)](#) path, or [IP address](#) of the destination computer can be connected to from the client computer.

```
expression.IsDestinationReachable(ByVal bstrDestination As String,  
[ByVal bstrBehavior As String ] As Boolean
```

expression Required. An expression that returns a reference to the [Application](#) object.

bstrDestination Required **String**. Specifies the location to check for network connectivity. IP addresses, a UNC paths, or a URLs are acceptable values.

returns A **Boolean** value indicating whether the destination is reachable. **True** if the specified destination can be connected to; otherwise, **False**.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Note If an IP address is specified, the security level for this method becomes level 3. It is not possible to determine the domain from the IP address, therefore the caller must have the full trust permission.

Remarks

For UNC and URL strings, valid values are those including only the server name, for example `http://MyServer` or `\\MyServer`. Values such as `http://MyServer/MyVirtualDirectory` or `\\MyServer\MyShare` are not considered valid.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

IsUserMemberOf Method

Returns a **Boolean** value that indicates whether the current user is a member of the specified group.

```
expression.IsUserMemberOf(ByVal bstrGroupname As String) As Boolean
```

expression Required. An expression that returns a reference to the **User** object.

bstrGroupname Required **String**. The group name in the format of "domain\groupname."

Security Level

0: Can be accessed without restrictions.

Remarks

Even though the **IsUserMemberOf** method is marked as security level 0, it is not always accessible. When a call is made to the **IsUserMemberOf** method, InfoPath first performs a security check to determine if access to this method is allowed. The security check confirms whether the calling code is trusted or not and it determines the location of the calling code.

If the calling code is trusted (as it is when, for example, the **IsUserMemberOf** method is called from an installed or signed InfoPath [form template](#), or from trusted external code, such as an executable file on the local computer), InfoPath will allow full access to the **IsUserMemberOf** method.

If the calling code is not trusted (as in the circumstance of a call coming from a domain-based InfoPath form template or from script executing in Microsoft Internet Explorer), InfoPath first checks where the call is being made from. If the call is from code that is not trusted in an InfoPath form template at a URL such as "http://www.contoso.com/example.xsn", then the call is from the Internet zone. InfoPath will deny access to the **IsUserMemberOf** method for all calls that are not trusted coming from the Internet zone. If the call is from a URL such as "http://contoso.com/example.xsn", then the call is from the Intranet zone. For a call that is not trusted from the Intranet zone, InfoPath checks whether the Internet Explorer user authentication settings allow automatic logon (in the Intranet zone only or for every logon). If Internet Explorer is configured for automatic logon, then InfoPath checks the user's group membership. If the group membership is public, InfoPath allows full access to the **IsUserMemberOf** method. If the group membership is not fully public, InfoPath hides the result. That is, InfoPath treats the group membership as if it were not visible. The return value in this case is always **False** because the group membership is not public. InfoPath does not report that access is denied.

Note Group membership information is fully public if access to it is allowed and the information is visible by all non-anonymous users. If even a single non-anonymous user has been specifically denied access to membership information, then the group membership information is not

fully public.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **IsUserMemberOf** method of the **User** object is used to determine if the current user is a member of the "GroupDomain\Administrators" group.

```
var fGroupMatched;  
fGroupMatched = Application.User.IsUserMemberOf("GroupDomain
```

Match Method

Returns a **Boolean** value that indicates whether the test matches the specified pattern.

```
expression.Match(ByVal bstrValue As String, ByVal bstrPattern As String) As Boolean
```

bstrValue Required **String**. The string to test against the pattern.

bstrPattern Required **String**. The pattern to use.

returns A **Boolean** value that indicates whether the string matches the pattern.

Security Level

0: Can be accessed without restrictions.

Remarks

The **Match** method can be used to test any string against a regular expression. The regular expression must conform to the W3C's XML Schema specification for regular expressions (<http://www.w3.org/TR/xmlschema-2/#regexs>).

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Note The XML Schema specification for regular expressions is different from regular expressions in Perl.

Example

In the following example, the variable `isSSNValid` is set to a value that indicates whether or not the value stored in the `my:SSN` node is a legal Social Security Number:

```
var SSN = XDocument.DOM.selectSingleNode("//my:SSN");  
var isSSNValid = XDocument.Util.Match(SSN.text, "\\d\\d\\d-\\d\\d-\\d\\d");
```

Max Method

Returns a **Variant** that is the largest value of all of the numerical elements in a node set.

```
expression.Max(ByVal pxmllistInput As IXMLDOMNodeList) As Variant
```

expression Required. An expression that returns a reference to the [Math](#) object.

pxmllistInput Required **IXMLDOMNodeList**. The node set to search for the largest value.

returns A **Variant** that represents the largest value of all the numerical elements in a node set.

Security Level

0: Can be accessed without restrictions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, maxAge is set to the maximum value of all of the numerical elements in the my:ages node set:

```
var ages = XDocument.DOM.selectNodes("//my:ages");  
var maxAge = XDocument.Util.Math.Max(ages);
```

Min Method

Returns a **Variant** that is the smallest value of all of the numerical elements in a node set.

```
expression.Min(ByVal pxmllistInput As IXMLDOMNodeList) As Variant
```

expression Required. An expression that returns a reference to the [Math](#) object.

pxmllistInput Required **IXMLDOMNodeList**. The node set to search for the smallest value.

returns A **Variant** that represents the smallest value of all the numerical elements in a node set.

Security Level

0: Can be accessed without restrictions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the variable `minAge` is set to the minimum value of all of the numerical elements in the `my:ages` node set:

```
var ages = XDocument.DOM.selectNodes("//my:ages");  
var maxAge = XDocument.Util.Math.Min(ages);
```


Navigate Method

Loads the specified [HTML](#) document into the Microsoft Office InfoPath 2003 [custom task pane](#).

```
expression.Navigate(ByVal bstrURL As String)
```

expression Required. An expression that returns a reference to the [HTMLTaskPane](#) object.

bstrURL Required **String**. The [Uniform Resource Locator \(URL\)](#) of the HTML document to navigate to.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Navigate** method of the **HTMLTaskPane** object is one of the methods inherited by the [TaskPane](#) object when the type of the task pane is 0, which means that it is the custom task pane.

Note The **Navigate** method cannot be called during an [OnLoad](#) event because the view is not yet loaded when this event occurs, and task panes are associated with the view.

Example

In the following example, the **Navigate** method of the **HTMLTaskPane** object is used to load an HTML document into the custom task pane. The HTML document that it loads is one that is included in the [form files](#) of the [form template](#):

```
var objTaskPane;  
  
// Set a reference to the custom task pane.  
objTaskPane = XDocument.View.Window.TaskPanes(0);  
objTaskPane.Navigate("taskpane2.htm");
```


New Method (ExternalApplication Object)

Creates a new Microsoft Office InfoPath 2003 form based on a specified form.

```
expression.New(ByVal bstrDocumentURI As String, [ByVal dwBehavior  
As Long = 1])
```

expression Required. An expression that returns a reference to the [ExternalApplication](#) object.

bstrDocumentURI Required **String**. The string value that specifies the [Uniform Resource Identifier \(URI\)](#) of a form.

dwBehavior Optional **Long**. Default value is 1. A long value that specifies how the form should be opened. The values are based on the [XdDocumentVersionMode](#) enumeration.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

The **New** method can be used to only create a new form based on an existing form; it cannot be used to create a new form based on a [form template](#). To create a form from a form template, use the [NewFromSolution](#) method of the **ExternalApplication** object.

When you use the **New** method, InfoPath is opened and the new form is ready to be filled out.

Note You cannot use the [Close](#) method of the **ExternalApplication** object to close a form that has been opened with the **New** method. When the **New** method creates a form, the name of that form is not yet known.

Example

In the following example, which is written in the [Visual Basic for Applications \(VBA\)](#) programming language, the **New** method of the **ExternalApplication** object is used to create a new form based on a specified form:

```
Public Sub CreateFromForm()
```

```
    Dim objIP As Object
```

```
    'Create a reference to the ExternalApplication object.
```

```
    Set objIP = CreateObject("InfoPath.ExternalApplication")
```

```
    'Create an InfoPath form.
```

```
    objIP.New ("C:\My Forms\Form1.xml")
```

```
    MsgBox ("The InfoPath form has been created.")
```

```
    Set objIP = Nothing
```

```
End Sub
```

New Method (Index)

The **New** method creates a new item based on a form. This method is implemented in a number of Microsoft Office InfoPath 2003 object model collections and objects. Click a **New** method link below to view the Help topic for a specific implementation of the **New** method.

[New](#) method as it applies to the [ExternalApplication](#) object.

[New](#) method as it applies to the [XDocuments](#) collection.

New Method (XDocuments Collection)

Creates a new Microsoft Office InfoPath 2003 form based on the specified form.

```
expression.New(ByVal varURI As Variant, [ByVal dwBehavior As Long = 1]) As XDocument
```

expression Required. An expression that returns a reference to the [XDocuments](#) collection.

varURI Required **Variant**. Specifies the [Uniform Resource Identifier \(URI\)](#) of a form.

dwBehavior Optional **Long**. Default value is 1. A long value that specifies how the form should be opened. The values are based on the [XdDocumentVersionMode](#) enumeration.

returns A reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **New** method can only be used to create a new form based on an existing form; it cannot be used to create a new form based on a [form template](#). To create a form from a form template, use the [NewFromSolution](#) method of the **XDocuments** collection.

When you use the **New** method, the new form opens in InfoPath and is ready to be filled out.

Note If you use the optional *dwBehavior* argument in the **New** method, you can only pass the numerical value of the **XdDocumentVersionMode** enumeration. Because InfoPath uses scripting languages for working with the object model, named values cannot be used.

Example

In the following example, the **New** method of the **XDocuments** collection is passed the URI of an existing form, and a new form is created and its associated **XDocument** object returned:

```
var objXDoc;
```

```
objXDoc = Application.XDocuments.New("C:\\MyForm.xml");
```


NewADODBConnection Method

Creates and returns a reference to an empty [ActiveX Data Objects \(ADO\) Connection](#) object.

expression.NewADODBConnection() ADODB.Connection

expression Required. An expression that returns a reference to the [Application](#) object.

returns A reference to an ADO **Connection** object.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

If the form is not fully trusted, the **NewADODBConnection** method will return a "permission denied" error.

Example

In the following example, the **NewADODBConnection** method of the **Application** object is used to set a reference to an empty ADO **Connection** object:

```
var objADOConnection;
```

```
objADOConnection = Application.NewADODBConnection();
```


NewADODBRecordset Method

Creates and returns a reference to an empty [ActiveX Data Objects \(ADO\) Recordset](#) object.

expression.NewADODBRecordset() ADODB.Recordset

expression Required. An expression that returns a reference to the [Application](#) object.

returns A reference to an ADO **Recordset** object.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

If the form is not fully trusted, the **NewADODBRecordset** method will return a "permission denied" error.

Example

In the following example, the **NewADODBRecordset** method of the **Application** object is used to set a reference to an empty ADO **Recordset** object:

```
var objADORecordset;
```

```
objADORecordset = Application.NewADODBRecordset();
```


NewFromSolution Method (ExternalApplication Object)

Creates a new Microsoft Office InfoPath 2003 form based on the specified form template.

expression.NewFromSolution(ByVal *bstrSolutionURI* As String)

expression Required. An expression that returns a reference to the [ExternalApplication](#) object.

bstrSolutionURI Required **String**. The string value that specifies the [Uniform Resource Identifier \(URI\)](#) of a form template.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

The **NewFromSolution** method can be used only to create a new form based on an existing form template; it cannot be used to create a new form based on an existing form. To create a form from an existing form, use the [New](#) method of the **ExternalApplication** object.

When you use the **NewFromSolution** method, InfoPath is opened and the new form is ready to be filled out.

Note You cannot use the [Close](#) method of the **ExternalApplication** object to close a form that has been opened with the **NewFromSolution** method. When the **NewFromSolution** method creates a form, the name of that form is not yet known.

Example

In the following example, which is written in the [Visual Basic for Applications \(VBA\)](#) programming language, the **NewFromSolution** method of the **ExternalApplication** object is used to create a new form based on a specified form template:

```
Public Sub CreateFromFormTemplate()
```

```
    Dim objIP As Object
```

```
    'Create a reference to the ExternalApplication object.
```

```
    Set objIP = CreateObject("InfoPath.ExternalApplication")
```

```
    'Create an InfoPath form from a form template.
```

```
    objIP.NewFromSolution ("C:\My Forms\MyFormTemplate.xsn")
```

```
    MsgBox ("The InfoPath form has been created.")
```

```
    Set objIP = Nothing
```

```
End Sub
```


NewFromSolution Method (Index)

The **NewFromSolution** method creates a new item based on a [form template](#). This method is implemented in a number of Microsoft Office InfoPath 2003 object model collections and objects. Click a **NewFromSolution** method link below to view the Help topic for a specific implementation of the **NewFromSolution** method.

[NewFromSolution](#) method as it applies to the [ExternalApplication](#) object.

[NewFromSolution](#) method as it applies to the [XDocuments](#) collection.

NewFromSolution Method (XDocuments Collection)

Creates a new Microsoft Office InfoPath 2003 form based on the specified form template.

expression.**NewFromSolution**(ByVal *varURI* As Variant) As XDocument

expression Required. An expression that returns a reference to the [XDocuments](#) collection.

varURI Required **Variant**. Specifies the [Uniform Resource Identifier \(URI\)](#) of a form.

returns A reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **NewFromSolution** method can only be used to create a new form based on an existing [form template](#); it cannot be used to create a new form based on a form. To create a form from an existing form, use the [New](#) method of the **XDocuments** collection.

When you use the **NewFromSolution** method, the new form opens in InfoPath and is ready to be filled out.

Example

In the following example, the **NewFromSolution** method of the **XDocuments** collection is passed the URI of an existing [form template](#), and a new form is created and its associated **XDocument** object returned:

```
var objXDoc;
```

```
objXDoc = Application.XDocuments.NewFromSolution("C:\\MyForm
```


NewFromSolutionWithData Method

Creates a new Microsoft Office InfoPath 2003 form using the specified XML data and form template.

```
expression. NewFromSolutionWithData(ByVal varXMLData As Variant, ByVal varSolutionURI As Variant, [ByVal dwBehavior As Long = 1]) As XDocument
```

expression An expression that returns a reference to an [XDocuments](#) collection.

varXMLData Required **Variant**. Provides the XML data to be used as a template for the form. Can be a string that specifies the [Uniform Resource Identifier \(URI\)](#) of an XML document, or an **XMLDOMNode** that contains the XML to be used as the XML document (template).

varSolutionURI Required **Variant**. String which specifies the Uniform Resource Identifier (URI) of a form template file (an .xsf or .xsn file).

dwBehavior Optional **Long**. Reserved for future use. This value should be omitted or set to 1.

returns A reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The following related methods of the **XDocuments** collection are also available:

The [New](#) method enables creating a new instance of InfoPath by using a specified XML document. The XML document must correspond to an InfoPath form. A new form is opened in InfoPath, using the supplied XML document as initial data, and its associated form template as specified in the [processing instructions](#) in the header of the document.

The [NewFromSolution](#) method enables creating a new instance of InfoPath using a specified form template, and its associated XML form (the initial template data).

The **NewFromSolutionWithData** method supports a scenario that is not provided by the foregoing two methods: specifying both the XML document used as initial data and the form template. The XML document does not need to have been created by InfoPath. It can be supplied either as a URI or as an **XMLDOMNode**.

When you use the **NewFromSolutionWithData** method, the new form opens in InfoPath and is ready to be filled out.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **NewFromSolutionWithData** method of the **XDocuments** collection is passed the URIs of an existing XML document, and an existing [form template](#), and a new form is created and its associated **XDocument** object is returned:

```
var objXDoc = Application.XDocuments.NewFromSolutionWithDat
```

In the following example, the **NewFromSolutionWithData** method of the **XDocuments** collection is passed an **XMLDOMNode** as initial data and the URI of an existing form, and a new form is created and its associated **XDocument** object is returned:

```
var objXMLNode = XDocument.DOM.selectSingleNode("/employees
```

```
var objXDoc = Application.XDocuments.NewFromSolutionWithDat
```

Now Method

Returns a **Variant** that represents the current system date and time in ISO format (yyyy-mm-ddThh:mm:ss).

```
expression.Now() As Variant
```

expression Required. An expression that returns a reference to a [Date](#) object.

returns A **Variant** that represents the current system date and time.

Security Level

0: Can be accessed without restrictions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the variable `currentDateTime` is set to the current system date and time:

```
var currentDateTime = XDocument.Util.Date.Now();
```

Nz Method

Returns a **Variant** that is identical to the node set passed to the function, except empty values in the node list are replaced with zeros (0).

```
expression.Nz(ByVal pxmllistInput As IXMLDOMNodeList) As Variant
```

expression Required. An expression that returns a reference to the **Math** object.

pxmllistInput Required. The node set that will have its empty values replaced with zeroes.

returns A **Variant** that represents a node set.

Security Level

0: Can be accessed without restrictions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the variable `noZero` is set to a node set that is identical to the `my:ages` node set, with all blank values replaced by zeros:

```
var ages = XDocument.DOM.selectNodes("//my:ages");  
var noZero = XDocument.Util.Math.Nz(ages);
```


Open Method (ExternalApplication Object)

Opens the specified Microsoft Office InfoPath 2003 form.

```
expression.Open(ByVal bstrDocumentURI As String, [ByVal  
dwBehavior As Long = 1])
```

expression Required. An expression that returns a reference to the [ExternalApplication](#) object.

bstrDocumentURI Required **String**. The string value that specifies the [Uniform Resource Identifier \(URI\)](#) of a form.

dwBehavior Optional **Long**. Default value is 1. A long value that specifies how the form should be opened. The values are based on the [XdDocumentVersionMode](#) enumeration.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

The **Open** method can be used only to open a form; it cannot be used to open a [form template](#). To create a form from a form template, use the [NewFromSolution](#) method of the **ExternalApplication** object. To create a form based on an existing form, use the [New](#) method of the **ExternalApplication** object.

Example

In the following example, which is written in the [Visual Basic for Applications \(VBA\)](#) programming language, the **Open** method of the **ExternalApplication** object is used to open a specified form:

```
Public Sub AutomateInfoPathForm()

    Dim objIP As Object

    'Create a reference to the ExternalApplication object.
    Set objIP = CreateObject("InfoPath.ExternalApplication")

    'Open an InfoPath form.
    objIP.Open ("C:\My Forms\Form1.xml")
    MsgBox ("The InfoPath form has been opened.")

    'Close the InfoPath form.
    objIP.Close ("C:\My Forms\Form1.xml")
    MsgBox ("The InfoPath form has been closed.")

    'Quit the InfoPath application.
    objIP.Quit
    MsgBox ("The InfoPath application has been closed.")

    Set objIP = Nothing

End Sub
```

Open Method (Index)

The **Open** method opens a form. This method is implemented in a number of Microsoft Office InfoPath 2003 object model collections and objects. Click an **Open** method link below to view the Help topic for a specific implementation of the **Open** method.

[Open](#) method as it applies to the [ExternalApplication](#) object.

[Open](#) method as it applies to the [XDocuments](#) collection.

Open Method (XDocuments Collection)

Opens the specified Microsoft Office InfoPath 2003 form.

```
expression.Open(ByVal varURI As Variant, [ByVal dwBehavior As Long = 1]) As XDocument
```

expression Required. An expression that returns a reference to the [XDocuments](#) collection.

varURI Required **Variant**. Specifies the [Uniform Resource Identifier \(URI\)](#) of a form.

dwBehavior Optional **Long**. Default value is 1. A long value that specifies how the form should be opened. The values are based on the [XdDocumentVersionMode](#) enumeration.

returns A reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Open** method can only be used to open a form; it cannot be used to open a [form template](#). To create a form from a form template, use the [NewFromSolution](#) method of the **XDocuments** collection. To create a form based on an existing form, use the [New](#) method of the **XDocuments** collection.

When you use the **Open** method, the specified form opens in InfoPath and is ready to be filled out.

Note If you use the optional *dwBehavior* argument in the **Open** method, you can only pass the numerical value of the **XdDocumentVersionMode** enumeration. Because InfoPath uses scripting languages for working with the object model, named values cannot be used.

Example

In the following example, the **Open** method of the **XDocuments** collection is passed the URI of an existing form, and the form is opened and its associated **XDocument** object returned:

```
var objXDoc;
```

```
objXDoc = Application.XDocuments.Open("C:\\MyForm.xml");
```

PerformSaveOperation Method

Performs the save operation requested by the user.

expression.**PerformSaveOperation()**

expression Required. An expression that returns a reference to a [SaveEvent](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **PerformSaveOperation** method performs a "save" or "save as" operation depending on the value of the [IsSaveAs](#) property of the **SaveEvent** object. If the operation is successful, the [IsDirty](#) property of the [XDocument](#) object is set to **False** and the **PerformSaveOperation** method returns **True**.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **PerformSaveOperation** method of the **SaveEvent** object is used to save the form using the internal definition of save.

```
eventObj.PerformSaveOperation();
```

PrintOut Method

Prints the form content as it is rendered in the window corresponding to the form's active view.

expression.**PrintOut**()

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

The **PrintOut** method uses the current printer settings. Returns a permission denied error if called from a form that is not fully trusted.

It is recommended that you don't call the **PrintOut** method from an [OnLoad](#) or [OnSwitchView](#) event procedure. You may encounter one or more of the following behaviors when calling the **PrintOut** method from an **OnLoad** or **OnSwitchView** event procedure:

The view may print correctly.

A blank document may be printed.

The incorrect view may be printed.

The following exception may occur:

Invalid context for the OM call.

Example

In the following example, the **PrintOut** method of the **XDocument** object is used to print the current view:

```
XDocument.PrintOut();
```

Query Method (ADOAdapter Object)

Reads data from the associated data adapter.

```
expression.Query()
```

expression Required. An expression that returns a reference to an [ADOAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Query** method fails if the [QueryAllowed](#) property of the **ADOAdapter** object is **False**.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Query Method (DataObject Object)

Reads data from the data adapter that is associated with the [DataObject](#) object and repopulates the **DataObject** object's associated XML Document Object Model (DOM).

```
expression.Query()
```

expression Required. An expression that returns a reference to the **DataObject** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Query** method can be used to refresh the data contained in the XML DOM that is associated with a **DataObject** object.

Example

In the following example, the **Query** method of the **DataObject** object is used to refresh the data contained in the XML DOM associated with the **DataObject** object:

```
XDocument.DataObjects("CityList").Query();
```

After you have refreshed the data in the **DataObject** object, you can call the **ForceUpdate** method of the **View** object to synchronize the data contained in the **DataObject** object and the view:

```
XDocument.View.ForceUpdate();
```

Query Method (DAVAdapter Object)

The **Query** method is available for the [DAVAdapter](#) object but, because the **DAVAdapter** object is available for submitting data only, the method will always generate a run-time error when it is called on that object.

```
expression.Query()
```

expression Required. An expression that returns a reference to the [DAVAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Query Method (EmailAdapter Object)

The **Query** method is available for the [EmailAdapter](#) object but, because the **EmailAdapter** object is available for submitting data only, the method will always generate a run-time error when it is called on that object.

```
expression.Query()
```

expression Required. An expression that returns a reference to the [EmailAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Query Method (HWSAdapter Object)

The **Query** method is available for the [HWSAdapter](#) object but, because the **HWSAdapter** object is available for submitting data only, the method will always generate a run-time error when it is called on that object.

```
expression.Query()
```

expression Required. An expression that returns a reference to the **HWSAdapter** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Query Method (Index)

The **Query** method retrieves data from the associated data adapter and stores it in an XML [Document Object Model \(DOM\)](#). This method is implemented in a number of Microsoft Office InfoPath 2003 object model objects. Click a **Query** method link below to view the Help topic for a specific implementation of the **Query** method.

[Query](#) method as it applies to the [ADOAdapter](#) object.

[Query](#) method as it applies to the [DataObject](#) object.

[Query](#) method as it applies to the [DAVAdapter](#) object.

[Query](#) method as it applies to the [HWSAdapter](#) object.

[Query](#) method as it applies to the [EmailAdapter](#) object.

[Query](#) method as it applies to the [SharepointListAdapter](#) object.

[Query](#) method as it applies to the [WebServiceAdapter](#) object.

[Query](#) method as it applies to the [XDocument](#) object.

[Query](#) method as it applies to the [XMLFileAdapter](#) object.

Query Method (SharePointListAdapter Object)

Reads data from the associated data adapter.

```
expression.Query()
```

expression Required. An expression that returns a reference to a [SharepointListAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Query** method fails if the [QueryAllowed](#) property of the **SharePointListAdapter** object is **False**.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Query Method (WebServiceAdapter Object)

Reads data from the associated data adapter.

```
expression.Query()
```

expression Required. An expression that returns a reference to a [WebServiceAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Query** method fails if the [QueryAllowed](#) property of the **WebServiceAdapter** object is **False**.

Note An ADO.Net DataSet cannot be used as a query parameter for the **WebServiceAdapter** object or any other [data adapter](#).

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Query Method (XDocument Object)

Retrieves data from a form's associated [data adapter](#) object and stores the data in the form's underlying XML [Document Object Model \(DOM\)](#).

```
expression.Query()
```

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Query** method will return an error if the form does not have an associated data adapter object.

Example

In the following example, the **Query** method of the **XDocument** object is used to query the associated data adapter:

```
XDocument.Query();
```

Query Method (XMLFileAdapter Object)

Reads data from the associated data adapter.

```
expression.Query()
```

expression Required. An expression that returns a reference to an [XMLFileAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Query** method fails if the [QueryAllowed](#) property of the **XMLFileAdapter** object is **False**.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Quit Method (Application Object)

Quits the Microsoft Office InfoPath 2003 application.

```
expression.Quit([ByVal bForce As Boolean = False])
```

expression Required. An expression that returns a reference to the [Application](#) object.

bForce Optional **Boolean**. Default value is **False**. Determines whether open forms will be saved during the quit operation. If set to **False**, all forms will be closed without saving, even if the data in the forms has been changed. If set to **True**, the user will be prompted to save the forms.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

If the **Quit** method is used in a form that is not fully trusted, the method will return a "permission denied" error.

Example

In the following example, the **Quit** method of the **Application** object is used to quit InfoPath without saving any of the currently open forms:

```
Application.Quit(false);
```


Quit Method (ExternalApplication Object)

Quits the Microsoft Office InfoPath 2003 application.

expression.Quit

expression Required. An expression that returns a reference to the [ExternalApplication](#) object.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

If you use the [Close](#) method of the **ExternalApplication** object before using the **Quit** method, data that has been changed in the form will not be saved, nor will users be prompted to save it. However, if you do not use the **Close** method but only use the **Quit** method, users will be prompted to save the form before quitting the InfoPath application.

Example

In the following example, which is written in the [Visual Basic for Applications \(VBA\)](#) programming language, the **Quit** method of the **ExternalApplication** object is used to quit the InfoPath application:

```
Public Sub AutomateInfoPathForm()

    Dim objIP As Object

    'Create a reference to the ExternalApplication object.
    Set objIP = CreateObject("InfoPath.ExternalApplication")

    'Open an InfoPath form.
    objIP.Open ("C:\My Forms\Form1.xml")
    MsgBox ("The InfoPath form has been opened.")

    'Close the InfoPath form.
    objIP.Close ("C:\My Forms\Form1.xml")
    MsgBox ("The InfoPath form has been closed.")

    'Quit the InfoPath application.
    objIP.Quit
    MsgBox ("The InfoPath application has been closed.")

    Set objIP = Nothing

End Sub
```

Quit Method (Index)

The **Quit** method quits the application. This method is implemented in a number of Microsoft Office InfoPath 2003 object model objects. Click a **Quit** method link below to view the Help topic for a specific implementation of the **Quit** method.

[Quit](#) method as it applies to the [Application](#) object.

[Quit](#) method as it applies to the [ExternalApplication](#) object.

RegisterSolution Method (Application Object)

Installs the specified Microsoft Office InfoPath form template.

```
expression.RegisterSolution(ByVal bstrSolutionURL As String, [ByVal bstrBehavior As String = "overwrite"])
```

expression Required. An expression that returns a reference to an [Application](#) object.

bstrSolutionURL Required **String**. The string that specifies the [Uniform Resource Locator \(URL\)](#) of the form template. This parameter can be specified as a [form definition \(.xsf\) file](#) or a form template (.xsn) file.

bstrBehavior Optional **String**. Default value is "overwrite". The string that specifies how the form template is to be installed. The only other valid value for this parameter is "new-only".

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

If the form template has already been registered, and "new-only" is specified for the *bstrBehavior* parameter, the **RegisterSolution** method will return an error. If "overwrite" is specified, the form template's registration record will be overwritten.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following [Visual Basic for Applications \(VBA\)](#) example, the **RegisterSolution** method of the **Application** object is used to install a form template:

```
Public Sub InstallForm()  
  
    Dim objIP As Object  
  
    ' Create a reference to the Application object.  
    Set objIP = CreateObject("InfoPath.Application")  
  
    ' Register the InfoPath form template.  
    objIP.RegisterSolution ("C:\\My Forms\\MyFormTemplate.xsn")  
    MsgBox "The InfoPath form template has been registered."  
  
    Set objIP = Nothing  
  
End Sub
```


RegisterSolution Method (ExternalApplication Object)

Installs the specified Microsoft Office InfoPath 2003 form template.

```
expression.RegisterSolution(ByVal bstrSolutionURL As String, [ByVal bstrBehavior As String = "overwrite"])
```

expression Required. An expression that returns a reference to the [ExternalApplication](#) object.

bstrSolutionURL Required **String**. The string value that specifies the [Uniform Resource Locator \(URL\)](#) of the form template. This parameter can be specified as a [form definition \(.xsf\) file](#) or a form template (.xsn) file.

bstrBehavior Optional **String**. Default value is **overwrite**. The string value that specifies how the form template is to be installed. The only other valid value for this parameter is **new-only**.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

If the form template has already been registered, and the **new-only** value is used for the ***bstrBehavior*** parameter, the **RegisterSolution** method will return an error. If the **overwrite** value is used, the form template's registration record will be overwritten.

Example

In the following example, which is written in the [Visual Basic for Applications \(VBA\)](#) programming language, the **RegisterSolution** method of the **ExternalApplication** object is used to install a form template:

```
Public Sub InstallForm()  
  
    Dim objIP As Object  
  
    'Create a reference to the ExternalApplication object.  
    Set objIP = CreateObject("InfoPath.ExternalApplication")  
  
    'Register the InfoPath form template.  
    objIP.RegisterSolution ("C:\\My Forms\\MyFormTemplate.xsn")  
    MsgBox ("The InfoPath form template has been registered.")  
  
    Set objIP = Nothing  
  
End Sub
```


RegisterSolution Method (Index)

The **RegisterSolution** method installs the specified Microsoft Office InfoPath 2003 [form template](#). This method is implemented in several InfoPath object model collections. Click a **RegisterSolution** method link below to view the Help topic for a specific implementation of the **RegisterSolution** method.

[RegisterSolution](#) method as it applies to the [Application](#) object.

[RegisterSolution](#) method as it applies to the [ExternalApplication](#) object.

ReportError Method

Creates an [Error](#) object and adds it to the [Errors](#) collection.

```
expression.ReportError(ByVal varNode As Variant, ByVal bstrShortErrorMessage As String, ByVal fSiteIndependent As Boolean, [ByVal bstrDetailedErrorMessage As String], [ByVal !ErrorCode As Long], [ByVal bstrType As String = "modeless"]) As Error
```

expression Required. Returns a reference to the [DataDOMEvent](#) object.

varNode Required **Variant**. The XML [Document Object Model \(DOM\)](#) node that the error is associated with.

bstrShortErrorMessage Required **String**. The text to be used for the short error message.

fSiteIndependent Optional **Boolean**. Sets the condition for automatic deletion of the **Error** object. If **True**, the **Error** object will be deleted on change for any nodes that matched the XPath expression corresponding to the **Error** object. If **False**, the **Error** object will be deleted when the node returned by the **Site** property of a given event object has been changed.

bstrDetailedErrorMessage Optional **String**. The text to be used for the detailed error message.

!ErrorCode Optional **Long**. The number to be used as the error code.

bstrType Optional **String**. Default value is "modeless". Determines whether the change in value will be automatically rejected or whether the user will be prompted to accept or reject the change. The other value is "modal".

returns A reference to an **Error** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

When the **ReportError** method of the **DataDOMEvent** object is called, Microsoft Office InfoPath 2003 creates an **Error** object and adds it to the **Errors** collection. Errors are removed from the collection when the validation constraint is no longer invalid, or when they are explicitly removed using the [Delete](#) or [DeleteAll](#) methods of the **Errors** collection.

Errors can also be created using the [Add](#) method of the **Errors** collection.

Note Site-independent errors should be used when you want the errors to apply to all XML DOM nodes of the same type. If you want the error to apply to a specific XML DOM node, use site-dependent errors.

Example

In the following example from the Data Validation developer sample form, the **ReportError** method of the **DataDOMEvent** object is used to create a custom error and add it to the errors collection:

```
function msoxd__total::OnValidate(eventObj)
{
    if (parseInt(eventObj.Site.nodeTypeValue, 10) > 75)
        eventObj.ReportError(eventObj.Site, "The total is too high. " +
            "The total number of blocks cannot be greater than 75.", false);

    if (parseInt(eventObj.Site.nodeTypeValue, 10) < 0)
        eventObj.ReportError(eventObj.Site, "The total is too low. " +
            "The total number of blocks cannot be less than 0.", false);
}
```


Save Method

Saves the form to the [Uniform Resource Locator \(URL\)](#) that it is currently associated with.

```
expression.Save()
```

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

The **Save** method will return an error if called from a form that is not fully trusted.

Example

In the following example, the **Save** method of the **XDocument** object is used to save a form:

```
XDocument.Save();
```


SaveAs Method

Saves the form to the specified [Uniform Resource Locator \(URL\)](#).

```
expression.SaveAs(ByVal bstrFileUrl As String)
```

expression Required. An expression that returns a reference to an [XDocument](#) object.

bstrFileUrl Required **String**. The URL address that the form should be saved to.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

The **SaveAs** method will return an error if called from a form that is not fully trusted.

Note The URL that the form is saved to must be in the same domain as the form that calls the **SaveAs** method.

Example

In the following example, the **SaveAs** method of the **XDocument** object is used to save a form:

```
XDocument.SaveAs("C:\MyForm.xml");
```

SaveState Method

Defines a method that InfoPath calls when it needs to save the state of an instance of the control in a view.

expression.**SaveState()**

expression Required. An expression that returns a reference to the [InfoPathControl](#) object.

Remarks

The **InfoPathControl** and [InfoPathControlSite](#) objects and their methods and properties are designed to be used only from the implementation of an ActiveX control. These objects and their members are not supported in InfoPath form code. For more information on how to create ActiveX controls that work with InfoPath, see the [InfoPath Developer Center](#).

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form template that contains a view with an ActiveX control that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

SelectNodes Method

Selects a range of nodes in a view based on the specified starting XML Document Object Model (DOM) node, the ending XML DOM node, and the view context.

```
expression.SelectNodes(ByRef pxnStartNode As XMLDOMNode,  
[ByVal varEndNode As Variant], [ByVal varViewContext As Variant])
```

expression Required. An expression that returns a reference to the [View](#) object.

pxnStartNode Required **XMLDOMNode**. The XML DOM node that begins the range.

varEndNode Optional **Variant**. The XML DOM node that ends the range. If not specified, only the starting XML DOM node will be used.

varViewContext Optional **Variant**. The ID of the control that is used for the context, which is an element with the specified view context of `xd:CtrlId`.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If a view context is specified, all of the XML DOM nodes that are to be selected must be within that context.

If any of the arguments to the **SelectNodes** method are null or are not exposed in the view, the **SelectNodes** method will return an error. In addition, if there are more than one set of view elements which map to the same specified XML DOM nodes, within the specified view context, then the **SelectNodes** method will also return an error.

Example

In the following example, the **SelectNodes** method of the **View** object is used to set selection on a single item in the view, corresponding to the specified XML DOM node and then, using the [GetSelectedNodes](#) method of the **View** object, determines whether the selection has been successful by displaying information about the XML DOM node in a message box:

```
function SelectEmployee()
{
    var objXMLNodes;
    var objXMLNode;

    objXMLNode = XDocument.DOM.selectSingleNode("/employees/e
XDocument.View.SelectNodes(objXMLNode);

    objXMLNodes = XDocument.View.GetSelectedNodes();
    if (objXMLNodes.Count > 0)
    {
        XDocument.UI.Alert(objXMLNodes(0).nodeName + "\n\n" + obj>
    }
}
```


SelectText Method

Selects the text contained in an editable field that is bound to the specified XML [Document Object Model \(DOM\)](#) node.

```
expression.SelectText(ByRef pxnField As XMLDOMNode, [ByVal varViewContext As Variant])
```

expression Required. An expression that returns a reference to the [View](#) object.

pxnField Required **XMLDOMNode**. The XML DOM node.

varViewContext Optional **Variant**. The ID of the control that is used for the context, which is an element with the specified view context of `xd:CtrlId`.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If a view context is specified, then the editable field that is to be selected must be within that context.

If there are more than one set of view elements which map to the same specified XML DOM node, within the given view context, then the **SelectText** method will return an error. In addition, if any of the arguments to the **SelectText** method are null or are not exposed in the view, then the **SelectText** method will also return an error.

Example

In the following example, the **SelectText** method of the **View** object is used to select a field that is bound to an XML DOM node:

```
var objXMLNode;
```

```
objXMLNode = XDocument.DOM.selectSingleNode("/employees/em  
XDocument.View.SelectText(objXMLNode);
```


SetDataVariable Method

Sets the value of a predefined variable stored as a [processing instruction](#) attribute in the form's underlying XML document.

```
expression.SetDataVariable(ByVal IVariableNumber As Long, ByVal bstrVariableValue As String)
```

expression Required. An expression that returns a reference to an [XDocument](#) object.

IVariableNumber Required **Long**. The number of the variable.

bstrVariableValue Required **String**. The value of the variable.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the variable being set is not a valid processing instruction attribute, the **SetDataVariable** method will return an error.

To get the value of a variable, use the [GetDataVariable](#) method of the **XDocument** object.

Note Microsoft Office InfoPath 2003 only supports using the initialView variable, which is the variable used to specify the initial view displayed when a form is opened. The number of this variable is always 1, and its value must be the name of a view within the form.

Example

In the following example, the **SetDataVariable** method of the **XDocument** object is used to set the value of the first variable:

```
XDocument.SetDataVariable(1, "View 2");
```

SetDirty Method

Sets the **IsDirty** property on the **XDocument** object to a **Boolean** value that indicates whether the data in a Microsoft Office InfoPath 2003 form has been modified since it was last saved.

```
expression.SetDirty(ByVal vflsDirty As Boolean)
```

expression Required. An expression that returns a reference to an **XDocument** object.

vflsDirty Required **Boolean**. Specifies whether the form is to be marked as unmodified or not.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **SetDirty** method can be used from the [OnSubmitRequest](#) event handler to force a document to be marked as unchanged. InfoPath will therefore not request the user to save the form when it is closed.

The **SetDirty** method can also be used from the [OnSaveRequest](#) event handler to programmatically mark the form as changed or unchanged since it was last saved.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **IsDirty** property of the current form is set to **False** so that InfoPath will not prompt the user to save the form when it is closed.

```
XDocument.SetDirty(false);
```

SetNamedNodeProperty Method

Sets the value of a named property for the supplied XML node, which must be a nonattribute node in the main data source.

```
expression.SetNamedNodeProperty(ByVal pxmlMainDOMNode As  
IXMLDOMNode, ByVal bstrPropertyName As String, ByVal bstrValue  
As String)
```

expression Required. An expression that returns a reference to an [XDocument](#) object.

pxmlMainDOMNode Required **IXMLDOMNode**. An XML node corresponding to a nonattribute node in the main data source, for which a named property is to be set.

bstrPropertyName Required **String**. Specifies the name of the property being set.

bstrValue Required **String**. Specifies the value to which the property will be set.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Named properties allow users to associate strings with user-defined properties of XML element nodes in the main data source. The value of a named property can be set by using the **SetNamedNodeProperty** method. Use the [GetNamedNodeProperty](#) method to read the value of a named property.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the value of a named property (with the name "cost") of an XML node (called "item") is set by using the **SetNamedNodeProperty** method:

```
var objXMLNode = XDocument.DOM.selectSingleNode("/items/item")
XDocument.SetNamedNodeProperty(objXMLNode, 'cost', '100');
var strTest = XDocument.GetNamedNodeProperty(myNode, 'cost', 'err');
```

SetSaveAsDialogFileName Method

Sets the default file name for a form in the **Save As** dialog box.

```
expression.SetSaveAsDialogFileName(ByVal strFileName As String)
```

expression Required. An expression that returns a reference to a [UI](#) object.

strFileName Required. The file name of the form supplied to the **Save As** dialog box.

Security

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If the **strFileName** argument is **null**, "Form" is used as the file name. The **SetSaveAsDialogFileName** method may be used in conjunction with the [SetSaveAsDialogLocation](#) method.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **SetSaveAsDialogFileName** method of the **UI** object is used to set the default file name for the **Save As** dialog box:

```
XDocument.UI.SetSaveAsDialogFileName("Status Report.xml");
```

SetSaveAsDialogLocation Method

Sets the initial location at which the **Save As** dialog starts to browse when it is opened.

expression.**SetSaveAsDialogLocation**(ByVal *strLocationUrl* As String)

expression Required. An expression that returns a reference to the [UI](#) object.

strLocationUrl Required. The location, expressed as a URL, at which the **Save As** dialog box starts browsing.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The location specified must be an absolute path and it should not include a file name. InfoPath handles invalid paths, however, and no error message is generated if an invalid path is specified for the **strLocationUrl** argument. The **SetSaveAsDialogLocation** method may be used in conjunction with the [SetSaveAsDialogFileName](#) method.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **SetSaveAsDialogLocation** method of the **UI** object is used to set the initial location for the **Save As** dialog:

```
XDocument.UI.SetSaveAsDialogLocation("\\\\MyShare\\Forms");
```

ShowMailItem Method

Creates an e-mail message in the default e-mail editor and attaches the currently open Microsoft Office InfoPath 2003 form to the message.

```
expression.ShowMailItem(ByVal bstrTo As String, ByVal bstrCC As String, ByVal bstrBCC As String, ByVal bstrSubject As String, ByVal bstrBody As String)
```

expression Required. An expression that returns a reference to the [UI](#) object.

bstrTo Required **String**. The e-mail address to send the e-mail message to.

bstrCC Required **String**. The e-mail address to copy the e-mail message to.

bstrBCC Required **String**. The e-mail address to blind copy the e-mail message to.

bstrSubject Required **String**. The subject of the e-mail message.

bstrBody Required **String**. The body text of the e-mail message.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

After the e-mail message is created, the default e-mail editor will display the e-mail message; users can then inspect and edit the e-mail message before sending it. The **ShowMailItem** method will return an error if no e-mail editing program is available.

Note The **ShowMailItem** method does not send the e-mail messages it creates; users must manually send the e-mail messages.

You can also use the [MailEnvelope](#) object, accessed through the [MailEnvelope](#) property of the [Window](#) object, to programmatically create e-mail messages.

Example

In the following example, the **ShowMailItem** method of the **UI** object is used to create a custom e-mail message:

```
XDocument.UI.ShowMailItem("someone@example.com", "", "",  
    "Updated Form", "Here is the updated form that you requested.");
```


ShowModalDialog Method

Displays a custom modal dialog box in a Microsoft Office InfoPath 2003 form.

```
expression.ShowModalDialog(ByVal bstrName As String, [ByVal varArguments As Variant], [ByVal varHeight As Variant], [ByVal varWidth As Variant], [ByVal varTop As Variant], [ByVal varLeft As Variant]) As Variant
```

expression Required. An expression that returns a reference to the [UI](#) object.

bstrName Required **String**. The name of the [.html](#) file used for the modal dialog box.

varArguments Optional **Variant**. Specifies the arguments to use when displaying the modal dialog box. Can be any type of value, including an array of values.

varHeight Optional **Variant**. Sets the height of the modal dialog box.

varWidth Optional **Variant**. Sets the width of the modal dialog box.

varTop Optional **Variant**. Sets the top position of the modal dialog box relative to the upper left corner of the desktop.

varLeft Optional **Variant**. Sets the left position of the modal dialog box relative to the upper left corner of the desktop.

returns **Variant**.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

The **ShowModalDialog** method of the **UI** object allows you to display custom dialog boxes to users as they fill out a fully trusted form. Custom dialog boxes are implemented as .html files created in any type of HTML editor, such as Microsoft FrontPage. You can use scripting code in a custom dialog box that interacts with the InfoPath object model if you pass objects to the custom dialog box using the ***varArguments*** parameter.

To use a custom dialog box in an InfoPath form, you must first add the .html file of the custom dialog box to the form's set of resource files by using the **Resource Manager** dialog box. The **Resource Manager** dialog box is available from the **Tools** menu in [design mode](#). After you have added the custom dialog box file to the form, you can use the **ShowModalDialog** method to display it.

Note Although the **ShowModalDialog** method can only be used in fully trusted forms, you can create a custom dialog box in standard forms using the **showModalDialog** method of the [Dynamic HTML \(DHTML\)](#) object model.

Example

In the following example, the **ShowModalDialog** method of the **UI** object is used to display a custom dialog box. Note that the [XDocument](#) object is passed to the custom dialog box using the **varArguments** parameter.

```
XDocument.UI.ShowModalDialog("SimpleDialog.htm", XDocument
```

The following example is the HTML code used to implement a simple custom dialog box. Note the use of the **dialogArguments** property of the DHTML **window** object to get the values passed to the custom dialog box, which in this case is the **XDocument** object of the InfoPath object model, from the **ShowModalDialog** method. When a user clicks the **Show Alert** button in the custom dialog box, the source XML of the form's underlying XML document appears in a message box.

```
<html>
  <head>
    <script language="jscript">
      var gobjXDocument = null;

      function Initialize()
      {
        // Save a reference to the XDocument object.
        if (typeof window.dialogArguments == "object")
          gobjXDocument = window.dialogArguments;
      }
    </script>

    <title>A Simple Custom Dialog Box</title>
  </head>

  <body style="BACKGROUND-COLOR: window" onLoad="Initiali
    <strong>Click one of the following buttons:</strong>
```

```
<br/>
<br/>
<div id="divButtons" tyle="align:center">
  <input id="btnShowAlert" style="WIDTH: 106px; HEIGHT: 24px;
    onclick='gobjXDocument.UI.Alert(gobjXDocument.DOM.xml
    type="button" size="21" value="Show Alert"></input>
  <input id="btnCancel" style="WIDTH: 106px; HEIGHT: 24px"
    onclick="window.close();" type="button" size="21"
    value="Cancel"></input>
</div>
</body>
</html>
```

ShowSignatureDialog Method

Displays the Microsoft Office InfoPath 2003 **Digital Signatures** dialog box.

expression.**ShowSignatureDialog()**

expression Required. An expression that returns a reference to the [UI](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **ShowSignatureDialog** method can be used only in forms that have been enabled for digital signing. The method will return an error if used in a form that is not enabled for digital signing.

Example

In the following example, the **ShowSignatureDialog** method of the **UI** object is used to display the InfoPath **Digital Signatures** dialog box:

```
XDocument.UI.ShowSignatureDialog();
```

Sign Method (Index)

This method is implemented in several Microsoft Office InfoPath 2003 object model collections. Click a **Sign** method link below to view the Help topic for a specific implementation of the **Sign** method.

[Sign](#) method as it applies to the [Signature](#) object.

[Sign](#) method as it applies to the [SignedDataBlock](#) object.

Sign Method (Signature Object)

Writes the XML digital signature block and computes the cryptographic hash for the signed data. This method can only be called from the [OnSign](#) event handler. Calling this method displays the **Digital Signatures** dialog box.

```
expression.Sign()
```

expression Required. An expression that returns a reference to a [Signature](#) object.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

Use the [Create](#) method of the [Signatures](#) collection to create a digital signature. Then, use the **Sign** method to write the digital signature.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Sign Method (SignedDataBlock Object)

Invokes the **Digital Signatures** dialog box to add a digital signature to a set of signed data in a Microsoft Office InfoPath 2003 [form](#). The new signature uses the default signature template, and it is applied to the [SignedDataBlock](#) object.

```
expression.Sign()
```

expression Required. An expression that returns a reference to a **SignedDataBlock** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Submit Method (ADOAdapter Object)

Executes the submit operation on the associated adapter.

expression.**Submit()**

expression Required. An expression that returns a reference to a [ADOAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Fails if the [SubmitAllowed](#) property of the **ADOAdapter** object is **False**.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Submit Method (DAVAdapter Object)

Executes the submit operation on the associated adapter.

```
expression.Submit()
```

expression Required. An expression that returns a reference to a [DAVAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Submit** method of the **DAVAdapter** object submits the form's entire underlying XML document, whereas the [SubmitData](#) method of the **DAVAdapter** object can submit any XML DOM element or document.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Submit Method (EmailAdapter Object)

Executes the submit operation on the associated adapter.

```
expression.Submit()
```

expression Required. An expression that returns a reference to an [EmailAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

The **Submit** method of the **EmailAdapter** object submits the form's entire underlying XML document, whereas the [SubmitData](#) method of the **EmailAdapter** object can submit any XML DOM element or document.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Submit Method (HWSAdapter Object)

Executes the submit operation for the associated adapter.

expression.**Submit()**

expression Required. An expression that returns a reference to an [HWSAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Submit Method (Index)

The **Submit** method executes the submit operation on the associated [data adapter](#). This method is implemented in several Microsoft Office InfoPath 2003 object model collections. Click a **Submit** method link below to view the Help topic for a specific implementation of the **Submit** method.

[Submit](#) method as it applies to the [ADOAdapter](#) object.

[Submit](#) method as it applies to the [DAVAdapter](#) object.

[Submit](#) method as it applies to the [EmailAdapter](#) object.

[Submit](#) method as it applies to the [HWSAdapter](#) object.

[Submit](#) method as it applies to the [SharepointListAdapter](#) object.

[Submit](#) method as it applies to the [WebServiceAdapter](#) object.

[Submit](#) method as it applies to the [XDocument](#) object.

[Submit](#) method as it applies to the [XMLFileAdapter](#) object.

Submit Method (SharePointListAdapter Object)

The **Submit** method is available for the [SharepointListAdapter](#) object but, because the **SharePointListAdapter** object is available for receiving data only, the method will always generate a run-time error when it is called on that object.

```
expression.Submit()
```

expression Required. An expression that returns a reference to a [SharepointListAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Submit Method (WebServiceAdapter Object)

Executes the submit operation on the associated adapter.

```
expression.Submit()
```

expression Required. An expression that returns a reference to a [WebServiceAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Fails if the [SubmitAllowed](#) property of the **WebServiceAdapter** object is **False**.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Submit Method (XDocument Object)

Executes the predefined submit operation in a Microsoft Office InfoPath 2003 form.

expression.**Submit**()

expression Required. An expression that returns a reference to an [XDocument](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Using the **Submit** method is similar to, but not exactly like, using the submit operation from the InfoPath user interface. Calling the **Submit** method simply performs the submit operation, but the submit operation must first be enabled for a form in [design mode](#).

Note You can write a custom submit operation using the [OnSubmitRequest](#) event, and then you can programmatically call the event using the **Submit** method.

Example

In the following example, the **Submit** method of the **XDocument** object is used to submit the form using the predefined submit operation:

```
XDocument.Submit();
```

Submit Method (XMLFileAdapter Object)

The **Submit** method is available for the [XMLFileAdapter](#) object but, because the **XMLFileAdapter** object is available for receiving data only, the method will always generate a run-time error when it is called on that object.

```
expression.Submit()
```

expression Required. An expression that returns a reference to a [XMLFileAdapter](#) object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

SubmitData Method (DavAdapter Object)

Submits the specified DOM element or DOM document to a [data adapter](#).

expression.**SubmitData**(ByVal *pData* As IXMLDOMNode)

expression Required. An expression that returns a reference to a [DAVAdapter](#) object.

pData Required **IXMLDOMNode**. The XML data that is to be merged into the currently open form.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

SubmitData Method (EmailAdapter Object)

Submits the specified DOM element or DOM document to a [data adapter](#).

```
expression.SubmitData(ByVal pData As IXMLDOMNode)
```

expression Required. An expression that returns a reference to an [EmailAdapter](#) object.

pData Required **IXMLDOMNode**. The XML data that is to be merged into the currently open form.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

SubmitData Method (Index)

The **SubmitData** method executes the submit operation for the associated [data adapter](#). This method is implemented in several Microsoft Office InfoPath 2003 object model collections. Click a **SubmitData** method link below to view the Help topic for a specific implementation of the **SubmitData** method.

[SubmitData](#) method as it applies to the [DAVAdapter](#) object.

[SubmitData](#) method as it applies to the [EmailAdapter](#) object.

SwitchView Method

Changes the active view of a Microsoft Office InfoPath 2003 form to a specified view.

```
expression.SwitchView(ByVal bstrName As String)
```

expression Required. An expression that returns a reference to the [View](#) object.

bstrName Required **String**. The name of the view to switch to. If an empty string is used, the default view is displayed.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

If an empty string is used as the ***bstrName*** parameter, the view is changed to the default view of the form.

Example

In the following example, the **SwitchView** method of the **View** object is used to change the current view to the default view:

```
XDocument.View.SwitchView("");
```

Today Method

Returns a **Variant** containing the current system date in ISO format (yyyy-mm-dd).

expression.**Today**() As Variant

expression Required. An expression that returns a reference to the **Date** object.

returns A **Variant** containing the current system date.

Security Level

0: Can be accessed without restrictions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the variable `currentDate` is set to the current system date.

```
var currentDate = XDocument.Util.Date.Today();
```

Uninit Method

Defines a method that must be provided by the developer to perform any clean up routines that are required before an instance of the control is removed from a view.

expression.**Uninit()**

expression Required. An expression that returns a reference to the [InfoPathControl](#) object.

Remarks

InfoPath calls the **Uninit** method immediately before the control is removed from the view and destroyed. Note that because InfoPath forms make use of XSL transforms to represent views, any changes in the data or explicit calls to the object model can cause InfoPath to automatically synchronize the view with the data that is contained in a form's underlying XML document, which destroys and recreates the view. This means that ActiveX controls are likely to be created and destroyed much more often within a given session than controls in Visual Basic forms or Web forms. ActiveX controls which need to preserve state information independent of the bound data, such as the state of the scroll position, should create routines in the **Init** method of the control that use the [SetNamedNodeProperty](#) method to save this information, and use the [GetNamedNodeProperty](#) method to restore state information during the **Init** method call.

The **InfoPathControl** and [InfoPathControlSite](#) objects and their methods and properties are designed to be used only from the implementation of an ActiveX control. These objects and their members are not supported in InfoPath form code. For more information on how to create ActiveX controls that work with InfoPath, see the [InfoPath Developer Center](#).

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form template that contains a view with an ActiveX control that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

UnregisterSolution Method (Application Object)

Uninstalls the specified Microsoft Office InfoPath 2003 form template.

expression.**UnregisterSolution**(ByVal *bstrSolutionURI* As String)

expression Required. An expression that returns a reference to an [Application](#) object.

bstrSolutionURI Required **String**. The string that specifies the [Uniform Resource Identifier \(URI\)](#) of the form template.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

If the form template cannot be unregistered, the **UnregisterSolution** method will return an error.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following [Visual Basic for Applications \(VBA\)](#) example, the **UnregisterSolution** method of the **Application** object is used to uninstall a form template:

```
Public Sub UninstallForm()
```

```
    Dim objIP As Object
```

```
    ' Create a reference to the ExternalApplication object.
```

```
    Set objIP = CreateObject("InfoPath.Application")
```

```
    ' Unregister the InfoPath form template.
```

```
    objIP.UnregisterSolution ("C:\\My Forms\\MyFormTemplate.xsn")
```

```
    MsgBox ("The InfoPath form template has been unregistered.")
```

```
    Set objIP = Nothing
```

```
End Sub
```


UnregisterSolution Method (ExternalApplication Object)

Uninstalls the specified Microsoft Office InfoPath 2003 form template.

expression.**UnregisterSolution**(ByVal *bstrSolutionURI* As String)

expression Required. An expression that returns a reference to the [ExternalApplication](#) object.

bstrSolutionURI Required **String**. The string value that specifies the [Uniform Resource Identifier \(URI\)](#) of the form template.

Security Level

3: Can be accessed only by fully trusted forms.

Remarks

If the form template cannot be unregistered, the **UnregisterSolution** method will return an error.

Example

In the following example, which is written in the [Visual Basic for Applications \(VBA\)](#) programming language, the **UnregisterSolution** method of the **ExternalApplication** object is used to uninstall a form template:

```
Public Sub UninstallForm()
```

```
    Dim objIP As Object
```

```
    'Create a reference to the ExternalApplication object.
```

```
    Set objIP = CreateObject("InfoPath.ExternalApplication")
```

```
    'Unregister the InfoPath form template.
```

```
    objIP.UnregisterSolution ("C:\\My Forms\\MyFormTemplate.xsn")
```

```
    MsgBox ("The InfoPath form template has been unregistered.")
```

```
    Set objIP = Nothing
```

```
End Sub
```


UnregisterSolution Method (Index)

The **UnregisterSolution** method uninstalls the specified Microsoft Office InfoPath 2003 [form template](#). This method is implemented in several Microsoft Office InfoPath 2003 object model collections. Click an **UnregisterSolution** method link below to view the Help topic for a specific implementation of the **UnregisterSolution** method.

[UnregisterSolution](#) method as it applies to the [Application](#) object.

[UnregisterSolution](#) method as it applies to the [ExternalApplication](#) object.

OnAfterChange Event

Occurs after changes to a form's underlying XML document have been accepted and after the [OnValidate](#) event has occurred.

```
Function node::OnAfterChange(ByRef pDataDOMEvent As  
DataDOMEvent)
```

pDataDOMEvent Required **DataDOMEvent**. A reference to the [DataDOMEvent](#) object.

Remarks

This [event handler](#) does not allow users to cancel an operation.

The **OnAfterChange** event is typically used for changing data in a form after other changes have occurred— for example, making calculations or changing the structure of a form's underlying XML document.

Note In some cases, events related to changes in a form's underlying XML document may occur more than once. For example, when existing data is changed, an insert and delete operation occurs.

Example

In the following example from the Events developer sample form, the `OnAfterChange` event handler is used to call a custom function that performs calculations:

```
function msoxd__ContactDates::OnAfterChange(eventObj)
{
    if (eventObj.IsUndoRedo)
    {
        // An undo or redo operation has occurred and the DOM is read-on
        return;
    }
    CalculateTotalCampaignCost();
}
```


OnAfterImport Event

Occurs after the import (or merge) operation has successfully completed.

Function *XDocument*::**OnAfterImport**(ByRef *pEvent* As DocEvent)

pEvent Required **DocEvent**. A reference to the [DocEvent](#) object.

Remarks

This [event handler](#) does not allow users to cancel an operation.

If the merge operation includes merging multiple forms, the **OnAfterImport** event occurs only after all forms have been merged and the complete operation is successful.

Note The OnAfterImport event handler cannot be created using Microsoft Office InfoPath 2003 [design mode](#); it must be created manually.

▶ [Creating the OnAfterImport event handler](#)

Example

In the following example, the `OnAfterImport` event handler is used to display a message box that informs the user that the merge operation completed successfully:

```
function XDocument::OnAfterImport(eventObj)
{
    XDocument.UI.Alert("Merge operation was successful.")
}
```


OnBeforeChange Event

Occurs after changes to a form's underlying XML document have been made but before the changes are accepted.

```
Function node::OnBeforeChange(ByRef pDataDOMEvent As  
DataDOMEvent)
```

pDataDOMEvent Required **DataDOMEvent**. A reference to the [DataDOMEvent](#) object.

Remarks

This [event handler](#) allows users to cancel an operation.

During the **OnBeforeChange** event, the form's underlying XML document is placed in read-only mode. If the [ReturnStatus](#) property of the **DataDOMEvent** object is set to False, Microsoft Office InfoPath 2003 rejects the changes that were made and a message box is displayed to the user. If an error occurs in the scripting code for the OnBeforeChange event handler, InfoPath rejects the changes and restores the data to its previous state.

Notes

It is best to avoid switching views during the **OnBeforeChange** event; changes have not yet been accepted, and switching to another view may result in an error.

In some cases, events related to changes in a form's underlying XML document may occur more than once. For example, when existing data is changed, an insert and delete operation occurs.

If a validation error is encountered in any **OnBeforeChange** event handler, the document fails to load. A try-catch block in the [OnLoad](#) event can be used to catch this validation failure and to load the document despite the error.

Example

In the following example from the Events developer sample form, the `OnBeforeChange` event handler is used to validate the data in a field. If the data is not valid, the **ReturnStatus** property of the **DataDOMEvent** object is used to reject the changes.

```
function msoxd__RepVisitDt::OnBeforeChange(eventObj)
{
    var oNode = XDocument.DOM.selectSingleNode
        ("/Customers/CustomerInfo/ContactDates/PhoneContactDt");

    if (!oNode.text)
    {
        eventObj.ReturnMessage = "The Phone Contact Start date must be
            set prior to the Representative Visit date.";
        eventObj.ReturnStatus = false;
        return;
    }

    // If the data is valid, eventObj.ReturnStatus = true.
    eventObj.ReturnStatus = true;
    return;
}
```


OnClick Event

Occurs when a button control is clicked within a view in a Microsoft Office InfoPath 2003 form.

Function *ScriptID::***OnClick**(ByRef *eventObj* As DocActionEvent)

eventObj Required **DocActionEvent**. A reference to the [DocActionEvent](#) object.

Remarks

This [event handler](#) does not allow users to cancel an operation.

Note The **OnClick** event for the InfoPath button control is the only control event that is supported.

Example

In the following example from the Events developer sample form, the `OnClick` event handler is used to perform [data validation](#) on some of the fields contained in the New Customer view when a user clicks a button to switch to another view:

```
function btnSwitchContact::OnClick(eventObj)
{
  if (XDocument.View.Name == "New Customer")
  {
    if (XDocument.DOM.selectSingleNode
      ('/Customers/CustomerInfo/CustomerName').text == ""
      && XDocument.DOM.selectSingleNode
      ('/Customers/CustomerInfo/CustomerID').text == "")
    {
      XDocument.UI.Alert("The Customer Name and ID must " +
        "be filled in prior to switching the view.");
      return;
    }
    else if (XDocument.DOM.selectSingleNode
      ('/Customers/CustomerInfo/CustomerName').text == "")
    {
      XDocument.UI.Alert("The Customer Name must be filled " +
        "in prior to switching the view.");
      return;
    }
    else if (XDocument.DOM.selectSingleNode
      ('/Customers/CustomerInfo/CustomerID').text == "")
    {
      XDocument.UI.Alert("The Customer ID must be filled in " +
        "prior to switching the view.");
      return;
    }
  }
}
```

```
    }  
  }  
  XDocument.View.SwitchView('Contact Customer');  
}
```


OnContextChange Event

Occurs after the context node changes.

```
Function XDocument::OnContextChange(ByRef pEvent As  
DocContextChangeEvent)
```

pEvent Required **DocContextChangeEvent**. A reference to the [DocContextChangeEvent](#) object.

Remarks

The context node is the XML DOM node mapped to the view that corresponds to the container (or item) with the current XML selection. For example, if the current selection in the view is in a [text box](#), the context node is the node to which the text box is bound. If the current selection is a [repeating section](#), the context node is the node for that item. If two repeating sections are selected, the context node is the ancestor XML DOM for both items mapped to the view.

The **OnContextChange** event is asynchronous. It does not fire on every change in the context node; instead, it fires after the application has stopped processing other events.

When the document loads, or when a view change occurs, the **OnContextChange** event will occur after the [OnLoad](#) and [OnSwitchView](#) events occur.

When the [IsUndoRedo](#) property of the **DocContextChangeEvent** object is **True**, the context change was caused by a user's undo or redo operation rather than an explicit user context change. Operations performed within the **OnContextChange** event handler that modify the XML DOM should be avoided in response to undo or redo operations, because they may interfere with the user's intention to revert data to a previous state.

For rich text box controls, the **OnContextChange** event is not raised for context changes within the XHTML content— that is, selection changes to the rich text in the control. The [GetContextNodes](#) method can be used to determine the selection within rich text box controls.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, instructions specific to different context nodes in a form are added to the body of a custom task pane:

```
function XDocument::OnContextChange(eventObj)
{
    var oContextNode = eventObj.Context;

    var strText = "";
    if( oContextNode.nodeName == "my:root" )
        strText = "";
    else if( oContextNode.nodeName == "my:singleName" )
        strText = "Type your full name.";
    else if( oContextNode.nodeName == "my:webSite" )
        strText = "Type the Web address of your personal web page.";

    var oTaskPane = XDocument.View.Window.TaskPanels.Item(0);
    oTaskPane.HTMLDocument.body.innerHTML = strText;
}
```


OnLoad Event

Occurs after a Microsoft Office InfoPath 2003 form has been loaded, but before any views have been initialized.

Function *XDocument::OnLoad*(ByRef *pEvent* As DocReturnEvent)

pEvent Required **DocReturnEvent**. A reference to the [DocReturnEvent](#) object.

Remarks

This [event handler](#) allows users to cancel an operation.

If the [ReturnStatus](#) property of the **DocReturnEvent** object is set to False, InfoPath cancels the loading of the form. If an error occurs in the scripting code for the OnLoad event handler, InfoPath ignores it and relies on the **ReturnStatus** property of the **DocReturnEvent** object. If the **ReturnStatus** property is not explicitly set, the default value of True is used.

Note When the **OnLoad** event occurs, the view is not initialized and the [XSL Transformation \(XSLT\)](#) used for the view is not yet loaded. The [XDocument](#) object is not added to the [XDocuments](#) collection until after the **OnLoad** event has occurred. However, the **XDocument** object is available during the **OnLoad** event.

Example

In the following example from the Sales Report sample form, the `OnLoad` event handler is used to determine whether the form has been digitally signed, and if it hasn't, to initialize some date values using a combination of scripting functions and custom functions:

```
function XDocument::OnLoad(objEvent)
{
    // Avoid DOM updates when the document has been digitally signed.
    if (XDocument.IsSigned)
        return;

    var today = new Date();
    initializeNodeValue("/sls:salesReport/sls:date", getDateString(today))
    initializeNodeValue("/sls:salesReport/sls:year", today.getFullYear());
}
```

This **Onload** event handler example depends on two custom functions, which are also from the Sales Report sample form: `initializeNodeValue` and `setNodeValue`.

```
function initializeNodeValue(xpath, strValue)
{
    var xmlNode = getNode(xpath);

    // Set the node value *ONLY* if the node is empty.
    if (xmlNode.text == "")
        setNodeValue(xmlNode, strValue);
}
```

```
function setNodeValue(xpath, value)
{
    var xmlNode = getNode(xpath);
```

```
if (!xmlNode)
    return;

// The xsi:nil needs to be removed before we set the value.
if (value != "" && xmlNode.getAttribute("xsi:nil"))
    xmlNode.removeAttribute("xsi:nil");

// Setting the value would mark the document as dirty.
// Let's do that if the value has really changed.
if (xmlNode.text != value)
    xmlNode.text = value;
}
```


OnMergeRequest Event

Occurs when the merge operation is invoked either from the Microsoft Office InfoPath 2003 user interface or from the command line by using the *aggregate* option.

Function *XDocument::OnMergeRequest*(ByRef *pEvent* As MergeEvent)
pEvent Required **MergeEvent**. A reference to the [MergeEvent](#) object.

Remarks

If the [ReturnStatus](#) property of the **MergeEvent** object is set to **False**, InfoPath cancels the merge operation. If an error occurs in the code for the **OnMergeRequest** event handler, InfoPath ignores the error and relies on the **ReturnStatus** property of the **MergeEvent** object. If the **ReturnStatus** property is not explicitly set, the default value of **False** is used.

For InfoPath forms stored in a Windows SharePoint Services form library, the **OnMergeRequest** event also occurs when the **MergeDocuments2** method of the **OpenXMLDocuments** control is executed. For more information on the **OpenXMLDocuments** control, see the SharePoint Products and Technologies 2003 Software Development Kit, which is available on the [Microsoft SharePoint Products and Technologies](#) Web site.

If you create an event handler for the **OnMergeRequest** event of a form template, you must edit the [form definition file \(.xsf\)](#) to set the [useScriptHandler](#) attribute to "yes" before it will run.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **OnMergeRequest** event handler performs a merge operation, and it sets variables to indicate the status of the merge operation:

```
var g_fMerging = false;
```

```
function XDocument::OnMergeRequest(eventObj)
{
    // Set global property to indicate that forms are being merged.
    if (eventObj.Index == 0)
        g_fMerging = true;

    XDocument.ImportDOM(eventObj.DOM);
    eventObj.ReturnStatus = true;

    if (eventObj.Index + 1 == eventObj.Count)
    {
        g_fMerging = false;
        XDocument.UI.Alert("Your request to merge " + eventObj.Count +
            " files is now complete.");
    }
}
```

OnSaveRequest Event

Occurs when the save operation is invoked from the Microsoft Office InfoPath 2003 user interface or by using the [Save](#) or [SaveAs](#) method of the [XDocument](#) object in the InfoPath object model.

Function *XDocument*::**OnSaveRequest**(ByRef *pEvent* As SaveEvent)

pEvent Required **SaveEvent**. A reference to the [SaveEvent](#) object.

Remarks

If the [ReturnStatus](#) property of the **SaveEvent** object is set to **False**, InfoPath cancels the save operation. If an error occurs in the code for the **OnSaveRequest** event handler, InfoPath ignores the error and relies on the **ReturnStatus** property of the **SaveEvent** object. If the **ReturnStatus** property is not explicitly set, the default value of **False** is used.

The **ReturnStatus** property works in conjunction with the [IsCancelled](#) property when the InfoPath form is closing. If the document has changes that have not been saved and the user cancels the save operation, the **IsCancelled** property can be set to **True** to allow InfoPath to close.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

Example

In the following example, the **OnSaveRequest** event handler is used to create an **XMLHTTP** object for transporting the form's underlying XML document:

```
function XDocument::OnSaveRequest(eventObj)
{
    // Write the code to be run before saving here.

    XDocument.UI.Alert("Begin saving form.");

    eventObj.IsCancelled = eventObj.PerformSaveOperation();

    // Write the code to be run after saving here.

    XDocument.UI.Alert("Form saved.");

    eventObj.ReturnStatus = true;
}
```

OnSign Event

Occurs after a set of signed data has been selected to sign.

Function *XDocument*::**OnSign**(ByRef *pEvent* As SignEvent)

pEvent Required **SignEvent**. A reference to the [SignEvent](#) object.

Remarks

You can use this event to add additional data to the digital signature. For example, you can add data from a trusted timestamp server, or add a server-side countersignature of the transaction. You can also use this event to block signing if the current user is not a member of a particular group.

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.

OnSubmitRequest Event

Occurs when the submit operation is invoked either from the Microsoft Office InfoPath 2003 user interface or by using the [Submit](#) method of the [XDocument](#) object in the InfoPath object model.

```
Function XDocument::OnSubmitRequest(ByRef pEvent As  
DocReturnEvent)
```

pEvent Required **DocReturnEvent**. A reference to the [DocReturnEvent](#) object.

Remarks

This [event handler](#) allows users to cancel an operation.

If the [ReturnStatus](#) property of the **DocReturnEvent** object is set to False, InfoPath cancels the submit operation. If an error occurs in the scripting code for the OnSubmitRequest event handler, InfoPath ignores it and relies on the **ReturnStatus** property of the **DocReturnEvent** object. If the **ReturnStatus** property is not explicitly set, the default value of False is used.

Example

In the following example, the `OnSubmitRequest` event handler is used to create an **XMLHTTP** object for transporting the form's underlying XML document:

```
function XDocument::OnSubmitRequest(eventObj)
{
    // Create an XMLHTTP object for document transport.
    try
    {
        var objXmlHttp = new ActiveXObject("MSXML2.XMLHTTP");
    }
    catch(ex)
    {
        XDocument.UI.Alert("Could not create MSXML2.XMLHTTP
            object.\r\n" + ex.number + " - " + ex.description);

        // Return with eventObj.ReturnStatus == false,
        // because no change was made to this value.
        return;
    }

    // Post the XML document to strUrl.
    objXmlHttp.open("POST", strUrl, false);
    try
    {
        objXmlHttp.send(XDocument.DOM.xml);
    }
    catch(ex)
    {
        XDocument.UI.Alert("Could not post (ASP) document to " +
```

```
        strUrl + "\r\n" + ex.number + " - " + ex.description);

    // Return with eventObj.ReturnStatus == false.
    return;
}
// If here, the submit operation is has been successful.
eventObj.ReturnStatus = true;
}
```


OnSwitchView Event

Occurs after a view in a Microsoft Office InfoPath 2003 form has been successfully switched.

Function *XDocument::OnSwitchView*(ByRef *pEvent* As DocEvent)

pEvent Required **DocEvent**. A reference to the [DocEvent](#) object.

Remarks

This [event handler](#) does not allow users to cancel an operation.

Note The **OnSwitchView** event also occurs when a form is first opened.

Example

In the following example from the Events developer sample form, the OnSwitchView event handler is used to check the name of the current view. If the view has the name "Archive Customer," data is added to the form's underlying XML document.

```
function XDocument::OnSwitchView(eventObj)
{
    var oDate = new Date();

    if (XDocument.View.Name == "Archive Customer")
    {
        var oNotesNode = XDocument.DOM.
            selectSingleNode("/Customers/CustomerInfo/Notes");
        var oDivNode = XDocument.DOM.
            createNode(1, "div", "http://www.w3.org/1999/xhtml");

        oDivNode.text = "Note recorded " + oDate.toString();
        oNotesNode.appendChild(oDivNode);
    }
}
```


OnValidate Event

Occurs after changes to a form's underlying XML document have been accepted but before the [OnAfterChange](#) event occurs.

Function *node::OnValidate*(ByRef *pDataDOMEvent* As DataDOMEvent)

pDataDOMEvent Required **DataDOMEvent**. A reference to the [DataDOMEvent](#) object.

Remarks

This [event handler](#) does not allow users to cancel an operation.

During the **OnValidate** event, the form's underlying XML document is placed in read-only mode.

The **OnValidate** event is typically used for handling errors and working with the [Errors](#) collection— for example, adding new errors or deleting existing ones.

Note In some cases, events related to changes in a form's underlying XML document may occur more than once. For example, when existing data is changed, an insert and delete operation occurs.

Example

In the following partial example from the Events developer sample form, the OnValidate event handler is used to validate contact information. If the data is invalid, the [ReportError](#) method of the **DataDOMEvent** object is used to create an error.

```
function msoxd__ContactDates::OnValidate(eventObj)
{
    var iNumberOfDays = 0;
    var objEmailDate = XDocument.DOM.selectSingleNode
        ('/Customers/CustomerInfo/ContactDates/EmailCampaignDt');
    var objPhoneContactDate = XDocument.DOM.selectSingleNode
        ('/Customers/CustomerInfo/ContactDates/PhoneContactDt');
    var objRepVisitDate = XDocument.DOM.selectSingleNode
        ('/Customers/CustomerInfo/ContactDates/RepVisitDt');

    // First validate the email and phone contact dates.
    if (!objEmailDate || !objPhoneContactDate)
        return;

    var emailDate =
        new Date(objEmailDate.text.replace(/(.*)-(.*)-(.*)/, "$2-$3-$1"));
    var phoneContactDate =
        new Date(objPhoneContactDate.text.replace(/(.*)-(.*)-(.*)/, "$2-$3

    if (isNaN(emailDate) || isNaN(phoneContactDate))
        return;

    // Get the number of days between the two dates.
    iNumberOfDays = GetElapsedDays(emailDate, phoneContactDate);

    if (iNumberOfDays < REQUIRED_PHONE_EMAIL_INTERVAL)
```

```
eventObj.ReportError(objPhoneContactDate,  
    "The Phone Contact Start date must occur after " +  
    REQUIRED_PHONE_EMAIL_INTERVAL +  
    " days from the start of the Email Campaign.", false);  
...  
}
```


OnVersionUpgrade Event

Occurs when the version number of a Microsoft Office InfoPath 2003 form being opened is older than the version number of the [form template](#) on which it is based.

Function *XDocument::OnVersionUpgrade*(ByRef *pEvent* As VersionUpgradeEvent)

pEvent Required **VersionUpgradeEvent**. A reference to the [VersionUpgradeEvent](#) object.

Remarks

This [event handler](#) allows users to cancel an operation.

During the **OnVersionUpgrade** event, the form's underlying XML document is placed in read-only mode, and it is not validated against the form's associated [XML Schema](#). If the [ReturnStatus](#) property of the **VersionUpgradeEvent** object is set to False, InfoPath cancels the opening of the form. If an error occurs in the scripting code for the OnVersionUpgrade event handler, InfoPath ignores it and relies on the **ReturnStatus** property of the **VersionUpgradeEvent** object. If the **ReturnStatus** property is not explicitly set, the default value of True is used.

Example

In the following example from the Events developer sample form, the **OnVersionUpgrade** event handler is used to determine whether the form with the incorrect older version number contains an **EmailAddress** element. If it does not, one is added.

```
function XDocument::OnVersionUpgrade(eventObj)
{
  if (!XDocument.DOM.selectSingleNode("/Customers/CustomerInfo/
  {
    try
    {
      // Create the new element.
      var objItemNode = XDocument.DOM.selectSingleNode("/Custo:
        .ownerDocument.createElement("EmailAddress");

      // Add the new <item> element to the XML document as a
      // child of the <order> element.
      XDocument.DOM.selectSingleNode("/Customers/CustomerInfo'
        .appendChild(objItemNode);
      eventObj.ReturnStatus=true;
    }
    catch(ex)
    {
      XDocument.UI.Alert("There was an error inserting the " +
        "<EmailAddress> node.\nDescription: " + ex.description);
      eventObj.ReturnStatus=false;
    }
  }
}
```


XdCertificateStatus Enumeration

The **XdCertificateStatus** enumeration is used to determine the status of a [digital certificate](#). These enumerated values are returned by the [Status](#) property of the [Certificate](#) object.

| Name | Value | Description |
|--------------------------------------|-------|---|
| xdCertificateStatusError | 0 | The status of the specified digital certificate cannot be determined. |
| xdCertificateStatusValid | 1 | The specified digital certificate is valid. |
| xdCertificateStatusExpired | 2 | The specified digital certificate has expired. |
| xdCertificateStatusNotTrusted | 3 | The specified digital certificate is not trusted. |
| xdCertificateStatusRevoked | 4 | The specified digital certificate has been revoked. |

XdConfirmButtons Enumeration

The **XdConfirmButtons** enumeration is used to determine the type of buttons to be displayed in a message box. These enumerated values are used by the [Confirm](#) method of the [UI](#) object.

| Name | Value | Description |
|----------------------|-------|---|
| xdOKCancel | 1 | OK and Cancel buttons are displayed. |
| xdYesNoCancel | 3 | Yes , No , and Cancel buttons are displayed. |
| xdYesNo | 4 | Yes and No buttons are displayed |

XdConfirmChoice Enumeration

The **XdConfirmChoice** enumeration is used to determine which button is clicked in a message box. These enumerated values are used by the [Confirm](#) method of the [UI](#) object.

| Name | Value | Description |
|-----------------|-------|---------------------------------------|
| xdOK | 1 | The OK button was clicked. |
| xdCancel | 2 | The Cancel button was clicked. |
| xdYes | 6 | The Yes button was clicked. |
| xdNo | 7 | The No button was clicked. |

XdDocumentVersionMode Enumeration

The **XdDocumentVersionMode** enumeration is used to determine how a Microsoft Office InfoPath 2003 form will be opened. These enumerated values are used as arguments to the [New](#) and [Open](#) methods of the [ExternalApplication](#) object.

| Name | Value | Description |
|---|-------|---|
| xdCanOpenInReadOnlyMode | 8 | The form is opened in read-only mode. |
| xdCanTransformSigned | 16 | The form is opened and its transform applied, even though it has a digital signature . |
| xdFailOnVersionMismatch | 0 | The form is created or opened only if it is accessible and its version number matches the version number of the form template . |
| xdFailOnVersionOlder | 1 | The form is created or opened only if it is accessible and its version number is not older than the version number of the form template. |
| xdIgnoreDataAdaptersQueryFailure | 64 | The form is opened even if the query associated with its data adapter fails to return data. |
| xdPromptTransformSigned | 32 | The form is opened and the Digital Signatures dialog box is displayed before the form's transform is applied. |
| xdUseExistingVersion | 2 | The form is created or opened using the existing |

version number of the form
template.

XdMachineOnlineState Enumeration

The **XdMachineOnlineState** enumeration is used to determine the state of the connection for the client computer. These enumerated values are returned by the [MachineOnlineState](#) property of the [Application](#) object.

| Name | Value | Description |
|----------------------------|-------|--|
| xdOffline | 0 | The client computer is not connected to the network. |
| xdOnline | 1 | The client computer is connected to the network. |
| xdIEIsInOfflineMode | 2 | Microsoft Internet Explorer is in offline mode. |

XdSignatureRelation Enumeration

The **XdSignatureRelation** enumeration is used to determine how digital signatures can be added to a [SignedDataBlock](#) object. These enumerated values are read-only.

| Name | Value | Description |
|---------------------------------------|-------|---|
| xdSignatureRelationCoSign | 2 | Specifies that signatures are independent of each other in the SignedDataBlock object. |
| xdSignatureRelationCounterSign | 3 | Specifies that each signature signs the preceding signature in the SignedDataBlock object. |
| xdSignatureRelationSingle | 1 | Specifies that only one signature can exist in the SignedDataBlock object. |

XdSignatureStatus Enumeration

The **XdSignatureStatus** enumeration is used to determine the status of a [digital signature](#). These enumerated values are returned by the [Status](#) property of the [Signature](#) object.

| Name | Value | Description |
|-------------------------------------|-------|---|
| xdSignatureStatusError | 0 | The status of the specified digital signature cannot be determined. |
| xdSignatureStatusValid | 1 | The specified digital signature is valid. |
| xdSignatureStatusInvalid | 2 | The specified digital signature is invalid. |
| xdSignatureStatusUnsupported | 3 | The specified digital signature is not supported by Microsoft Office InfoPath 2003. |

XdTaskPaneType Enumeration

The **XdTaskPaneType** enumeration is used to determine the type of Microsoft Office InfoPath 2003 task pane that is currently displayed. These enumerated values are returned by the **Type** property of the **TaskPane** object. These values are also used as the positional argument to the **Item** property of the **TaskPanes** collection.

| Name | Value | Description |
|-----------------------------------|-------|--|
| xdTaskPaneBulletsNumbering | 9 | The Bullets and Numbering task pane. Used for applying bullet and numbering styles to a form. |
| xdTaskPaneClipArt | 5 | The Clip Art task pane. Used for inserting clip art into a form. |
| xdTaskPaneDesignerNew | 2 | The Design a Form task pane. Used to design a new form, open a form in design mode , or select a form to fill out. |
| xdTaskPaneFillOutAForm | 1 | The Fill Out a Form task pane. Used to open a form to fill out or design. |
| xdTaskPaneFind | 6 | The Find task pane. Used to search for text in a form. |
| xdTaskPaneFormatting | 8 | The Font task pane. Used to format text. |
| xdTaskPaneHelp | 4 | The Help task pane. Used to display the InfoPath Help system. |
| xdTaskPaneHTML | 0 | The InfoPath custom task pane . Used by form developers to provide extra form functionality. |
| xdTaskPaneHWSWorkflow | 12 | The Microsoft BizTalk Server 2004 Human Workflow Services (HWS) Workflow task pane. |

| | | |
|---------------------------------|----|---|
| xdTaskPaneParaFormatting | 11 | The Paragraph task pane. Used to configure properties of paragraphs including alignment and spacing. |
| xdTaskPaneReplace | 7 | The Replace task pane. Used to find and replace text in a form. |
| xdTaskPaneSearchResults | 3 | The Search Results task pane. Used to display the results of a search. |
| xdTaskPaneSpelling | 10 | The Spelling task pane. Used to check the spelling in a form. |

XdWindowState Enumeration

The **XdWindowState** enumeration is used to determine the state of the Microsoft Office InfoPath 2003 window that is represented by the [Window](#) object. These enumerated values are returned or set by the [WindowState](#) property of the **Window** object.

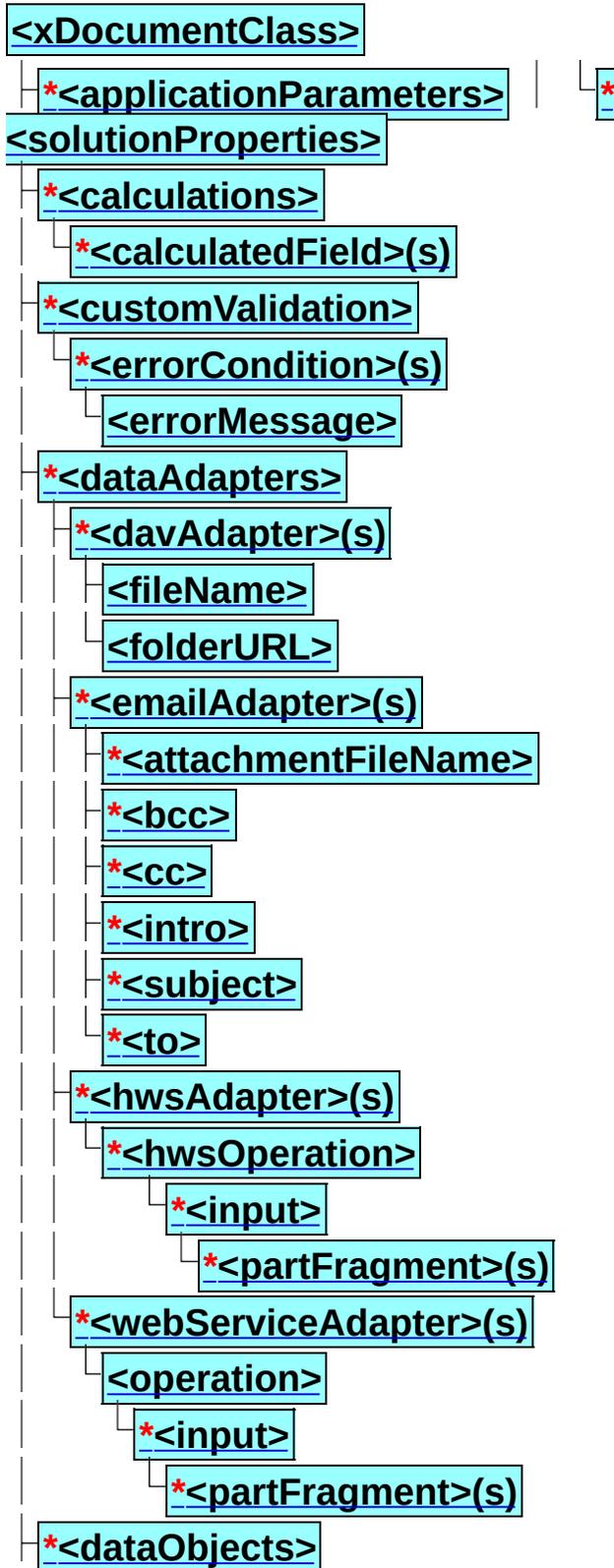
| Name | Value | Description |
|------------------------------|-------|---|
| xdWindowStateMaximize | 1 | The window is maximized. |
| xdWindowStateNormal | 2 | The window is not maximized or minimized. |
| xdWindowStateMinimize | 3 | The window is minimized. |

XdWindowType Enumeration

The **XdWindowType** enumeration is used to determine the type of Microsoft Office InfoPath 2003 window that is currently displayed. These enumerated values are returned by the **Type** property of the **Window** object.

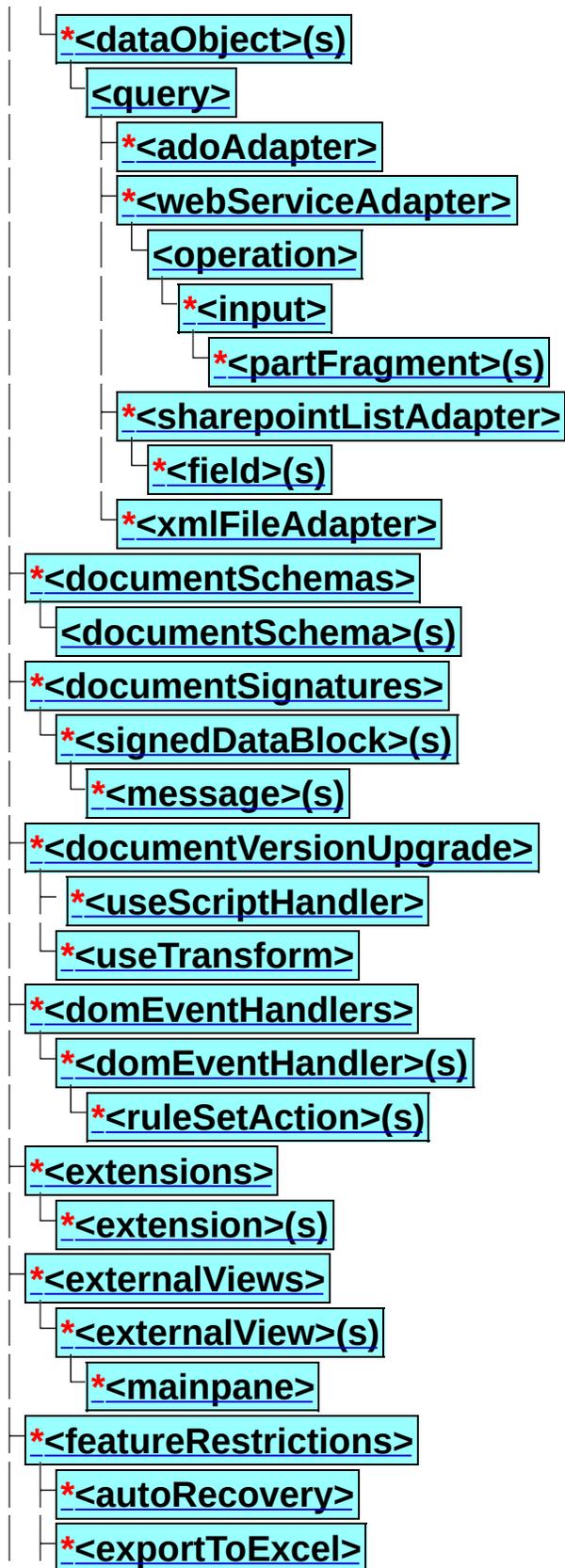
| Name | Value | Description |
|-------------------------|-------|---|
| xdDesignerWindow | 1 | The current window is the window that is displayed in design mode . |
| xdEditorWindow | 0 | The current window is the window that is displayed when filling out a form. |

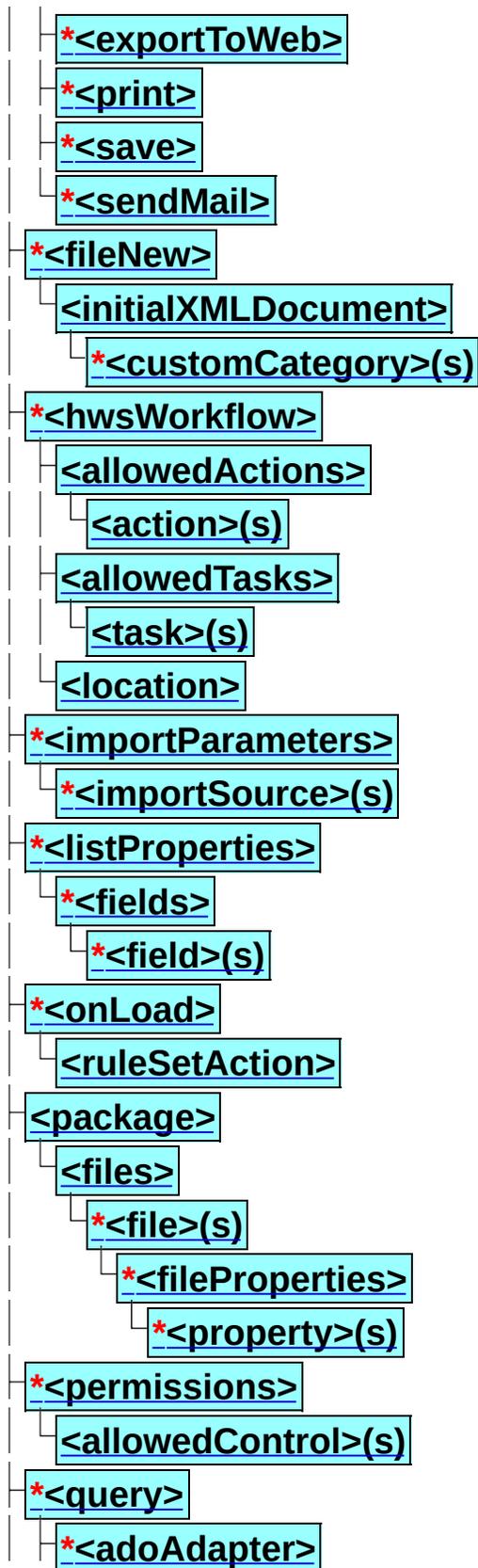
InfoPath XSF Diagram

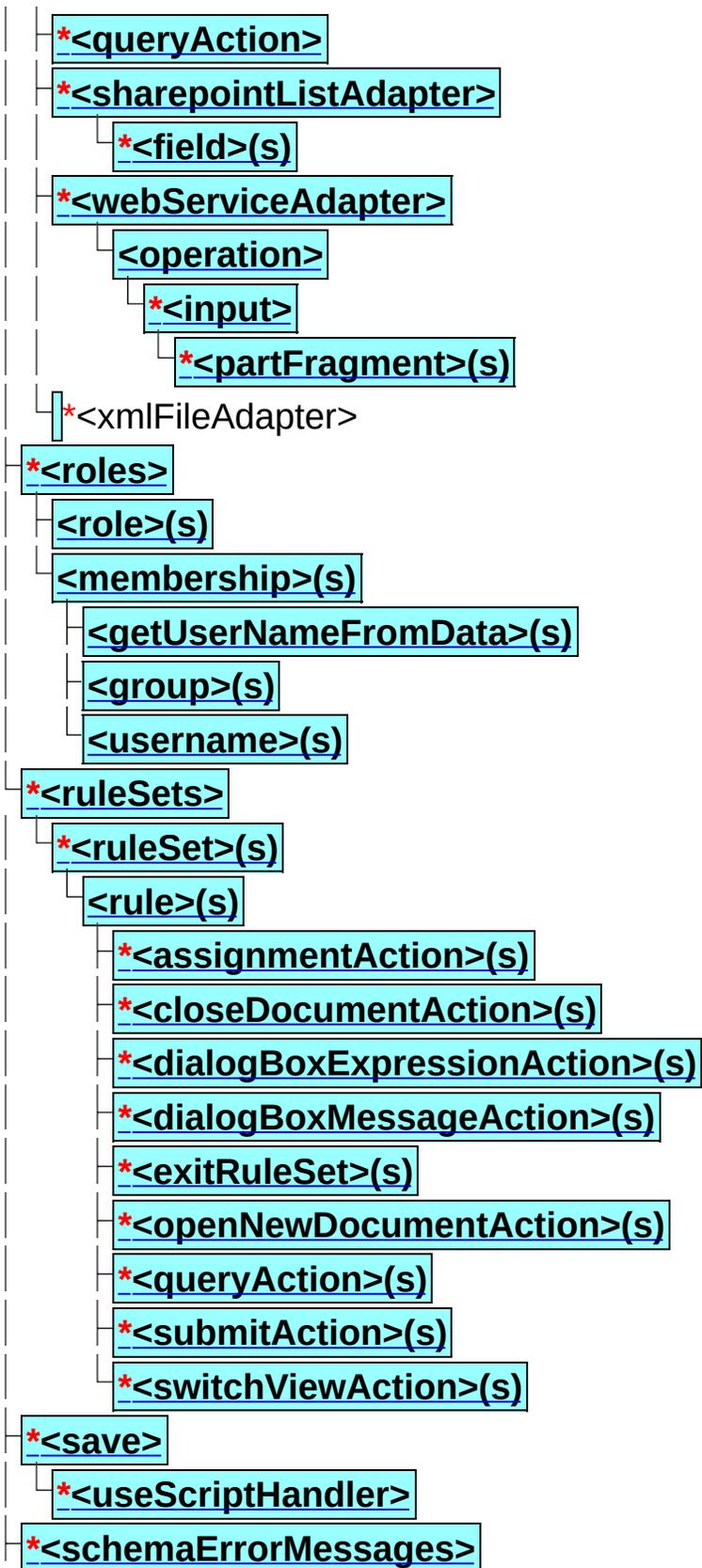


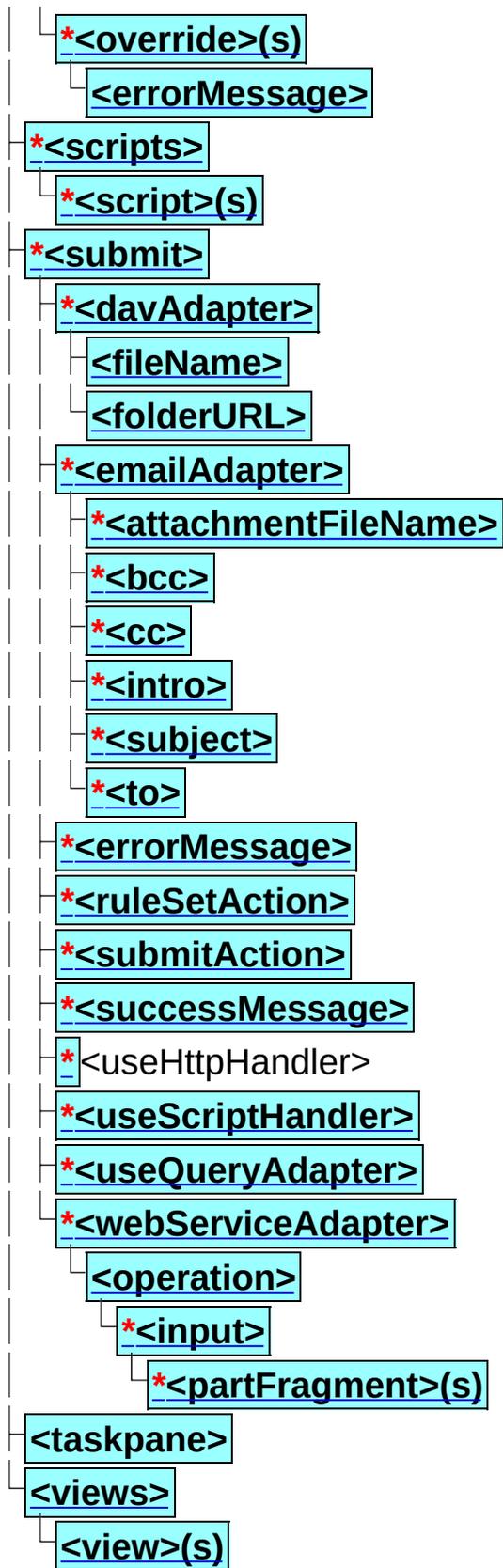
Legend

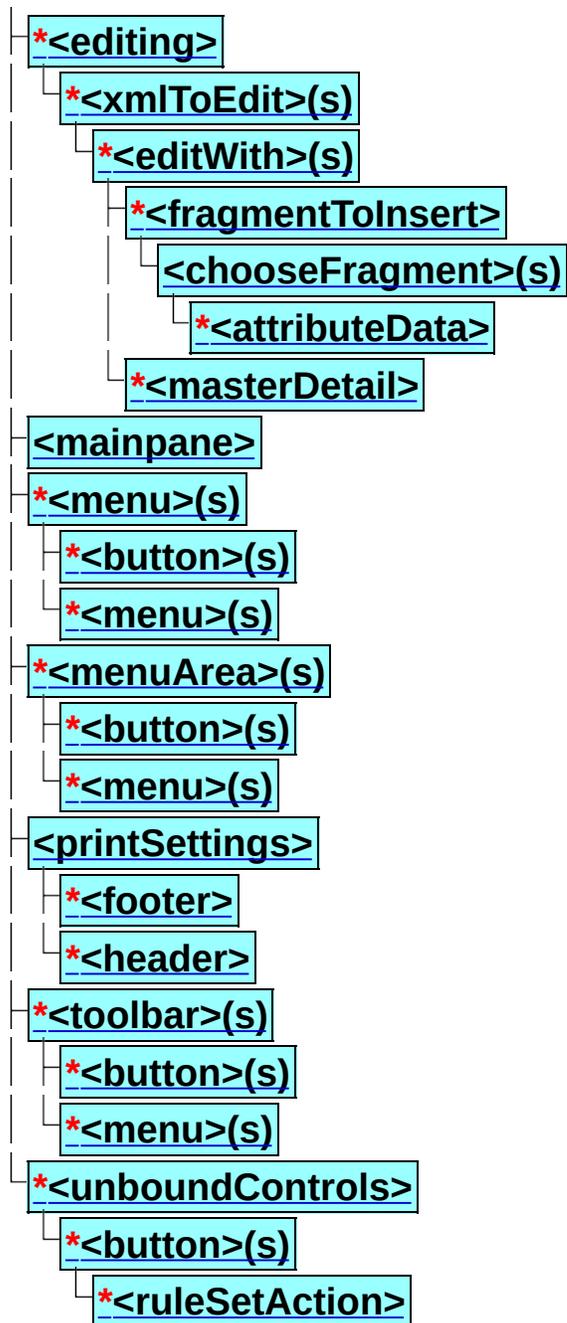
- * Optional Element
- (s) Repeating Element












```
<xsd:simpleType name="xdYesNo">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="yes" />
    <xsd:enumeration value="no" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdEnabledDisabled type -->
<xsd:simpleType name="xdEnabledDisabled">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="enabled" />
    <xsd:enumeration value="disabled" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdManualAuto type -->
<xsd:simpleType name="xdManualAuto">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="manual" />
    <xsd:enumeration value="automatic" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdExpressionLiteral type -->
<xsd:simpleType name="xdExpressionLiteral">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="expression" />
    <xsd:enumeration value="literal" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdFileName type -->
<xsd:simpleType name="xdFileName">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1" />
    <xsd:maxLength value="64" />
  </xsd:restriction>
</xsd:simpleType>
```

```

</xsd:simpleType>
<!-- xdScriptLanguage type -->
<xsd:simpleType name="xdScriptLanguage">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:pattern value="((([Jj][Aa][Vv][Aa])|((([Jj])|([Vv])[Bb
  </xsd:restriction>
</xsd:simpleType>
<!-- xdSolutionVersion type -->
<xsd:simpleType name="xdSolutionVersion">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="(([0-9]{1,4}.){3}[0-9]{1,4})" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdEmptyString type -->
<xsd:simpleType name="xdEmptyString">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="0" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdErrorMessage type -->
<xsd:simpleType name="xdErrorMessage">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="1023" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdDesignMode type -->
<xsd:simpleType name="xdDesignMode">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="normal" />
    <xsd:enumeration value="protected" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdTrustLevel type -->

```

```

<xsd:simpleType name="xdTrustLevel">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="restricted" />
    <xsd:enumeration value="domain" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdSignedDataBlockName type -->
<xsd:simpleType name="xdSignedDataBlockName">
  <xsd:restriction base="xsd:ID">
    <xsd:minLength value="1" />
    <xsd:maxLength value="255" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdSignedDataBlockMessage type -->
<xsd:simpleType name="xdSignedDataBlockMessage">
  <xsd:restriction base="xsd:string">
    <xsd:maxLength value="255" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdSignatureRelationEnum type -->
<xsd:simpleType name="xdSignatureRelationEnum">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="countersign" />
    <xsd:enumeration value="cosign" />
    <xsd:enumeration value="single" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xdHWSname type -->
<xsd:simpleType name="xdHWSname">
  <xsd:restriction base="xsd:NCName">
    <xsd:pattern value="^[^-\.\@\[\]\[\^\]\|\+^?\*\^@\{\}\}^\s" />
  </xsd:restriction>
</xsd:simpleType>

```

```
<!-- xdHWSCaption type -->
<xsd:simpleType name="xdHWSCaption">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1" />
    <xsd:maxLength value="255" />
  </xsd:restriction>
</xsd:simpleType>
<!-- xDocumentClass -->
<xsd:element name="xDocumentClass">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf:package" minOccurs="1" />
      <xsd:element ref="xsf:permissions" minOccurs="0" />
      <xsd:element ref="xsf:views" minOccurs="1" />
      <xsd:element ref="xsf:hwsWorkflow" minOccurs="0" />
      <xsd:element ref="xsf:externalViews" minOccurs="0" />
      <xsd:element ref="xsf:scripts" minOccurs="0" />
      <xsd:element ref="xsf:schemaErrorMessages" minOccurs="0" />
      <xsd:element ref="xsf:documentSchemas" minOccurs="0" />
      <xsd:element ref="xsf:applicationParameters" minOccurs="0" />
      <xsd:element ref="xsf:featureRestrictions" minOccurs="0" />
      <xsd:element ref="xsf:fileNew" minOccurs="0" />
      <xsd:element ref="xsf:customValidation" minOccurs="0" />
      <xsd:element ref="xsf:domEventHandlers" minOccurs="0" />
      <xsd:element ref="xsf:importParameters" minOccurs="0" />
      <xsd:element ref="xsf:listProperties" minOccurs="0" />
      <xsd:element ref="xsf:taskpane" minOccurs="0" />
      <xsd:element ref="xsf:documentSignatures" minOccurs="0" />
      <xsd:element ref="xsf:dataObjects" minOccurs="0" />
      <xsd:element ref="xsf:dataAdapters" minOccurs="0" />
      <xsd:element ref="xsf:query" minOccurs="0" />
      <xsd:element ref="xsf:submit" minOccurs="0" />
      <xsd:element ref="xsf:save" minOccurs="0" />
    </xsd:all>
  </xsd:complexType>
</xsd:element>

```

```

    <xsd:element ref="xsf:roles" minOccurs="0" />
    <xsd:element ref="xsf:onLoad" minOccurs="0" />
    <xsd:element ref="xsf:documentVersionUpgrade"
    <xsd:element ref="xsf:extensions" minOccurs="0"
    <xsd:element ref="xsf:ruleSets" minOccurs="0" />
    <xsd:element ref="xsf:calculations" minOccurs="(
</xsd:all>
<xsd:attribute name="name" type="xsd:string" use="op
<xsd:attribute name="author" type="xsd:string" use="o
<xsd:attribute name="description" use="optional">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="255" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="solutionVersion" type="xsf:xdSol
<xsd:attribute name="productVersion" type="xsd:string
<xsd:attribute name="solutionFormatVersion" type="xs
<xsd:attribute name="dataFormSolution" type="xsf:xd\
<xsd:attribute name="requireFullTrust" type="xsf:xdYe
<xsd:attribute name="trustLevel" type="xsf:xdTrustLev
<xsd:attribute name="trustSetting" type="xsf:xdManua
<xsd:attribute name="publishUrl" type="xsd:string" use
</xsd:complexType>
<xsd:key name="view_name_key">
    <xsd:selector xpath="./xsf:views/xsf:view" />
    <xsd:field xpath="@name" />
</xsd:key>
<xsd:key name="externalView_name_key">
    <xsd:selector xpath="./xsf:externalViews/xsf:externalV
    <xsd:field xpath="@name" />
</xsd:key>

```

```

<xsd:key name="view_or_externalView_name_key">
  <xsd:selector xpath="./xsf:views/xsf:view | ./xsf:externa
  <xsd:field xpath="@name" />
</xsd:key>
<xsd:key name="ruleset_name_key">
  <xsd:selector xpath="./xsf:ruleSets/xsf:ruleSet" />
  <xsd:field xpath="@name" />
</xsd:key>
<xsd:key name="dataObject_name_key">
  <xsd:selector xpath="./xsf:dataObjects/xsf:dataObject"
  <xsd:field xpath="@name" />
</xsd:key>
<xsd:unique name="adapter_name_unique">
  <xsd:selector xpath="./xsf:dataObjects/xsf:dataObject/x
  <xsd:field xpath="@name" />
</xsd:unique>
<xsd:key name="adapter_name_key">
  <xsd:selector xpath="./xsf:dataAdapters/*" />
  <xsd:field xpath="@name" />
</xsd:key>
<xsd:unique name="view_external_name_unique">
  <xsd:selector xpath="./xsf:views/xsf:view | ./xsf:externa
  <xsd:field xpath="@name" />
</xsd:unique>
</xsd:element>
<!-- schemaErrorMessages -->
<xsd:element name="schemaErrorMessages">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:override" minOccurs="0" n
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

```

```

<!-- override -->
<xsd:element name="override">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:errorMessage" />
    </xsd:sequence>
    <xsd:attribute name="match" type="xsd:string" use="required" />
  </xsd:complexType>
</xsd:element>
<!-- applicationParameters -->
<xsd:element name="applicationParameters">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf:solutionProperties" minOccurs="1" />
    </xsd:all>
    <xsd:attribute name="application" use="required" />
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="InfoPath Designer" />
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:complexType>
</xsd:element>
<!-- solutionProperties -->
<xsd:element name="solutionProperties">
  <xsd:complexType>
    <xsd:attribute name="allowCustomization" type="xsf:boolean" />
    <xsd:attribute name="lastOpenView" use="optional" />
    <xsd:attribute name="scriptLanguage" type="xsf:xdScriptLanguage" />
    <xsd:attribute name="automaticallyCreateNodes" type="xsf:boolean" />
    <xsd:attribute name="lastVersionNeedingTransform" type="xsf:boolean" />
    <xsd:attribute name="fullyEditableNamespace" type="xsf:boolean" />
  </xsd:complexType>
</xsd:element>

```

```

        <xsd:attribute name="publishSaveUrl" type="xsd:string" />
    </xsd:complexType>
</xsd:element>
<!-- featureRestrictions -->
<xsd:element name="featureRestrictions">
    <xsd:complexType>
        <xsd:all>
            <xsd:element name="save" minOccurs="0">
                <xsd:complexType>
                    <xsd:attribute name="ui" type="xsd:boolean" />
                </xsd:complexType>
            </xsd:element>
            <xsd:element ref="xsd:exportToWeb" minOccurs="0" />
            <xsd:element ref="xsd:exportToExcel" minOccurs="0" />
            <xsd:element ref="xsd:print" minOccurs="0" />
            <xsd:element ref="xsd:sendMail" minOccurs="0" />
            <xsd:element ref="xsd:autoRecovery" minOccurs="0" />
        </xsd:all>
    </xsd:complexType>
</xsd:element>
<!-- exportToWeb -->
<xsd:element name="exportToWeb">
    <xsd:complexType>
        <xsd:attribute name="ui" type="xsd:boolean" />
    </xsd:complexType>
</xsd:element>
<!-- exportToExcel -->
<xsd:element name="exportToExcel">
    <xsd:complexType>
        <xsd:attribute name="ui" type="xsd:boolean" />
    </xsd:complexType>
</xsd:element>
<!-- print -->

```

```

<xsd:element name="print">
  <xsd:complexType>
    <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" />
  </xsd:complexType>
</xsd:element>
<!-- sendMail -->
<xsd:element name="sendMail">
  <xsd:complexType>
    <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" />
  </xsd:complexType>
</xsd:element>
<!-- autoRecovery -->
<xsd:element name="autoRecovery">
  <xsd:complexType>
    <xsd:attribute name="feature" type="xsf:xdEnabledDis" />
  </xsd:complexType>
</xsd:element>
<!-- query -->
<xsd:element name="query">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="xsf:queryAction" />
      <xsd:element ref="xsf:adoAdapter" />
      <xsd:element ref="xsf:webServiceAdapter" />
      <xsd:element ref="xsf:xmlFileAdapter" />
      <xsd:element ref="xsf:sharepointListAdapter" />
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
<!-- scripts -->
<xsd:element name="scripts">
  <xsd:complexType>
    <xsd:sequence>

```

```

        <xsd:element ref="xsf:script" minOccurs="0" max
    </xsd:sequence>
    <xsd:attribute name="language" type="xsf:xdScriptLan
    <xsd:attribute name="enforceScriptTimeout" type="xsf
    </xsd:complexType>
</xsd:element>
<xsd:element name="script">
    <xsd:complexType>
        <xsd:attribute name="src" type="xsf:xdFileName" use=
    </xsd:complexType>
</xsd:element>
<!-- dataObjects -->
<xsd:element name="dataObjects">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="xsf:dataObject" />
        </xsd:choice>
    </xsd:complexType>
    <xsd:unique name="dataObjects_name_unique">
        <xsd:selector xpath="./xsf:dataObject" />
        <xsd:field xpath="@name" />
    </xsd:unique>
</xsd:element>
<xsd:element name="dataObject">
    <xsd:complexType>
        <xsd:choice>
            <xsd:element name="query">
                <xsd:complexType>
                    <xsd:choice>
                        <xsd:element ref="xsf:adoAdapter"
                        <xsd:element ref="xsf:webService/
                        <xsd:element ref="xsf:xmlFileAdař
                        <xsd:element ref="xsf:sharepointLi

```

```

        </xsd:choice>
    </xsd:complexType>
</xsd:element>
</xsd:choice>
<xsd:attribute name="name" type="xsd:string" use="required"/>
<xsd:attribute name="schema" type="xsd:string" use="required"/>
<xsd:attribute name="initOnLoad" type="xsd:boolean" use="optional"/>
</xsd:complexType>
</xsd:element>
<!-- dataAdapters -->
<xsd:element name="adoAdapter">
    <xsd:complexType>
        <xsd:attribute name="name" type="xsd:string" use="optional"/>
        <xsd:attribute name="connectionString" type="xsd:string" use="optional"/>
        <xsd:attribute name="commandText" type="xsd:string" use="optional"/>
        <xsd:attribute name="queryAllowed" type="xsd:boolean" use="optional"/>
        <xsd:attribute name="submitAllowed" type="xsd:boolean" use="optional"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="webServiceAdapter">
    <xsd:complexType>
        <xsd:choice>
            <xsd:element ref="xsd:operation" />
        </xsd:choice>
        <xsd:attribute name="name" type="xsd:string" use="optional"/>
        <xsd:attribute name="wsdlUrl" type="xsd:string" use="optional"/>
        <xsd:attribute name="queryAllowed" type="xsd:boolean" use="optional"/>
        <xsd:attribute name="submitAllowed" type="xsd:boolean" use="optional"/>
        <xsd:attribute name="useDataSet" type="xsd:boolean" use="optional"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="hwsAdapter">
    <xsd:complexType>

```

```

        <xsd:choice>
            <xsd:element ref="xsd:hwsOperation" />
        </xsd:choice>
        <xsd:attribute name="name" type="xsd:string" use="required" />
        <xsd:attribute name="wsdlUrl" type="xsd:string" use="required" />
        <xsd:attribute name="queryAllowed" type="xsd:boolean" use="optional" />
        <xsd:attribute name="submitAllowed" type="xsd:boolean" use="optional" />
    </xsd:complexType>
</xsd:element>
<xsd:element name="operation">
    <xsd:complexType>
        <xsd:choice>
            <xsd:element ref="xsd:input" minOccurs="0" />
        </xsd:choice>
        <xsd:attribute name="name" type="xsd:string" use="required" />
        <xsd:attribute name="soapAction" type="xsd:string" use="optional" />
        <xsd:attribute name="serviceUrl" type="xsd:string" use="optional" />
    </xsd:complexType>
</xsd:element>
<xsd:element name="hwsOperation">
    <xsd:complexType>
        <xsd:choice>
            <xsd:element ref="xsd:input" />
        </xsd:choice>
        <xsd:attribute name="type" type="xsd:string" use="required" />
        <xsd:attribute name="typeID" type="xsd:string" use="required" />
        <xsd:attribute name="serviceUrl" type="xsd:string" use="optional" />
    </xsd:complexType>
</xsd:element>
<xsd:element name="input">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="xsd:partFragment" />
        </xsd:choice>
    </xsd:complexType>
</xsd:element>

```

```

        </xsd:choice>
        <xsd:attribute name="source" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="partFragment">
    <xsd:complexType>
        <xsd:attribute name="match" type="xsd:string" use="required"/>
        <xsd:attribute name="replaceWith" type="xsd:string" use="required"/>
        <xsd:attribute name="sendAsString" type="xsd:boolean" use="optional"/>
        <xsd:attribute name="dataObject" type="xsd:string" use="optional"/>
        <xsd:attribute name="filter" type="xsd:string" use="optional"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="xmlFileAdapter">
    <xsd:complexType>
        <xsd:attribute name="name" type="xsd:string" use="optional"/>
        <xsd:attribute name="fileUrl" type="xsd:anyURI" use="optional"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="sharepointListAdapter">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="field" minOccurs="0" maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:attribute name="sharepointName" type="xsd:string" use="optional"/>
                    <xsd:attribute name="infopathName" type="xsd:string" use="optional"/>
                    <xsd:attribute name="isLookup" type="xsd:boolean" use="optional"/>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string" use="required"/>
        <xsd:attribute name="siteUrl" type="xsd:string" use="required"/>
        <xsd:attribute name="sharepointGuid" type="xsd:string" use="optional"/>
    </xsd:complexType>
</xsd:element>

```

```

        <xsd:attribute name="infopathGroup" type="xsd:string" />
        <xsd:attribute name="queryAllowed" type="xsd:boolean" />
        <xsd:attribute name="submitAllowed" type="xsd:boolean" />
    </xsd:complexType>
</xsd:element>
<xsd:element name="davAdapter">
    <xsd:complexType>
        <xsd:all>
            <xsd:element name="folderURL">
                <xsd:complexType>
                    <xsd:attribute name="value" type="xsd:string" />
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="fileName">
                <xsd:complexType>
                    <xsd:attribute name="value" type="xsd:string" />
                    <xsd:attribute name="valueType" type="xsd:string" />
                </xsd:complexType>
            </xsd:element>
        </xsd:all>
        <xsd:attribute name="name" type="xsd:string" use="required" />
        <xsd:attribute name="overwriteAllowed" type="xsd:boolean" />
        <xsd:attribute name="queryAllowed" type="xsd:boolean" />
        <xsd:attribute name="submitAllowed" type="xsd:boolean" />
    </xsd:complexType>
</xsd:element>
<xsd:element name="emailAdapter">
    <xsd:complexType>
        <xsd:all>
            <xsd:element name="to" minOccurs="0">
                <xsd:complexType>
                    <xsd:attribute name="value" type="xsd:string" />
                    <xsd:attribute name="valueType" type="xsd:string" />
                </xsd:complexType>
            </xsd:element>
        </xsd:all>
    </xsd:complexType>
</xsd:element>

```

```

        </xsd:complexType>
    </xsd:element>
    <xsd:element name="cc" minOccurs="0">
        <xsd:complexType>
            <xsd:attribute name="value" type="xsd:
            <xsd:attribute name="valueType" type='
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="bcc" minOccurs="0">
        <xsd:complexType>
            <xsd:attribute name="value" type="xsd:
            <xsd:attribute name="valueType" type='
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="subject" minOccurs="0">
        <xsd:complexType>
            <xsd:attribute name="value" type="xsd:
            <xsd:attribute name="valueType" type='
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="intro" minOccurs="0">
        <xsd:complexType>
            <xsd:attribute name="value" type="xsd:
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="attachmentFileName" minOc
        <xsd:complexType>
            <xsd:attribute name="value" type="xsd:
            <xsd:attribute name="valueType" type='
        </xsd:complexType>
    </xsd:element>
</xsd:all>
<xsd:attribute name="name" type="xsd:xdTitle" use="re

```

```

        <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" />
        <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" />
    </xsd:complexType>
</xsd:element>
<xsd:element name="dataAdapters">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="xsf:adoAdapter" />
            <xsd:element ref="xsf:webServiceAdapter" />
            <xsd:element ref="xsf:xmlFileAdapter" />
            <xsd:element ref="xsf:sharepointListAdapter" />
            <xsd:element ref="xsf:davAdapter" />
            <xsd:element ref="xsf:emailAdapter" />
            <xsd:element ref="xsf:hwsAdapter" />
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
<!-- documentSchemas -->
<xsd:element name="documentSchemas">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsf:documentSchema" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="documentSchema">
    <xsd:complexType>
        <xsd:attribute name="location" type="xsd:string" use="required" />
        <xsd:attribute name="rootSchema" type="xsf:xdYesNo" />
    </xsd:complexType>
</xsd:element>
<!-- customValidation -->
<xsd:element name="customValidation">

```

```

    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:errorCondition" minOccurs="1" maxOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="errorCondition">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="xsf:errorMessage" />
      </xsd:sequence>
      <xsd:attribute name="match" type="xsd:string" use="required" />
      <xsd:attribute name="expression" type="xsd:string" use="required" />
      <xsd:attribute name="expressionContext" type="xsd:string" use="optional" />
      <xsd:attribute name="showErrorOn" type="xsd:string" use="optional" />
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="errorMessage">
    <xsd:complexType>
      <xsd:simpleContent>
        <xsd:extension base="xsf:xdErrorMessage" />
        <xsd:attribute name="type" use="optional">
          <xsd:simpleType>
            <xsd:restriction base="xsd:NMTOKEN" />
            <xsd:enumeration value="mod" />
            <xsd:enumeration value="mod" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="shortMessage" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="127" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>

```

```

                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
</xsd:element>
<!-- domEventHandlers -->
<xsd:element name="domEventHandlers">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsd:domEventHandler" minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
    </xsd:complexType>
    <xsd:unique name="domEventHandler_handlerObject_uniqueness" />
    <xsd:selector xpath="." />
    <xsd:field xpath="@handlerObject" />
</xsd:unique>
</xsd:element>
<xsd:element name="domEventHandler">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsd:ruleSetAction" minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="dataObject" type="xsd:string" use="optional" />
        <xsd:attribute name="match" type="xsd:string" use="optional" />
        <xsd:attribute name="handlerObject" type="xsd:string" use="optional" />
    </xsd:complexType>
    <xsd:keyref name="domEventHandler_ruleSetAction" refer="xsd:ruleSetAction" />
    <xsd:selector xpath="./xsd:ruleSetAction" />
    <xsd:field xpath="@ruleSet" />
</xsd:keyref>
</xsd:element>

```

```

<!-- importParameters -->
<xsd:element name="importParameters">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:importSource" minOccurs=
</xsd:sequence>
      <xsd:attribute name="enabled" type="xsf:xdYesNo" use="re
      <xsd:attribute name="useScriptHandler" type="xsf:xdY
    </xsd:complexType>
  </xsd:element>
<xsd:element name="importSource">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:string" use="re
    <xsd:attribute name="schema" type="xsf:xdFileName"
    <xsd:attribute name="transform" type="xsf:xdFileNam
    <xsd:attribute name="authoringOfTransform" type="xs
  </xsd:complexType>
</xsd:element>
<!-- listProperties -->
<xsd:element name="listProperties">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="xsf:fields" />
    </xsd:all>
  </xsd:complexType>
</xsd:element>
<xsd:element name="fields">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:field" minOccurs="0" maxC
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

```

```

<xsd:element name="field">
  <xsd:complexType>
    <xsd:attribute name="type" type="xsd:NMTOKEN" us
    <xsd:attribute name="name" type="xsf:xdTitle" use="re
    <xsd:attribute name="columnName" type="xsf:xdTitle'
    <xsd:attribute name="required" type="xsf:xdYesNo" us
    <xsd:attribute name="viewable" type="xsf:xdYesNo" u
    <xsd:attribute name="node" type="xsd:string" use="rec
    <xsd:attribute name="maxLength" type="xsd:byte" />
    <xsd:attribute name="aggregation" use="optional">
      <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
          <xsd:enumeration value="sum" />
          <xsd:enumeration value="count" />
          <xsd:enumeration value="average" />
          <xsd:enumeration value="min" />
          <xsd:enumeration value="max" />
          <xsd:enumeration value="first" />
          <xsd:enumeration value="last" />
          <xsd:enumeration value="merge" />
          <xsd:enumeration value="plaintext" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>
<!-- submit -->
<xsd:element name="submit">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="submitAction" minOccurs="
        <xsd:complexType>
          <xsd:attribute name="adapter" type="xsf

```

```

        </xsd:complexType>
        <xsd:keyref name="submitAdapter_name_ke
            <xsd:selector xpath="." />
            <xsd:field xpath="@adapter" />
        </xsd:keyref>
    </xsd:element>
    <xsd:element ref="xsf:useHttpHandler" minOccurs="1" maxOccurs="1" />
    <xsd:element ref="xsf:useScriptHandler" minOccurs="1" maxOccurs="1" />
    <xsd:element ref="xsf:ruleSetAction" minOccurs="1" maxOccurs="1" />
    <xsd:element ref="xsf:useQueryAdapter" minOccurs="1" maxOccurs="1" />
    <xsd:element ref="xsf:webServiceAdapter" minOccurs="1" maxOccurs="1" />
    <xsd:element ref="xsf:davAdapter" minOccurs="1" maxOccurs="1" />
    <xsd:element ref="xsf:emailAdapter" minOccurs="1" maxOccurs="1" />
    <xsd:element name="successMessage" type="xsd:string" />
    <xsd:element name="errorMessage" type="xsd:string" />
</xsd:all>
<xsd:attribute name="caption" type="xsd:string" use="optional" />
<xsd:attribute name="onAfterSubmit" use="optional" />
<xsd:simpleType>
    <xsd:restriction base="xsd:NMTOKEN">
        <xsd:enumeration value="close" />
        <xsd:enumeration value="keepOpen" />
        <xsd:enumeration value="openNew" />
    </xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="showStatusDialog" type="xsd:boolean" />
<xsd:attribute name="showSignatureReminder" type="xsd:boolean" />
<xsd:attribute name="disableMenuItem" type="xsd:boolean" />
</xsd:complexType>
<xsd:keyref name="submit_ruleSetAction" refer="xsf:ruleSetAction" />
<xsd:selector xpath="./xsf:ruleSetAction" />
<xsd:field xpath="@ruleSet" />

```

```

        </xsd:keyref>
    </xsd:element>
    <xsd:element name="useHttpHandler">
        <xsd:complexType>
            <xsd:attribute name="method" use="required">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:NMTOKEN">
                        <xsd:enumeration value="POST" />
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
            <xsd:attribute name="href" type="xsd:anyURI" use="re
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="useScriptHandler" />
    <xsd:element name="useQueryAdapter" />
    <!-- onLoad -->
    <xsd:element name="onLoad">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="xsf:ruleSetAction" minOccurs=
            </xsd:sequence>
        </xsd:complexType>
        <xsd:keyref name="load_ruleSetAction" refer="xsf:ruleset_1
            <xsd:selector xpath="./xsf:ruleSetAction" />
            <xsd:field xpath="@ruleSet" />
        </xsd:keyref>
    </xsd:element>
    <!-- save -->
    <xsd:element name="save">
        <xsd:complexType>
            <xsd:choice minOccurs="0" maxOccurs="1">
                <xsd:element ref="xsf:useScriptHandler" />
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>

```

```

        </xsd:choice>
    </xsd:complexType>
</xsd:element>
<!-- roles -->
<xsd:element name="roles">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsf:role" minOccurs="1" maxOccurs="1" />
            <xsd:element ref="xsf:membership" minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="default" type="xsd:string" use="optional" />
        <xsd:attribute name="initiator" type="xsd:string" use="optional" />
        <xsd:attribute name="hideStatusBarDisplay" type="xsd:boolean" use="optional" />
    </xsd:complexType>
    <!-- role names must be unique -->
    <xsd:unique name="roles_name_unique">
        <xsd:selector xpath="./xsf:role" />
        <xsd:field xpath="@name" />
    </xsd:unique>
    <!-- fields must reference existing role -->
    <xsd:key name="role_name_key">
        <xsd:selector xpath="./xsf:role" />
        <xsd:field xpath="@name" />
    </xsd:key>
    <xsd:keyref name="role_default" refer="xsf:role_name_key">
        <xsd:selector xpath="." />
        <xsd:field xpath="@default" />
    </xsd:keyref>
    <xsd:keyref name="role_initiator" refer="xsf:role_name_key">
        <xsd:selector xpath="." />
        <xsd:field xpath="@initiator" />
    </xsd:keyref>
    <xsd:keyref name="role_membership" refer="xsf:role_name_key">
        <xsd:selector xpath="." />
        <xsd:field xpath="@membership" />
    </xsd:keyref>

```

```

        <xsd:selector xpath="./xsf:membership/*" />
        <xsd:field xpath="@memberOf" />
    </xsd:keyref>
</xsd:element>
<xsd:element name="role">
    <xsd:complexType>
        <xsd:attribute name="name" type="xsf:xdRoleName" u
    </xsd:complexType>
</xsd:element>
<xsd:element name="membership">
    <xsd:complexType>
        <xsd:choice minOccurs="1" maxOccurs="unbounded">
            <xsd:element ref="xsf:getUserNameFromData" />
            <xsd:element ref="xsf:userName" />
            <xsd:element ref="xsf:group" />
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
<xsd:element name="getUserNameFromData">
    <xsd:complexType>
        <xsd:attribute name="dataObject" type="xsd:string" use="re
        <xsd:attribute name="select" type="xsd:string" use="re
        <xsd:attribute name="memberOf" type="xsd:string" use="re
    </xsd:complexType>
</xsd:element>
<xsd:element name="userName">
    <xsd:complexType>
        <xsd:attribute name="name" type="xsd:string" use="re
        <xsd:attribute name="memberOf" type="xsd:string" use="re
    </xsd:complexType>
</xsd:element>
<xsd:element name="group">
    <xsd:complexType>

```

```

        <xsd:attribute name="name" type="xsd:string" use="required"/>
        <xsd:attribute name="memberOf" type="xsd:string" use="optional"/>
    </xsd:complexType>
</xsd:element>
<!-- hwsWorkflow -->
<xsd:element name="hwsWorkflow">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsf:location" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="xsf:allowedActions" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="xsf:allowedTasks" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute name="taskpaneVisible" type="xsd:boolean" use="optional"/>
    </xsd:complexType>
    <xsd:unique name="hws_actiontask_name">
        <xsd:selector xpath="./xsf:allowedActions/xsf:action|./xsf:allowedTasks/xsf:task"/>
        <xsd:field xpath="@name" />
    </xsd:unique>
</xsd:element>
<!-- location -->
<xsd:element name="location">
    <xsd:complexType>
        <xsd:attribute name="url" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
<!-- allowedActions -->
<xsd:element name="allowedActions">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsf:action" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:unique name="hws_actionTypeID_unique">

```

```

        <xsd:selector xpath="./xf:action" />
        <xsd:field xpath="@actionTypeID" />
    </xsd:unique>
</xsd:element>
<!-- action -->
<xsd:element name="action">
    <xsd:complexType>
        <xsd:attribute name="name" type="xf:xdHWSname" u
        <xsd:attribute name="actionTypeID" type="xsd:string"
        <xsd:attribute name="canInitiateWorkflow" type="xf:
        <xsd:attribute name="caption" type="xf:xdHWSCapti
    </xsd:complexType>
</xsd:element>
<!-- allowedTasks -->
<xsd:element name="allowedTasks">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xf:task" minOccurs="1" maxC
        </xsd:sequence>
    </xsd:complexType>
    <xsd:unique name="hws_taskID_unique">
        <xsd:selector xpath="./xf:task" />
        <xsd:field xpath="@taskTypeID" />
    </xsd:unique>
</xsd:element>
<!-- task -->
<xsd:element name="task">
    <xsd:complexType>
        <xsd:attribute name="name" type="xf:xdHWSname" u
        <xsd:attribute name="taskTypeID" type="xsd:string" us
        <xsd:attribute name="caption" type="xf:xdHWSCapti
    </xsd:complexType>
</xsd:element>

```

```

<!-- fileNew -->
<xsd:element name="fileNew">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:initialXmlDocument" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="initialXmlDocument">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:customCategory" minOccurs="1" maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="caption" type="xsf:xdTitle" use="required" />
    <xsd:attribute name="href" type="xsf:xdFileName" use="required" />
  </xsd:complexType>
</xsd:element>
<!-- customCategory -->
<xsd:element name="customCategory">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required" />
  </xsd:complexType>
</xsd:element>
<!-- package -->
<xsd:element name="package">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:files" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="files">
  <xsd:complexType>

```

```

        <xsd:sequence>
            <xsd:element ref="xsf:file" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="file">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsf:fileProperties" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="name" type="xsf:xdFileName" use="required" />
    </xsd:complexType>
</xsd:element>
<xsd:element name="fileProperties">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsf:property" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="property">
    <xsd:complexType>
        <xsd:attribute name="name" type="xsd:string" use="required" />
        <xsd:attribute name="value" type="xsd:string" use="required" />
        <xsd:attribute name="type" type="xsd:QName" use="required" />
    </xsd:complexType>
</xsd:element>
<!-- permissions -->
<xsd:element name="permissions">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="xsf:allowedControl" />
        </xsd:choice>
    </xsd:complexType>
</xsd:element>

```

```

        </xsd:complexType>
    </xsd:element>
    <xsd:element name="allowedControl">
        <xsd:complexType>
            <xsd:attribute name="cabFile" type="xsd:string" use="o
            <xsd:attribute name="clsid" type="xsd:string" use="req
            <xsd:attribute name="version" type="xsd:string" use="o
        </xsd:complexType>
    </xsd:element>
    <!-- View and Context-Driven Editing definitions -->
    <!-- External Views -->
    <xsd:element name="externalViews">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="xsf:externalView" minOccurs="1
            </xsd:sequence>
            <xsd:attribute name="default" type="xsd:string" />
        </xsd:complexType>
        <xsd:unique name="externalViews_name_unique">
            <xsd:selector xpath="./xsf:externalView" />
            <xsd:field xpath="@default" />
        </xsd:unique>
        <xsd:keyref name="external_views_printView" refer="xsf:e:
            <xsd:selector xpath="." />
            <xsd:field xpath="@default" />
        </xsd:keyref>
    </xsd:element>
    <xsd:element name="externalView">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="xsf:mainpane" />
            </xsd:sequence>
            <xsd:attribute name="target" type="xsd:string" />

```

```

        <xsd:attribute name="name" type="xsf:xdViewName" />
        <xsd:attribute name="designMode" type="xsf:xdDesign" />
    </xsd:complexType>
</xsd:element>
<!-- attributeData -->
<xsd:element name="attributeData">
    <xsd:complexType>
        <xsd:attribute name="attribute" type="xsd:string" use="required" />
        <xsd:attribute name="value" type="xsd:string" use="required" />
    </xsd:complexType>
</xsd:element>
<!-- button -->
<xsd:element name="button">
    <xsd:complexType>
        <xsd:attribute name="caption" type="xsf:xdTitle" />
        <xsd:attribute name="icon" type="xsd:string" />
        <xsd:attribute name="tooltip" type="xsf:xdTitle" />
        <xsd:attribute name="name" type="xsd:NMTOKEN" />
        <xsd:attribute name="xmlToEdit" type="xsd:NMTOKEN" />
        <xsd:attribute name="action">
            <xsd:simpleType>
                <xsd:restriction base="xsd:NMTOKEN">
                    <xsd:enumeration value="xCollection::insert" />
                    <xsd:enumeration value="xCollection::insertAfter" />
                    <xsd:enumeration value="xCollection::insertBefore" />
                    <xsd:enumeration value="xCollection::replace" />
                    <xsd:enumeration value="xCollection::replaceAfter" />
                    <xsd:enumeration value="xCollection::replaceBefore" />
                    <xsd:enumeration value="xOptional::insert" />
                    <xsd:enumeration value="xOptional::replace" />
                    <xsd:enumeration value="xReplace::replace" />
                    <xsd:enumeration value="xFileAttachment" />
                    <xsd:enumeration value="xFileAttachment" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:complexType>
</xsd:element>

```

```

        <xsd:enumeration value="xFileAttachme
        <xsd:enumeration value="xFileAttachme
    </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="showIf">
    <xsd:simpleType>
        <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="always" />
            <xsd:enumeration value="enabled" />
            <xsd:enumeration value="immediate" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
<!-- chooseFragment -->
<xsd:element name="chooseFragment">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:any minOccurs="0" maxOccurs="unbounded
        </xsd:sequence>
        <xsd:attribute name="parent" type="xsd:string" />
        <xsd:attribute name="followingSiblings" type="xsd:stri
        <xsd:attribute name="innerFragment" type="xsd:string'
    </xsd:complexType>
</xsd:element>
<!-- editWith -->
<xsd:element name="editWith">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsf:masterDetail" minOccurs="
            <xsd:element ref="xsf:fragmentToInsert" minOccu

```

```

</xsd:sequence>
<xsd:attribute name="component" use="required">
  <xsd:simpleType>
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="xCollection" />
      <xsd:enumeration value="xOptional" />
      <xsd:enumeration value="xReplace" />
      <xsd:enumeration value="xTextList" />
      <xsd:enumeration value="xField" />
      <xsd:enumeration value="xImage" />
      <xsd:enumeration value="xFileAttachment" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="caption" type="xsf:xdTitle" use="required">
<xsd:attribute name="autoComplete" type="xsf:xdYesNo" use="optional">
<xsd:attribute name="proofing" type="xsf:xdYesNo" use="optional">
<xsd:attribute name="type" use="optional">
  <xsd:simpleType>
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="plain" />
      <xsd:enumeration value="formatted" />
      <xsd:enumeration value="plainMultiline" />
      <xsd:enumeration value="formattedMultiline" />
      <xsd:enumeration value="rich" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="useFilter" use="optional">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="yes" />
      <xsd:enumeration value="no" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>

```

```

        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="widgetIcon" use="optional">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="standard" />
            <xsd:enumeration value="filter" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="filterDependency" type="xsd:string" use="optional" />
<xsd:attribute name="field" type="xsd:string" use="optional" />
<xsd:attribute name="removeAncestors" type="xsd:boolean" use="optional" />
<xsd:attribute name="maxLength" use="optional">
    <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
            <xsd:minInclusive value="-1" />
            <xsd:maxInclusive value="9999" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="allowedFileTypes" type="xsd:string" use="optional" />
<xsd:anyAttribute namespace="http://schemas.microsoft.com/office/word/2003/wordml" />
</xsd:complexType>
</xsd:element>
<!-- unboundControls -->
<xsd:element name="unboundControls">
    <xsd:complexType>
        <xsd:sequence>
            <!-- button -->
            <xsd:element name="button" minOccurs="0" maxOccurs="1">
                <xsd:complexType>

```

```

        <xsd:sequence>
            <xsd:element ref="xsd:ruleSetActio
        </xsd:sequence>
        <xsd:attribute name="name" use="requi
            <xsd:simpleType>
                <!-- type of name is non qualif
                so these characters are  disabl
                <xsd:restriction base="xsd:NC
                    <xsd:pattern value="[^\.\|
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:complexType>
    <xsd:keyref name="button_ruleSetAction" re
        <xsd:selector xpath="./xsd:ruleSetAction
        <xsd:field xpath="@ruleSet" />
    </xsd:keyref>
</xsd:element>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>
<!-- editing -->
<xsd:element name="editing">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsd:xmlToEdit" minOccurs="0"
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<!-- Master Detail -->
<xsd:element name="masterDetail">
    <xsd:complexType>
        <xsd:attribute name="master" type="xsd:string" />

```

```

        <xsd:attribute name="masterViewContext" type="xsd:s
        <xsd:attribute name="masterKey" type="xsd:string" />
        <xsd:attribute name="detailKey" type="xsd:string" />
    </xsd:complexType>
</xsd:element>
<!-- fragmentToInsert -->
<xsd:element name="fragmentToInsert">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsf:chooseFragment" minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<!-- mainpane -->
<xsd:element name="mainpane">
    <xsd:complexType>
        <xsd:attribute name="transform" type="xsf:xdFileName" />
    </xsd:complexType>
</xsd:element>
<!-- printSettings -->
<xsd:element name="printSettings">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsf:header" minOccurs="0" maxOccurs="1" />
            <xsd:element ref="xsf:footer" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="orientation">
            <xsd:simpleType>
                <xsd:restriction base="xsd:NMTOKEN">
                    <xsd:enumeration value="portrait" />
                    <xsd:enumeration value="landscape" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:complexType>
</xsd:element>

```

```
</xsd:attribute>
<xsd:attribute name="header">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="255" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="footer">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="255" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="marginUnitsType">
  <xsd:simpleType>
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="in" />
      <xsd:enumeration value="cm" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="rightMargin">
  <xsd:simpleType>
    <xsd:restriction base="xsd:float">
      <xsd:minInclusive value="0" />
      <xsd:maxInclusive value="100" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="leftMargin">
  <xsd:simpleType>
```

```
        <xsd:restriction base="xsd:float">
            <xsd:minInclusive value="0" />
            <xsd:maxInclusive value="100" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="topMargin">
    <xsd:simpleType>
        <xsd:restriction base="xsd:float">
            <xsd:minInclusive value="0" />
            <xsd:maxInclusive value="100" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="bottomMargin">
    <xsd:simpleType>
        <xsd:restriction base="xsd:float">
            <xsd:minInclusive value="0" />
            <xsd:maxInclusive value="100" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="printerName">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="255" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="paperSize">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="255" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
```

```
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="paperSource">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="255" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="copies">
    <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
            <xsd:minInclusive value="1" />
            <xsd:maxInclusive value="9999" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="collate" type="xsf:xdYesNo" />
<xsd:attribute name="pageRangeStart">
    <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
            <xsd:minInclusive value="1" />
            <xsd:maxInclusive value="32000" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="pageRangeEnd">
    <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
            <xsd:minInclusive value="1" />
            <xsd:maxInclusive value="32000" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
```

```

        </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="printerSpecificSettings">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:maxLength value="255" />
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>
</xsd:complexType>
</xsd:element>
<xsd:element name="header">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:any minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="footer">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:any minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<!-- toolbar -->
<xsd:element name="toolbar">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:group ref="xsd:UIItem" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string" use="required" />
        <xsd:attribute name="caption" type="xsd:string" use="optional" />
    </xsd:complexType>
</xsd:element>

```

```

        </xsd:complexType>
</xsd:element>
<!-- menu -->
<xsd:element name="menu">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:group ref="xsd:UIItem" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="caption" type="xsd:xdTitle" use="optional" />
    </xsd:complexType>
</xsd:element>
<!-- menuArea -->
<xsd:element name="menuArea">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:group ref="xsd:UIItem" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="name" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:NMTOKEN">
                    <xsd:enumeration value="msoFileMenu" />
                    <xsd:enumeration value="msoEditMenu" />
                    <xsd:enumeration value="msoInsertMenu" />
                    <xsd:enumeration value="msoViewMenu" />
                    <xsd:enumeration value="msoFormatMenu" />
                    <xsd:enumeration value="msoToolsMenu" />
                    <xsd:enumeration value="msoTableMenu" />
                    <xsd:enumeration value="msoHelpMenu" />
                    <xsd:enumeration value="msoStructural" />
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:complexType>

```

```

</xsd:element>
<!-- UIContainer -->
<xsd:group name="UIContainer">
  <xsd:choice>
    <xsd:element ref="xsf:toolbar" />
    <xsd:element ref="xsf:menu" />
    <xsd:element ref="xsf:menuArea" />
  </xsd:choice>
</xsd:group>
<!-- UIItem -->
<xsd:group name="UIItem">
  <xsd:choice>
    <xsd:element ref="xsf:button" />
    <xsd:element ref="xsf:menu" />
  </xsd:choice>
</xsd:group>
<!-- taskpane -->
<xsd:element name="taskpane">
  <xsd:complexType>
    <xsd:attribute name="caption" type="xsd:string" use="optional"/>
    <xsd:attribute name="href" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
<!-- views -->
<xsd:element name="views">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:view" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="default" type="xsd:string" />
  </xsd:complexType>
  <xsd:unique name="views_name_unique">
    <xsd:selector xpath="./xsf:view" />
  </xsd:unique>
</xsd:element>

```

```

        <xsd:field xpath="@name" />
    </xsd:unique>
    <xsd:keyref name="view_printView" refer="xsf:view_or_ex
        <xsd:selector xpath="./xsf:view" />
        <xsd:field xpath="@printView" />
    </xsd:keyref>
    <xsd:keyref name="views_default" refer="xsf:view_name_k
        <xsd:selector xpath="." />
        <xsd:field xpath="@default" />
    </xsd:keyref>
</xsd:element>
<!-- ViewContent -->
<xsd:group name="ViewContent">
    <xsd:choice>
        <xsd:element ref="xsf:editing" minOccurs="0" />
        <xsd:element ref="xsf:mainpane" minOccurs="0" />
        <xsd:element ref="xsf:printSettings" minOccurs="0" />
        <xsd:group ref="xsf:UIContainer" minOccurs="0" max
        <xsd:element ref="xsf:unboundControls" minOccurs="0" />
    </xsd:choice>
</xsd:group>
<!-- view -->
<xsd:element name="view">
    <xsd:complexType>
        <xsd:group ref="xsf:ViewContent" minOccurs="0" max
        <xsd:attribute name="caption" type="xsf:xdViewName" />
        <xsd:attribute name="name" type="xsf:xdViewName" />
        <xsd:attribute name="showMenuItem" type="xsf:xdYesNo" />
        <xsd:attribute name="printView" type="xsd:string" />
        <xsd:attribute name="designMode" type="xsf:xdDesign" />
    </xsd:complexType>
    <xsd:unique name="toolbar_name_unique">
        <xsd:selector xpath="./xsf:toolbar" />

```

```

        <xsd:field xpath="@name" />
    </xsd:unique>
    <xsd:unique name="menuArea_name_unique">
        <xsd:selector xpath="./xsf:menuArea" />
        <xsd:field xpath="@name" />
    </xsd:unique>
    <xsd:unique name="xmlToEdit_name_unique">
        <xsd:selector xpath="./xsf:editing/xsf:xmlToEdit" />
        <xsd:field xpath="@name" />
    </xsd:unique>
    <xsd:key name="xmlToEdit_name_key">
        <xsd:selector xpath="./xsf:editing/xsf:xmlToEdit" />
        <xsd:field xpath="@name" />
    </xsd:key>
    <xsd:keyref name="button_xmlToEdit_reference" refer="xsf:xmlToEdit_name_key">
        <xsd:selector xpath="./xsf:menuArea/xsf:button | ./xsf:editing/xsf:xmlToEdit" />
        <xsd:field xpath="@xmlToEdit" />
    </xsd:keyref>
</xsd:element>
<!-- xmlToEdit -->
<xsd:element name="xmlToEdit">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsf:editWith" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:NMTOKEN" use="required" />
        <xsd:attribute name="item" type="xsd:string" use="required" />
        <xsd:attribute name="container" type="xsd:string" />
        <xsd:attribute name="viewContext">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:pattern value="((\.\|#[a-zA-Z0-9_])|<\/xsd:restriction>

```

```

        </xsd:simpleType>
    </xsd:attribute>
</xsd:complexType>
</xsd:element>
<!-- Digital Signatures -->
<xsd:element name="documentSignatures">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsf:signedDataBlock" minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="signatureLocation" type="xsd:string" />
    </xsd:complexType>
</xsd:element>
<xsd:element name="signedDataBlock">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="message" type="xsf:xdSignedDataBlock" />
        </xsd:sequence>
        <xsd:attribute name="name" type="xsf:xdSignedDataBlockName" />
        <xsd:attribute name="data" type="xsd:string" use="required" />
        <xsd:attribute name="signatureLocation" type="xsd:string" />
        <xsd:attribute name="mode" type="xsf:xdSignatureRelationship" />
    </xsd:complexType>
    <xsd:unique name="signedDataBlock_name_unique">
        <xsd:selector xpath="." />
        <xsd:field xpath="@name" />
    </xsd:unique>
</xsd:element>
<!-- Version Upgrade -->
<xsd:element name="documentVersionUpgrade">
    <xsd:complexType>
        <xsd:choice>
            <xsd:element ref="xsf:useScriptHandler" />

```

```

        <xsd:element ref="xsf:useTransform" />
    </xsd:choice>
</xsd:complexType>
</xsd:element>
<xsd:element name="useTransform">
    <xsd:complexType>
        <xsd:attribute name="transform" use="required">
            <xsd:simpleType>
                <xsd:union memberTypes="xsf:xdFileName xsf:xdFileContent" />
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="minVersionToUpgrade" type="xsd:string" />
        <xsd:attribute name="maxVersionToUpgrade" type="xsd:string" />
    </xsd:complexType>
</xsd:element>
<!-- XSF Extensions -->
<xsd:element name="extensions">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsf:extension" minOccurs="0" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="extension">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:any minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:NMTOKEN" />
        <xsd:anyAttribute processContents="skip" />
    </xsd:complexType>
</xsd:element>
<!-- Rules -->

```

```

<xsd:element name="ruleSetAction">
  <xsd:complexType>
    <xsd:attribute name="ruleSet" type="xsd:string" use="r
  </xsd:complexType>
</xsd:element>
<xsd:element name="rule">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:choice minOccurs="0" maxOccurs="unbound"
        <xsd:element ref="xsf:dialogBoxMessageAct
        <xsd:element ref="xsf:dialogBoxExpressionA
        <xsd:element ref="xsf:switchViewAction" />
        <xsd:element ref="xsf:assignmentAction" />
        <xsd:element ref="xsf:queryAction" />
        <xsd:element name="submitAction">
          <xsd:complexType>
            <xsd:attribute name="adapter" type
          </xsd:complexType>
        </xsd:element>
        <xsd:element ref="xsf:openNewDocumentAc
        <xsd:element ref="xsf:closeDocumentAction
      </xsd:choice>
      <xsd:element name="exitRuleSet" minOccurs="0"
        <xsd:complexType />
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="caption" type="xsd:string" use="r
    <xsd:attribute name="condition" type="xsd:string" use=
    <xsd:attribute name="isEnabled" type="xsf:xdYesNo" u
  </xsd:complexType>
</xsd:element>
<xsd:element name="dialogBoxMessageAction">
  <xsd:simpleType>

```

```

        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="1024" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="dialogBoxExpressionAction" type="xsd:string"/>
<xsd:element name="switchViewAction">
    <xsd:complexType>
        <xsd:attribute name="view" type="xsd:string" use="required"/>
    </xsd:complexType>
    <xsd:keyref name="switchViewAction_view_keyref" refer="view">
        <xsd:selector xpath="." />
        <xsd:field xpath="@view" />
    </xsd:keyref>
</xsd:element>
<xsd:element name="assignmentAction">
    <xsd:complexType>
        <xsd:attribute name="targetField" type="xsd:string" use="required"/>
        <xsd:attribute name="expression" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="queryAction">
    <xsd:complexType>
        <xsd:attribute name="adapter" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="openNewDocumentAction">
    <xsd:complexType>
        <xsd:attribute name="solutionURI" type="xsd:anyURI" use="required"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="closeDocumentAction">
    <xsd:complexType>

```

```

        <xsd:attribute name="promptToSaveChanges" type="xs
    </xsd:complexType>
</xsd:element>
<xsd:element name="ruleSet">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsf:rule" minOccurs="1" maxO
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string" use="re
    </xsd:complexType>
</xsd:element>
<xsd:element name="ruleSets">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsf:ruleSet" minOccurs="0" ma
        </xsd:sequence>
    </xsd:complexType>
    <xsd:unique name="ruleSets_name_unique">
        <xsd:selector xpath="./xsf:ruleSet" />
        <xsd:field xpath="@name" />
    </xsd:unique>
</xsd:element>
<!-- Declarative Calculations -->
<xsd:element name="calculations">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xsf:calculatedField" minOccurs
        </xsd:sequence>
        <xsd:attribute name="treatBlankValueAsZero" type="x
    </xsd:complexType>
</xsd:element>
<xsd:element name="calculatedField">
    <xsd:complexType>

```

```
<xsd:attribute name="target" type="xsd:string" use="re
<xsd:attribute name="expression" type="xsd:string" use
<xsd:attribute name="refresh" type="xsd:string" use="r
</xsd:complexType>
</xsd:element>
</xsd:schema>
```


xsf Namespace

The Microsoft Office InfoPath 2003 [form definition \(.xsf\) file](#) conforms to an XSD schema definition, and uses a corresponding [namespace](#).

Namespace URI

xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solutio

Remarks

All of the elements within the .xsf file are namespace qualified using the **xsf** namespace.

xdDesignMode Type

Specifies whether a [view](#) (*.xsl) file can be opened in design mode in Microsoft Office InfoPath 2003.

Type

xsd:NMTOKEN

Facets

| Name | Description |
|-------------|-------------|
| enumeration | normal |
| enumeration | protected |

Remarks

The **xdDesignMode** type is used for attributes in the [form definition file \(.xsf\)](#) to specify whether a view can be opened in design mode ("normal"), or whether the view is not permitted to be opened in design mode ("protected").

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following example is the declaration of the **xdDesignMode** type:

```
<xsd:simpleType name="xdDesignMode" >  
  <xsd:restriction base="xsd:NMTOKEN">  
    <xsd:enumeration value="normal" />  
    <xsd:enumeration value="protected" />  
  </xsd:restriction>  
</xsd:simpleType>
```


xdEmptyString Type

Specifies an empty string.

Type

xsd:string

Facets

| Name | Description |
|-------------|-------------|
| maxLength 0 | |

Remarks

The **xdEmptyString** type is used for attributes in the [form definition \(.xsf\) file](#) that specify an empty string.

Example

The following example is the declaration of the **xdEmptyString** type:

```
<xsd:simpleType name="xdEmptyString">  
  <xsd:restriction base="xsd:string">  
    <xsd:maxLength value="0" />  
  </xsd:restriction>  
</xsd:simpleType>
```

xdEnabledDisabled Type

Specifies whether a feature is enabled or disabled.

Type

xsd:NMTOKEN

Facets

| Name | Description |
|-------------|-------------|
| enumeration | enabled |
| enumeration | disabled |

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following example is the declaration of the **xdEnabledDisabled** type:

```
<xsd:simpleType name="xdEnabledDisabled">  
  <xsd:restriction base="xsd:NMTOKEN">  
    <xsd:enumeration value="enabled" />  
    <xsd:enumeration value="disabled" />  
  </xsd:restriction>  
</xsd:simpleType>
```


xdErrorMessage Type

Specifies an error message.

Type

xsd:string

Facets

| Name | Description |
|-----------|-------------|
| maxLength | 1023 |

Remarks

The **xdErrorMessage** type is used for attributes in the [form definition \(.xsf\) file](#) that specify an error message no greater than 1023 characters in length.

Example

The following example is the declaration of the **xdErrorMessage** type:

```
<xsd:simpleType name="xdErrorMessage">  
  <xsd:restriction base="xsd:string">  
    <xsd:maxLength value="1023" />  
  </xsd:restriction>  
</xsd:simpleType>
```


xdExpressionLiteral simpleType

Specifies a type for identifying whether a value should be interpreted as a literal value or an [XPath](#) expression that returns a value.

Type

xsd:NMTOKEN

Remarks

The **xdExpressionLiteral** type is used for the **valueType** attribute in the [davAdapter](#) and [emailAdapter](#) elements.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following example is the declaration of the **xdExpressionLiteral** type:

```
<xsd:simpleType name="xdExpressionLiteral">  
  <xsd:restriction base="xsd:NMTOKEN">  
    <xsd:enumeration value="expression" />  
    <xsd:enumeration value="literal" />  
  </xsd:restriction>  
</xsd:simpleType>
```


xdFileName Type

Specifies the name of a file.

Type

xsd:string

Facets

| Name | Description |
|-----------|-------------|
| minLength | 1 |
| maxLength | 64 |

Remarks

The **xdFileName** type is used for attributes in the [form definition \(.xsf\) file](#) that specify a file name that can be from 1 to 64 characters in length.

Example

The following example is the declaration of the **xdFileName** type:

```
<xsd:simpleType name="xdFileName">  
  <xsd:restriction base="xsd:string">  
    <xsd:minLength value="1" />  
    <xsd:maxLength value="64" />  
  </xsd:restriction>  
</xsd:simpleType>
```

xdHWSCaption simpleType

Specifies a type for the caption of the Microsoft Biztalk Server 2004 HWS (Human Workflow Services) [action](#) or [task](#) element.

Type

xsd:string

Facets

| Name | Description |
|-----------|----------------|
| maxLength | 255 characters |
| minLength | 1 character |

Remarks

The length of the caption cannot be less than one character or greater than 255 characters.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following example is the declaration of the **xdHWSCaption** type:

```
<xsd:simpleType name="xdHWSCaption" >  
  <xsd:restriction base="xsd:string">  
    <xsd:minLength value="1" />  
    <xsd:maxLength value="255" />  
  </xsd:restriction>  
</xsd:simpleType>
```

xdHWSname simpleType

Specifies a type for a unique name of the Microsoft Biztalk Server 2004 HWS (Human Workflow Services) [action](#) or [task](#) element as specified by the HWS workflow designer.

Type

xsd:NCName

Facets

| Name | Description |
|---------|---|
| pattern | [^-\\.^\\ \\[\\^\\ \\+^?*^@\\{\\}\\(\\)^>^<^=^;^,]* |

Remarks

The name cannot contain the following characters:

\/" [] : < > + = ; , ? * @

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

xdManualAuto Type

Specifies a "manual" or "automatic" value.

Type

xsd:NMTOKEN

Facets

| Name | Description |
|-------------|-------------|
| enumeration | manual |
| enumeration | automatic |

Remarks

The **xdTrustManualAuto** type is used for attributes in the [form definition \(.xsf\) file](#) that require a "manual" or "automatic" value. The default value is "manual".

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following example is the declaration of the **xdManualAuto** type:

```
<xsd:simpleType name="xdManualAuto" >  
  <xsd:restriction base="xsd:NMTOKEN">  
    <xsd:enumeration value="manual" />  
    <xsd:enumeration value="automatic" />  
  </xsd:restriction>  
</xsd:simpleType>
```

xdRoleName Type

Specifies the role name.

Type

xsf:xdViewName

Remarks

Role name has restrictions on values it can contain. These restrictions are the same as the constraining facets of the [xsf:xdViewName](#) type.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following example is the declaration of the **xdRoleName** type:

```
<xsd:simpleType name="xdRoleName" >  
  <xsd:restriction base="xsf:xdViewName"></xsd:restriction>  
</xsd:simpleType>
```


xdScriptLanguage Type

Specifies the name of a scripting language.

Type

xsd:NMTOKEN

Facets

| Name | Description |
|---------|---|
| pattern | ((([Jj][Aa][Vv][Aa]) ([Jj]) ([Vv][Bb])) ([Ss][Cc][Rr][Ii][Pp][Tt])) |

Remarks

The **xdScriptLanguage** type is used for attributes in the [form definition \(.xsf\) file](#) that specify the name of a scripting language.

Example

The following example is the declaration of the **xdScriptLanguage** type:

```
<xsd:simpleType name="xdScriptLanguage">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:pattern value="((([Jj][Aa][Vv][Aa])|([Jj])|([Vv][Bb]))
      ([Ss][Cc][Rr][Ii][Pp][Tt]))" />
  </xsd:restriction>
</xsd:simpleType>
```

xdSignatureRelationEnum Type

Contains the signature relation enumeration for the [signedDataBlock](#) element.

Type

xsd:string

Facets

| Name | Description |
|-------------|--|
| countersign | Specifies that each signature signs the previous signature in the signedDataBlock . |
| cosign | Specifies that each signature in the signedDataBlock is independent. |
| single | Specifies that only one signature is allowed for the signedDataBlock . |

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following example is the declaration of the **xdSignatureRelationEnum** type:

```
<xsd:simpleType name="xdSignatureRelationEnum" >  
  <xsd:restriction base="xsd:string">  
    <xsd:enumeration value="countersign" />  
    <xsd:enumeration value="cosign" />  
    <xsd:enumeration value="single" />  
  </xsd:restriction>  
</xsd:simpleType>
```

xdSignedDataBlockMessage Type

Defines the maximum string length for the signatures confirmation [message](#) element.

Type

xsd:simpleType

Facets

| Name | Description |
|-----------|----------------|
| maxLength | 255 characters |

Remarks

The signature confirmation message has a maximum length of 255 characters.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following example is the declaration of the **xdSignedDataBlockMessage** type:

```
<xsd:simpleType name="xdSignedDataBlockMessage">  
  <xsd:restriction base="xsd:string">  
    <xsd:maxLength value="255" />  
  </xsd:restriction>  
</xsd:simpleType>
```

xdSignedDataBlockName Type

Defines the maximum string length for the [name](#) attribute of the [signedDataBlock](#) element.

Type

xsd:simpleType

Facets

| Name | Description |
|-----------|----------------|
| maxLength | 255 characters |
| minLength | 1 character |

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following example is the declaration of the **xdSignedDataBlockName** type:

```
<xsd:simpleType name="xdSignedDataBlockName">  
  <xsd:restriction base="xsd:ID">  
    <xsd:minLength value="1" />  
    <xsd:maxLength value="255" />  
  </xsd:restriction>  
</xsd:simpleType>
```


xdSolutionVersion Type

Specifies a version number.

Type

xsd:string

Facets

| Name | Description |
|---------|-------------------------------|
| pattern | ((([0-9]{1,4}.){3}[0-9]{1,4}) |

Remarks

The **xdSolutionVersion** type is used for attributes in the [form definition \(.xsf\) file](#) that specify a version number.

Example

The following example is the declaration of the **xdSolutionVersion** type:

```
<xsd:simpleType name="xdSolutionVersion">  
  <xsd:restriction base="xsd:string">  
    <xsd:pattern value="([0-9]{1,4}.){3}[0-9]{1,4}" />  
  </xsd:restriction>  
</xsd:simpleType>
```


xdTitle Type

Specifies a title string.

Type

xsd:string

Facets

| Name | Description |
|-----------|--|
| minLength | 1 |
| maxLength | 255 |
| pattern | $([\^p\{Z\}\p\{Cc}\p\{Cf}\p\{Cn}])(([\^p\{Zl}\p\{Zp}\p\{Cc}]^*([\^p\{Z}\p\{Cc}\p\{Cf}\p\{Cn}])))?$ |

Remarks

The **xdTitle** type is used for attributes in the [form definition \(.xsf\) file](#) that are a string of characters that can be from 1 to 255 characters in length, and that follow a prescribed pattern.

Example

The following example is the declaration of the **xdTitle** type:

```
<xsd:simpleType name="xdTitle">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1" />
    <xsd:maxLength value="255" />
    <xsd:pattern value="([\p{Z}\p{Cc}\p{Cf}\p{Cn}])
      (([\p{Zl}\p{Zp}\p{Cc}])*([\p{Z}\p{Cc}\p{Cf}\p{Cn}])))?" />
  </xsd:restriction>
</xsd:simpleType>
```


xdTrustLevel Type

Specifies a "restricted" or "domain" value.

Type

xsd:string

Facets

| Name | Description |
|-------------|-------------|
| enumeration | restricted |
| enumeration | domain |

Remarks

The **xdTrustLevel** type is used for attributes in the [form definition \(.xsf\) file](#) that require a "restricted" or "domain" value. The default value is "domain".

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following example is the declaration of the **xdTrustLevel** type:

```
<xsd:simpleType name="xdTrustLevel">  
  <xsd:restriction base="xsd:string">  
    <xsd:enumeration value="restricted" />  
    <xsd:enumeration value="domain" />  
  </xsd:restriction>  
</xsd:simpleType>
```


xdViewName Type

Specifies a view name.

Type

xsd:string

Remarks

The **xdViewName** type is used for attributes in the [form definition \(.xsf\) file](#) that are a string of characters that can be from 1 to 255 characters in length, and that follow a prescribed pattern.

Example

The following example is the declaration of the **xdViewName** type:

```
<xsd:simpleType name="xdViewName">
  <xsd:restriction base="xsd:string">
    <xsd:minLength value="1" />
    <xsd:maxLength value="255" />
    <xsd:pattern value="([\p{Z}\p{C}^\#&quot;&gt;&lt;])
      (([\p{Zl}\p{Zp}\p{C}^\#&quot;&gt;&lt;])*([\p{Z}
        \p{C}^\#&quot;&gt;&lt;]))?" />
  </xsd:restriction>
</xsd:simpleType>
```


xdYesNo Type

Specifies a yes or no value.

Type

xsd:NMTOKEN

Facets

| Name | Description |
|-------------|-------------|
| enumeration | yes |
| enumeration | no |

Remarks

The **xdYesNo** type is used for attributes in the [form definition \(.xsf\) file](#) that require a yes or no value.

Example

The following example is the declaration of the **xdYesNo** type:

```
<xsd:simpleType name="xdYesNo">  
  <xsd:restriction base="xsd:NMTOKEN">  
    <xsd:enumeration value="yes" />  
    <xsd:enumeration value="no" />  
  </xsd:restriction>  
</xsd:simpleType>
```


UIContainer Group

Represents a collection of user interface (UI) elements.

Remarks

The **UIContainer** group is used as an element of the [ViewContent](#) group.

Example

The following example is the [XML Schema](#) declaration of the **UIContainer** group:

```
<xsd:group name="UIContainer">
  <xsd:choice>
    <xsd:element ref="xsf:toolbar" />
    <xsd:element ref="xsf:menu" />
    <xsd:element ref="xsf:menuArea" />
  </xsd:choice>
</xsd:group>
```


UIItem Group

Represents a collection of user interface (UI) elements.

Remarks

The **UIItem** group is used as an element of the [toolbar](#), [menu](#), and [menuArea](#) elements.

Example

The following example is the [XML Schema](#) declaration of the **UIItem** group:

```
<xsd:group name="UIItem">
  <xsd:choice>
    <xsd:element ref="xsf:button" />
    <xsd:element ref="xsf:menu" />
  </xsd:choice>
</xsd:group>
```


ViewContent Group

Represents a collection of elements used in a view.

Remarks

The **ViewContent** group is used as an element of the [view](#) element.

Example

The following example is the [XML Schema](#) declaration of the **ViewContent** group:

```
<xsd:group name="ViewContent">
  <xsd:choice>
    <xsd:element ref="xsf:editing" minOccurs="0" />
    <xsd:element ref="xsf:mainpane" minOccurs="0" />
    <xsd:element ref="xsf:printSettings" minOccurs="0" />
    <xsd:group ref="xsf:UIContainer" minOccurs="0" maxOccurs="unbounded" />
    <xsd:element ref="xsf:unboundControls" minOccurs="0" />
  </xsd:choice>
</xsd:group>
```


action Element

Contains the Microsoft Biztalk Server 2004 Human Workflow Services (HWS) action information that has been enabled for the [form](#).

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|--|-----------------|----------|--|---|
| <u>name</u> | xsfxdHWSname | Yes | The unique name of the action as specified by the HWS workflow designer, and used for the onClick event of the button in the Workflow task pane. | Cannot contain characters \ / " [+ = ; @ |
| <u>actionTypeID</u> | xsd:string | Yes | The unique ID string for the action . | |
| <u>canInitiateWorkflow</u> | xsfxdYesNo | Yes | Indicates whether the action can be used to start an activity flow. | <ul style="list-style-type: none"> • yes • no |
| <u>caption</u> | xsfxdHWSCaption | No | The label for the corresponding button in the Workflow task pane to start the action . | minLength = 1 maxLength = 255 |

Definition

```
<xsd:element name="action" >
  <xsd:complexType>
    <xsd:attribute name="name" type="xsf:xdHWSname" use="required" />
    <xsd:attribute name="actionTypeID" type="xsd:string" use="required" />
    <xsd:attribute name="canInitiateWorkflow" type="xsf:xdYesNo" use="optional" />
    <xsd:attribute name="caption" type="xsf:xdHWSCaption" use="optional" />
  </xsd:complexType>
</xsd:element>
```

Remarks

The action element is an optional element of the [allowedActions](#) element.

Each **action** enabled for the form must have a corresponding **action** element in the **allowedActions** section of the [form definition file \(.xsf\)](#).

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **action** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval" />
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate" />
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response" />
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response" />
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond" />
  </xsf:allowedTasks>
</xsf:hwsWorkflow>
```

```
<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID=
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1" />
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/a:some/b:thing" dataObject="Aux1" />
```

```
</xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```


adoAdapter Element

Defines an [ActiveX Data Objects \(ADO\) data adapter](#) that retrieves data from an ADO data source for the specified data object.

Type

xsd:complexType

Structure

| Name | Description |
|---|--|
| <u>commandText</u> | (Required attribute) A string property that contains the ADO SQL command text to be used for querying the data from the specified data source. |
| <u>connectionString</u> | (Required attribute) A string property that contains the ADO connection string to be used to connect to the data source. |
| <u>name</u> | (Optional attribute) Contains the name of the adoAdapter element. |
| <u>queryAllowed</u> | (Optional attribute) Specifies whether data can be retrieved from the data source through the Query method of the data adapter object. |
| <u>submitAllowed</u> | (Optional attribute) Specifies whether data can be submitted to the data source through the Submit method of the data adapter object. |

Remarks

The **adoAdapter** element is an optional element of the [query](#) element.

Example

The following is an example of the **adoAdapter** element:

```
<xsf:query>
  <xsf:adoAdapter
    connectionString="Provider=Microsoft.Jet.OLEDB.4.0;
      Password=&quot;&quot;;User ID=Admin;
      Data Source=infwind.mdb;Mode=Share Deny None;
      Extended Properties=&quot;&quot;;..."
    commandText="select [EmployeeID],[LastName],[FirstName]
      from [Employees] as [Employees]"
    queryAllowed="yes"
    submitAllowed="yes">
  </xsf:adoAdapter>
</xsf:query>
```


allowedActions Element

Contains the Microsoft Biztalk Server 2004 Human Workflow Services (HWS) actions enabled for the [form](#).

Type

xsd:complexType

Child Elements

| Element | Description |
|------------------------|--|
| action | Contains the information for an individual action. |

Attributes

None.

Definition

```
<xsd:element name="allowedActions" >
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:action" minOccurs="1" maxOccurs="20" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="hws_actionTypeID_unique">
    <xsd:selector xpath="/xsf:action" />
    <xsd:field xpath="@actionTypeID" />
  </xsd:unique>
</xsd:element>
```

Remarks

The **allowedActions** element is an optional element of the [hwsWorkflow](#) element.

Each action to be enabled for the form must have a corresponding action element in the **allowedActions** section of the [form definition file \(.xsf\)](#).

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **allowedActions** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval" />
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate" />
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response" />
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response" />
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond" />
  </xsf:allowedTasks>
</xsf:hwsWorkflow>
```

```
<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes" queryAllowed="no">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID="
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1" />
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/a:some/b:thing" dataObject="Aux1"/>
```

```
</xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```


allowedControl Element

Specifies the ActiveX controls that are allowed to be instantiated.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|--------------------------------|------------|----------|--|-----------------|
| <u>cabFile</u> | xsd:string | No | Specifies the name of the CAB file. | string |
| <u>clsid</u> | xsd:string | Yes | Specifies the CLSID (COM class ID) of the ActiveX control. | string |
| <u>version</u> | xsd:string | No | Specifies the ActiveX control version number. | string |

Definition

```
<xsd:element name="allowedControl">  
  <xsd:complexType>  
    <xsd:attribute name="cabFile" type="xsd:string" use="optional"></xsd:attribute>  
    <xsd:attribute name="clsid" type="xsd:string" use="required"></xsd:attribute>  
    <xsd:attribute name="version" type="xsd:string" use="optional"></xsd:attribute>  
  </xsd:complexType>  
</xsd:element>
```

Remarks

When the view contains an **OBJECT** tag, the control will be instantiated only if the CLSID is listed as an **allowedControl** element in the **permissions** element. Controls other than those corresponding to the CLSIDs listed in the **permissions** element are not allowed to be instantiated in the [view](#).

If an ActiveX control listed in the **permissions** element is not installed (that is, if a CLSID of a control is not registered), or if an earlier version of the control than that version specified in the **permissions** element is the only version installed, then the required CAB files will be installed. If the CAB files are not included, or if the installation is stopped, the form will not open.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **allowedControl** element:

```
<xsf:permissions>  
  <xsf:allowedControl  
    cabFile="{84F32C01-78D8-4B93-8ED4-106DA70224C2}.cab"  
    clsid="{84F32C01-78D8-4B93-8ED4-106DA70224C2}"  
    version="1,0,0,1" />  
  <xsf:allowedControl  
    clsid="{F08DF954-8592-11D1-B16A-00C0F0283630}" />  
</xsf:permissions>
```


allowedTasks Element

Contains the Microsoft Biztalk Server 2004 Human Workflow Services (HWS) tasks enabled for the [form](#).

Type

xsd:complexType

Child Elements

| Element | Description |
|----------------------|--|
| task | Contains the information for an individual task. |

Attributes

None.

Definition

```
<xsd:element name="allowedTasks">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:task" minOccurs="1" maxOccurs="20"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="hws_taskID_unique">
    <xsd:selector xpath="./xsf:task"/>
    <xsd:field xpath="@taskTypeID"/>
  </xsd:unique>
</xsd:element>
```

Remarks

The **allowedTasks** element is an optional element of the [hwsWorkflow](#) element.

Each task to be enabled for the form must have a corresponding task element in the **allowedTasks** section of the [form definition file \(.xsf\)](#).

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **allowedTasks** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval" />
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate" />
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response" />
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response" />
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond" />
  </xsf:allowedTasks>
</xsf:hwsWorkflow>
```

```
<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes" queryAllowed="no">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID="
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1" />
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/a:some/b:thing" dataObject="Aux1"/>
```

```
</xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```


applicationParameters Element

Contains form-specific properties that describe how a form should be used in Microsoft Office InfoPath 2003 [design mode](#).

Type

xsd:complexType

Structure

| Name | Description |
|------------------------------------|---|
| application | (Required attribute) Identifies the name of the application used to design the InfoPath form. |
| solutionProperties | (Optional element) Contains design-time information about the InfoPath form. |

Remarks

The **applicationParameters** element is an optional element of the [xDocumentClass](#) element.

Example

The following is an example of the **applicationParameters** element:

```
<xsf: applicationParameters application="InfoPath Design Mode">  
  <xsf: solutionProperties  
    allowCustomization="no"  
    lastOpenView="view1"  
    scriptLanguage="JScript"  
    automaticallyCreateNodes="no"  
    lastVersionNeedingTransform="1.1.0.10"  
    fullyEditableNamespace="urn:names?pace1:mynames"/>  
</xsf:applicationParameters>
```


assignmentAction Element

Defines an action to set the value of a [field](#).

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|------------------------------------|------------|----------|---|-----------------|
| <u>targetField</u> | xsd:string | Yes | Contains an <u>XPath</u> expression for the target node. | string |
| <u>expression</u> | xsd:string | Yes | Contains an <u>XPath</u> expression to populate the value of the targetField . | string |

Definition

```
<xsd:element name="assignmentAction" >
  <xsd:complexType>
    <xsd:attribute name="targetField" type="xsd:string" use="required"
    <xsd:attribute name="expression" type="xsd:string" use="required"
  </xsd:complexType>
</xsd:element>
```

Remarks

The **assignmentAction** element is a child element of the [rule](#) element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **assignmentAction** element:

```
<xf:rule caption="Receipts" condition="my:expense > 75">
  <xf:dialogBoxMessageAction>Don't forget your receipts!</xf:dialogBoxMessageAction>
  <xf:switchViewAction view="Approval View"/>
  <xf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression="sum(my:expenses/my:expense) * my:taxRate"/>
  <xf:queryAction adapter="Exchange Rates"/>
  <xf:submitAction adapter="Exchange Rates"/>
  <xf:openNewDocumentAction solutionURI="uri:microsoft-Expense1">
    <xf:closeDocumentAction promptToSaveChanges="yes"/>
  <xf:dialogBoxExpressionAction>my:group/my:field1</xf:dialogBoxExpressionAction>
  <xf:exitRuleSet/>
</xf:rule>
```


attachmentFileName Element

Contains the file name of the file attachment to be included with the e-mail message when the form is submitted by using the [emailAdapter](#) element.

Type

xsd:complexType

Parent Elements

| Element | Description |
|------------------------------|--|
| emailAdapter | Parent element that contains the information to submit a Microsoft Office InfoPath file as an attachment to an e-mail message, with a specified set of recipients, a subject, and an introduction. |

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Pos Val |
|---------------------------|-------------------------|----------|--|-----------------------|
| value | xsd:string | Yes | Contains the value of the attachmentFileName element. | string |
| valueType | xsf:xdExpressionLiteral | No | Specifies whether the value attribute is interpreted as an XPath expression or a literal string. <ul style="list-style-type: none">• expression• literal | expression literal |

Definition

```
<xsd:element name="attachmentFileName" minOccurs="0">  
  <xsd:complexType>  
    <xsd:attribute name="value" type="xsd:string" use="required"/></xsd:  
    <xsd:attribute name="valueType" type="xsd:string" use="optional" u  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **attachmentFileName** element is a child element of the **emailAdapter** element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **attachmentFileName** element:

```
<xf:emailAdapter name="Submit" submitAllowed="yes">  
  <xf:to value="prst@foo.com" valueType="literal"/>  
  <xf:cc value="my:ccNames" valueType="expression"/>  
  <xf:bcc value="john@bar.com" valueType="literal"/>  
  <xf:subject value="My report" valueType="literal"/>  
  <xf:intro value="See below"/>  
  <xf:attachmentFileName value="Status Report" valueType="literal'  
</xf:emailAdapter>
```

attributeData Element

Specifies the name, and associated value, of an attribute that will be inserted, or modified if it already exists, by the insert action of the xCollection or xOptional editing components.

Type

xsd:complexType

Structure

| Name | Description |
|---------------------------|---|
| attribute | (Required attribute) Specifies the name of the attribute to be inserted. |
| value | (Required attribute) Specifies the value of the attribute to be inserted. |

Remarks

The **attributeData** element is an optional element of the [chooseFragment](#) element.

Example

The following is an example of the **attributeData** element:

```
<xsf:editWith component="xOptional">
  <xsf:fragmentToInsert>
    <xsf:chooseFragment parent="report">
      <xsf:attributeData attribute="author" value="author name"/>
    </xsf:chooseFragment>
  </xsf:fragmentToInsert>
</xsf:editWith>
```

autoRecovery Element

Specifies whether the form will save AutoRecover information and whether the AutoRecover setting can be changed by the user.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|-------------------------|-----------------------|----------|--|--|
| feature | xsf:xdEnabledDisabled | required | Sets whether the AutoRecover feature is enabled. | <ul style="list-style-type: none">• enabled• disabled |

Definition

```
<xsd:element name="autoRecovery">  
  <xsd:complexType>  
    <xsd:attribute name="feature" type="xsd:boolean" use="optional" />  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **autoRecovery** element is an optional element of the [featureRestrictions](#) element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **autoRecovery** element:

```
<xsf:featureRestrictions>  
  <xsf:save ui="disabled"/>  
  <xsf:sendMail ui="disabled"/>  
  <xsf:exportToWeb ui="disabled"/>  
  <xsf:exportToExcel ui="disabled"/>  
  <xsf:print ui="enabled"/>  
  <xsf:autoRecovery feature="disabled"/>  
</xsf:featureRestrictions>
```


bcc Element

Contains information related to the BCC recipients of the e-mail message when the form is submitted using the [emailAdapter](#) element.

Type

xsd:complexType

Parent Elements

| Element | Description |
|------------------------------|--|
| emailAdapter | Parent element that contains the information needed to submit a Microsoft Office InfoPath 2003 SP1 file as an attachment to an e-mail, with a specified set of recipients, a subject, and an introduction. |

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|---------------------------|-------------------------|----------|---|-----------------|
| value | xsd:string | Yes | Contains the value of the bcc element. | string |
| valueType | xsf:xdExpressionLiteral | No | Specifies whether the value attribute is interpreted as an XPath expression or as a literal string. <ul style="list-style-type: none">• expression• literal | |

Definition

```
<xsd:element name="bcc" minOccurs="0">
  <xsd:complexType>
    <xsd:attribute name="value" type="xsd:string" use="required"></xsd:attribute>
    <xsd:attribute name="valueType" type="xsd:string" use="optional" default="text" />
  </xsd:complexType>
</xsd:element>
```

Remarks

If multiple addresses are specified for the value of the **bcc** element, the addresses must be separated by semicolons.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **bcc** element:

```
<xf:emailAdapter name="Submit" submitAllowed="yes">
  <xf:to value="someone@example.com" " valueType="literal"/>
  <xf:cc value="my:ccNames" valueType="expression"/>
  <xf:bcc value="someoneelse@example.com" valueType="literal"/>
  <xf:subject value="My report" valueType="literal"/>
  <xf:intro value="See below"/>
  <xf:attachmentFileName value="Status Report" valueType="literal"/
</xf:emailAdapter>
```


button Element

Defines a button that has an associated action.

Type

xsd:complexType

Structure

| Name | Description |
|----------------------------------|--|
| <u>action</u> | (Optional attribute) Specifies an action of an editing component, using the syntax "NameOfEditingComponent::NameOfAction". |
| <u>caption</u> | (Optional attribute) Provides the caption displayed on the button. |
| <u>icon</u> | (Optional attribute) Provides a Uniform Resource Locator (URL) to a bitmap (.bmp) or graphics interchange format (.gif) file, which is used for the button or menu item. |
| <u>name</u> | (Optional attribute) Used to associate the OnClick event handler of the button with a scripting function. |
| <u>showlf</u> | (Optional attribute) Specifies the editing context of the button. |
| <u>tooltip</u> | (Optional attribute) Provides the ScreenTip text to be used for the button. |
| <u>xmlToEdit</u> | (Optional attribute) Specifies the name of an xmlToEdit element, for which the button is used. |

Remarks

The **button** element is an optional element of the [toolbar](#), [menu](#), and [menuArea](#) elements. Each button element declaration corresponds to a button on a toolbar, menu, or menu area item, and each button has an action (or command) associated with it.

Example

The following is an example of the **button** element:

```
<xsf:menuArea name="msoInsertMenu">
  <xsf:menu caption="&Section">
    <xsf:button action="xCollection::insert" xmlToEdit="CD_10"
      caption="CD" showIf="always"></xsf:button>
    <xsf:button action="xCollection::insert" xmlToEdit="Track_14"
      caption="Track" showIf="always"></xsf:button>
    <xsf:button action="xOptional::insert" xmlToEdit="Label_16"
      caption="Label"></xsf:button>
  </xsf:menu>
</xsf:menuArea>
```


calculatedField Element

Defines an individual calculation, including the [formula](#), when the calculation is to be performed, and where the result will be stored.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|----------------------------|------------|----------|--|---|
| target | xsd:string | Yes | Contains the XPath location where the result of the expression will be stored. | string |
| expression | xsd:string | Yes | Contains the formula , in the form of an XPath expression, to be evaluated. The result is stored in the target location. | string |
| refresh | xsd:string | Yes | Specifies when the expression will be evaluated. | <ul style="list-style-type: none">• onInit• onChange |

Definition

```
<xsd:element name="calculatedField">
  <xsd:complexType>
    <xsd:attribute name="target" type="xsd:string" use="required"></xsd:attribute>
    <xsd:attribute name="expression" type="xsd:string" use="required"></xsd:attribute>
    <xsd:attribute name="refresh" type="xsd:string" use="required"></xsd:attribute>
  </xsd:complexType>
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **calculatedField** element:

```
<xsf:calculations>  
  <xsf:calculatedField  
    target="/my:myFields/my:average"  
    expression="xdMath:Avg(..my:expenses/my:expense/my:amount)"  
    refresh="onChange"/>  
</xsf:calculations>
```


calculations Element

Contains definitions for calculations performed in the [form](#) and specifies how blank values are handled.

Type

xsd:complexType

Child Elements

| Element | Description |
|---------------------------------|--|
| calculatedField | Defines an individual calculation, including the formula , when the calculation is to be performed, and where the result will be stored. |

Attributes

| Attribute | Type | Required | Description | Possible Values |
|---------------------------------------|-------------|----------|--|--|
| treatBlankValueAsZero | xsf:xdYesNo | No | Specifies whether a blank field should be calculated with a value of zero. Default is yes . | <ul style="list-style-type: none">• yes• no |

Definition

```
<xsd:element name="calculations">  
  <xsd:complexType>  
    <xsd:sequence>  
      <xsd:element ref="xsf:calculatedField" minOccurs="0" maxOccurs="1" />  
    </xsd:sequence>  
    <xsd:attribute name="treatBlankValueAsZero" type="xsf:xdYesNo" />  
  </xsd:complexType>  
</xsd:element>
```

Remarks

Every calculation in the [form](#) will have a **calculatedField** element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **calculations** element:

```
<xsf:calculations>  
  <xsf:calculatedField  
    target="/my:myFields/my:average"  
    expression="xdMath:Avg(..my:expenses/my:expense/my:amount)"  
    refresh="onChange" />  
</xsf:calculations>
```


cc Element

Contains information related to the CC recipients of the e-mail message when the form is submitted using the [emailAdapter](#) element.

Type

xsd:complexType

Parent Elements

| Element | Description |
|------------------------------|--|
| emailAdapter | Parent element that contains the information needed to submit a Microsoft Office InfoPath 2003 SP1 file as an attachment to an e-mail, with a specified set of recipients, a subject, and an introduction. |

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|---|------------|----------|--|-----------------|
| value | xsd:string | Yes | Contains the value of the cc element. | string |
| valueType xdExpressionLiteral | | No | Specifies whether the value attribute is interpreted as an XPath expression or a literal string. <ul style="list-style-type: none">• expression• literal | |

Definition

```
<xsd:element name="cc" minOccurs="0">  
  <xsd:complexType>  
    <xsd:attribute name="value" type="xsd:string" use="required"/></xsd:  
    <xsd:attribute name="valueType" type="xsd:string" use="optional" u  
  </xsd:complexType>  
</xsd:element>
```

Remarks

If multiple addresses are specified for the value of the **cc** element, the addresses must be separated by semicolons.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **cc** element:

```
<xsf:emailAdapter name="Submit" submitAllowed="yes">  
  <xsf:to value="someone@example.com" valueType="literal"/>  
  <xsf:cc value="my:ccNames" valueType="expression"/>  
  <xsf:bcc value="someoneelse@example.com" valueType="literal"/>  
  <xsf:subject value="My report" valueType="literal"/>  
  <xsf:intro value="See below"/>  
  <xsf:attachmentFileName value="Status Report" valueType="literal"/>  
</xsf:emailAdapter>
```


chooseFragment Element

Specifies an XML fragment.

Type

xsd:complexType

Structure

| Name | Description |
|-----------------------------------|--|
| attributeData | (Optional element) Specifies the name, and associated value, of an attribute that will be inserted, or modified if it already exists, by the insert action of the xCollection or xOptional editing components. |
| followingSiblings | (Optional attribute) Specifies a relative XPath expression from the parent node that specifies the XML Document Object Model (DOM) nodes prior to which the insertion of the XML fragment should occur. Note This is not necessary in Microsoft Office InfoPath 2003 Service Pack 1 and will not be automatically generated by InfoPath. |
| parent | (Optional attribute) Specifies a relative XPath expression from the container node that specifies the XML DOM node under which the XML fragment is inserted. |
| innerFragment | (Optional attribute) Specifies a relative XPath expression from the parent node to the smallest fragment to be inserted. Note Use of this attribute requires Microsoft Office InfoPath 2003 Service Pack 1. |

Remarks

The **chooseFragment** element is a required element of the [fragmentToInsert](#) element.

The **chooseFragment** element has an [open content model](#). It may contain text, or one or more element nodes, or mixed content (both element nodes and text nodes). In addition to or instead of XML data to be inserted directly as a fragment, it can contain one or more **attributeData** elements. In this case the **attributeData** elements are not included as inserted content, but are each used to specify setting an attribute value.

The **chooseFragment** elements are typically ordered in increasing size. The first will be the data fragment to be inserted by the insertBefore and insertAfter actions, when there is already at least one item in the collection. The insert action, on the other hand, can be invoked when there is currently a node in the XML tree corresponding to a container, but no node corresponding to an item (in other words, it can be used to insert the first item).

Note Microsoft Office InfoPath 2003 Service Pack 1 will generate only one **chooseFragment** node. For more information, see the [innerFragment](#) attribute.

Note Any element content within the XML fragment, other than attribute data, corresponds to new content to be inserted into the form's underlying XML document, and should be in the appropriate namespace.

Example

The following is an example of the **chooseFragment** element:

```
<xsf:editing>
  <xsf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xsf:editWith caption="CD"
      xd:autogeneration="template"
      component="xCollection">
      <xsf:fragmentToInsert>
        <xsf:chooseFragment parent="CDCollection">
          <CD>
            <Title></Title>
            <Artist></Artist>
            <Tracks>
              <Track></Track>
              <Track></Track>
            </Tracks>
          </CD>
        </xsf:chooseFragment>
      </xsf:fragmentToInsert>
    </xsf:editWith>
  </xsf:xmlToEdit>
</xsf:editing>
```


closeDocumentAction Element

Defines a [form](#) close action.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|-------------------------------------|-------------------------|----------|---|--|
| promptToSaveChanges | xdYesNo | Yes | Specifies whether the user is prompted to save changes to the form before the action completes. Default is "yes". | <ul style="list-style-type: none">• yes• no |

Definition

```
<xsd:element name="closeDocumentAction">  
  <xsd:complexType>  
    <xsd:attribute name="promptToSaveChanges" type="xsd:boolean"/>  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **closeDocumentAction** element is a child element of the [rule](#) element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **closeDocumentAction** element:

```
<xf:rule caption="Receipts" condition="my:expense > 75">
  <xf:dialogBoxMessageAction>Don't forget your receipts!</xf:dialogBoxMessageAction>
  <xf:switchViewAction view="Approval View"/>
  <xf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression="sum(my:expenses/my:expense) * my:taxRate"/>
  <xf:queryAction adapter="Exchange Rates"/>
  <xf:submitAction adapter="Exchange Rates"/>
  <xf:openNewDocumentAction solutionURI="uri:microsoft-ExpenseForm">
    <xf:closeDocumentAction promptToSaveChanges="yes"/>
  <xf:dialogBoxExpressionAction>my:group/my:field1</xf:dialogBoxExpressionAction>
  <xf:exitRuleSet/>
</xf:rule>
```


customCategory Element

Specifies the category that the [form template](#) appears under in the **Fill Out a Form** task pane.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|----------------------|------------|----------|--|---|
| name | xs:xdTitle | required | Specifies the name of the custom category. | minLength = 1 maxLength = 255 pattern = ([\p{Z}\p{Cc}\p{Cf}\p{Cn}]) (([\p{Zl}\p{Zp}\p{Cc}])*) ([\p{Z}\p{Cc}\p{Cf}\p{Cn}]?) |

Definition

```
<xsd:element name="customCategory">  
  <xsd:complexType>  
    <xsd:attribute name="name" type="xsd:string" use="required"/></xsd:attribute>  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The customCategory element is an optional element of the [initialXmlDocument](#) element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **customCategory** element:

```
<xsf:fileNew>  
  <xsf:initialXmlDocument  
    caption="Travel Report"  
    href="TravelReportTemplate.xml">  
    <xsf:customCategory name="Reports"/>  
  </xsf:initialXmlDocument>  
</xsf:fileNew>
```


customValidation Element

Defines rule-based custom validation on top of all validation enforced through the [XML Schema](#).

Type

xsd:complexType

Structure

| Name | Description |
|--------------------------------|--|
| errorCondition | (Optional element) Defines a custom validation (or error condition) for a specific XML Document Object Model (DOM) node in a form's underlying XML document. |

Remarks

The **customValidation** element is an optional element of the [xDocumentClass](#) element.

Note It is possible to create multiple error conditions on a field in a form using multiple **errorCondition** elements, but they will not appear in the **Data Validation** dialog box while in [design mode](#).

Example

The following is an example of the **customValidation** element:

```
<xsf:customValidation>
  <xsf:errorCondition
    match="/exp:expenseReport"
    expressionContext="exp:reportDate"
    expression="msxsl:string-compare(., ../exp:startDate) < 0 and ../ex]
    showErrorOn=".">
    <xsf:errorMessage
      type="modeless"
      shortMessage="The report date occurs before the end of the expe
      The report date occurs before the end of the expense period. Veri:
    </xsf:errorMessage>
  </xsf:errorCondition>
</xsf:customValidation>
```


dataAdapters Element

Contains information about [submit](#) adapters that are used as the main submit adapters for the [form](#) and that therefore are not declared inline within the **submit** element.

Type

xsd:complexType

Child Elements

| Element | Description |
|--|--|
| <u>webServiceAdapter</u> | Contains information to enable InfoPath forms to be submitted to an XML Web service. |
| <u>davAdapter</u> | Contains information to enable InfoPath forms to be submitted to a server running Microsoft Windows SharePoint Services or to a <u>Web-based Distributed Authoring and Versioning (WebDAV)</u> server. |
| <u>emailAdapter</u> | Contains information to enable submission of an Infopath form as an attachment to an e-mail, with a specified set of recipients, a subject, and an introduction. |
| <u>hwsAdapter</u> | Defines the Microsoft Biztalk 2004 Human Workflow Services (HWS) data adapter that is used to start or extend an activity flow and respond to a task. |

Attributes

None.

Definition

```
<xsd:element name="dataAdapters">  
  <xsd:complexType>  
    <xsd:choice minOccurs="0" maxOccurs="unbounded">  
      <xsd:element ref="xsf:webServiceAdapter"/>  
      <xsd:element ref="xsf:davAdapter"/>  
      <xsd:element ref="xsf:emailAdapter"/>  
      <xsd:element ref="xsf:hwsAdapter"/>  
    </xsd:choice>  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **dataAdapters** element is an optional element of the [xDocumentClass](#) element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **dataAdapters** element:

```
<xf:dataAdapters>  
  <xf:webServiceAdapter name="submit 2" ... submitAllowed="yes">  
    ...  
  </xf:webServiceAdapter>  
  <xf:emailAdapter name="submit 4" ... submitAllowed="yes"/>  
</xf:dataAdapters>
```


dataObject Element

Defines a secondary data object that is used in a Microsoft Office InfoPath 2003 form.

Type

xsd:complexType

Structure

| Name | Description |
|-----------------------------------|--|
| <u>initOnLoad</u> | (Optional attribute) Specifies whether the data object should be initialized on document load. |
| <u>name</u> | (Required attribute) The unique name for the data object. |
| <u>schema</u> | (Optional attribute) The name of an XML Schema file. |
| <u>query</u> | (Required element) Associates the data adapter with the data object or a form's underlying XML document. |

Remarks

The **dataObject** element is an optional element of the [dataObjects](#) element.

Multiple dataObject elements are allowed within a form. Each data object is an XML [Document Object Model \(DOM\)](#) populated from an external data source that can be accessed directly (by name) from the [XSL Transformation \(XSLT\)](#)-based view code and any script-based business logic code in the form.

Example

The following is an example of the **dataObject** element:

```
<xf:dataObjects>
  <xf:dataObject
    name="EmployeeNames"
    schema="EmployeeNames.xsd"
    initOnLoad="yes">
    <xf:query>
      <xf:adoAdapter
        connectionString="Provider=Microsoft.Jet.OLEDB.4.0;
          Password=&quot;&quot;;User ID=Admin;
          Data Source=infwind.mdb;Mode=Share Deny None;
          Extended Properties=&quot;&quot;;..."
        commandText="select [EmployeeID],[LastName],[FirstName]
          from [Employees] as [Employees]"
        queryAllowed="yes"
        submitAllowed="yes">
      </xf:adoAdapter>
    </xf:query>
  </xf:dataObject>
</xf:dataObjects>
```


dataObjects Element

Defines all secondary data objects used in a Microsoft Office InfoPath 2003 form.

Type

xsd:complexType

Structure

| Name | Description |
|----------------------------|--|
| dataObject | (Optional element) Defines a secondary data object that is used in an InfoPath form. |

Remarks

The **dataObjects** element is an optional element of the [xDocumentClass](#) element.

The **dataObjects** element contains a collection of data objects that are used to populate various XML [Document Object Models \(DOMs\)](#) from external data sources. These data objects can be accessed directly (by name) from the [XSL Transformation \(XSLT\)](#)-based view code and any script-based business logic code in the form.

Example

The following is an example of the **dataObjects** element:

```
<xf:dataObjects>
  <xf:dataObject
    name="EmployeeNames"
    schema="EmployeeNames.xsd"
    initOnLoad="yes">
    <xf:query>
      <xf:adoAdapter
        connectionString="Provider=Microsoft.Jet.OLEDB.4.0;
          Password=&quot;&quot;;User ID=Admin;
          Data Source=infwind.mdb;Mode=Share Deny None;
          Extended Properties=&quot;&quot;;..."
        commandText="select [EmployeeID],[LastName],[FirstName]
          from [Employees] as [Employees]"
        queryAllowed="yes"
        submitAllowed="yes">
      </xf:adoAdapter>
    </xf:query>
  </xf:dataObject>
</xf:dataObjects>
```


davAdapter Element

Contains information to enable InfoPath forms to be submitted to a server running Microsoft Windows SharePoint Services or to a [Web-based Distributed Authoring and Versioning \(WebDAV\)](#) server.

Type

xsd:complexType

Child Elements

| Element | Description |
|----------------------------------|---|
| <u>folderURL</u> | Contains the Uniform Resource Locator (URL) of the server to which the file is submitted. |
| <u>fileName</u> | Contains the name of the file as a literal string or XPath expression. |

Attributes

| Attribute | Type | Required | Description | Possible Values |
|----------------------------------|-------------------------|----------|---|---|
| name | xdTitle | Yes | The name of the adapter. Used when invoking the davAdapter from code. | minLength = 1 maxLength = 255 pattern = ([\p{Z}\p{Cc}]\p{Cf}) (([\p{Zl}\p{Zp}]\p{C}) ([\p{Z}\p{Cc}]\p{Cf}) |
| overwriteAllowed | xdYesNo | No | Specifies whether the adapter can overwrite an existing file. | <ul style="list-style-type: none"> • yes • no |
| queryAllowed | xdYesNo | No | Specifies whether the adapter can be used for querying the data source. Omitted for the davAdapter , corresponding to a default value of "no". | <ul style="list-style-type: none"> • yes • no |
| submitAllowed | xdYesNo | No | Specifies whether the adapter can be used for submitting to the data source. Always set to | <ul style="list-style-type: none"> • yes • no |

"yes" for the
davAdapter.

Definition

```
<xsd:element name="davAdapter" >
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="folderURL">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required">
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="fileName">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required">
          <xsd:attribute name="valueType" type="xsf:xdExpressionLitera
        </xsd:complexType>
      </xsd:element>
    </xsd:all>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required"></x
    <xsd:attribute name="overwriteAllowed" type="xsf:xdYesNo" use=
    <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="opt
    <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="opt
  </xsd:complexType>
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **davAdapter** element:

```
<xsf:davAdapter name="SubmitToSharePoint" overwriteAllowed="y  
<xsf:fileName value="my:myFields/my:fileName" valueType="expre  
<xsf:folderURL value="http://some_server/some_doc_lib"/>  
</xsf:davAdapter>
```


dialogBoxExpressionAction Element

Defines an [XPath](#) expression to be displayed in a dialog box.

Type

xsd:string

Child Elements

None.

Attributes

None.

Definition

```
<xsd:element name="dialogBoxExpressionAction" type="xsd:string">
```

Remarks

The **dialogBoxExpressionAction** element is a child element of the [rule](#) element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **dialogBoxExpressionAction** element:

```
<xsf:rule caption="Receipts" condition="my:expense > 75">
  <xsf:dialogBoxMessageAction>Don't forget your receipts!</xsf:dialogBoxMessageAction>
  <xsf:switchViewAction view="Approval View"/>
  <xsf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression="sum(my:expenses/my:expense) * my:taxRate"/>
  <xsf:queryAction adapter="Exchange Rates"/>
  <xsf:submitAction adapter="Exchange Rates"/>
  <xsf:openNewDocumentAction solutionURI="uri:microsoft-Expense1">
  <xsf:closeDocumentAction promptToSaveChanges="yes"/>
  <xsf:dialogBoxExpressionAction>my:group/my:field1</xsf:dialogBoxExpressionAction>
  <xsf:exitRuleSet/>
</xsf:rule>
```

dialogBoxMessageAction Element

Defines a literal message to be displayed in a dialog box.

Type

xsd:simpleType

Child Elements

None.

Attributes

None.

Definition

```
<xsd:element name="dialogBoxMessageAction">  
  <xsd:simpleType>  
    <xsd:restriction base="xsd:string">  
      <xsd:maxLength value="1024"/>  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:element>
```

Remarks

The **dialogBoxMessageAction** element is a child element of the [rule](#) element.

The maximum length of the message is 1,024 characters.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **dialogBoxMessageAction** element:

```
<<xf:rule caption="Receipts" condition="my:expense > 75">
  <xf:dialogBoxMessageAction>Don't forget your receipts!</xf:dialogBoxMessageAction>
  <xf:switchViewAction view="Approval View"/>
  <xf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression="sum(my:expenses/my:expense) * my:taxRate"/>
  <xf:queryAction adapter="Exchange Rates"/>
  <xf:submitAction adapter="Exchange Rates"/>
  <xf:openNewDocumentAction solutionURI="uri:microsoft-Expense1"/>
  <xf:closeDocumentAction promptToSaveChanges="yes"/>
  <xf:dialogBoxExpressionAction>my:group/my:field1</xf:dialogBoxExpressionAction>
  <xf:exitRuleSet/>
</xf:rule>
```


documentSchema Element

Defines an [XML Schema](#) for a form.

Type

xsd:complexType

Structure

| Name | Description |
|----------------------------|---|
| location | (Required attribute) Contains the namespace Uniform Resource Identifier (URI) and location (a Uniform Resource Locator (URL) , relative to the form definition (.xsf) file), and delimited by a white space, of the .xsd file defining the XML Schema. |
| rootSchema | (Optional attribute) Identifies an XML Schema as the top-level schema of the form being filled out. |

Remarks

The **documentSchema** element is a required element of the [documentSchemas](#) element. One **documentSchema** element is present for each declared XML Schema in the form.

Microsoft Office InfoPath 2003 includes all XML Schemas in the [form template](#) and modifies the schema references in the .xsd files to be relative path names. If a form's underlying XML document contains references to multiple XML Schemas, they are listed with separate **documentSchema** elements and the top-level XML Schema has its **rootSchema** attribute set to "yes".

If an XML Schema file contains references to other XML Schema files using the include or import settings, those referenced files do not have to be listed in the **documentSchemas** element. However, they must be included in the form template with their references changed to relative file names.

Example

The following is an example of the **documentSchema** element:

```
<xsf:documentSchemas>  
  <xsf:documentSchema  
    location="urn:schema:custom:Namespace customFilename.xsd"  
    rootSchema="yes"/>  
</xsf:documentSchemas>
```


documentSchemas Element

Defines the [XML Schemas](#) that the Microsoft Office InfoPath 2003 form is designed to handle.

Type

xsd:complexType

Structure

| Name | Description |
|--------------------------------|---|
| documentSchema | (Required element) Defines an XML Schema for a form. One documentSchema element is present for each declared XML Schema in the form. |

Remarks

The **documentSchemas** element is an optional element of the [xDocumentClass](#) element. It defines all of the target XML Schemas that are used in the form.

Example

The following is an example of the **documentSchemas** element:

```
<xsf:documentSchemas>  
  <xsf:documentSchema  
    location="urn:schema:custom:Namespace customFilename.xsd"  
    rootSchema="yes"/>  
</xsf:documentSchemas>
```


documentSignatures Element

Contains the [signedDataBlock](#) element, which defines how digital signatures are applied to a form or section of a form.

Type

xsd:complexType

Structure

| Name | Description |
|-----------------------------------|---|
| signatureLocation | (Optional attribute) Contains an XPath expression that points to the XML DOM node within the form's underlying XML document that is used for storing the digital signature. |

Child Elements

| Element | Description |
|---------------------------------|--|
| signedDataBlock | Defines a nodeset in the form's underlying XML document to which a digital signature can be applied. |

Definition

```
<xsd:element name="documentSignatures">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:signedDataBlock" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="signatureLocation" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="signedDataBlock">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="message" type="xsf:xdSignedDataBlock"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsf:xdSignedDataBlockName"/>
    <xsd:attribute name="data" type="xsd:string" use="required"/>
    <xsd:attribute name="signatureLocation" type="xsd:string"/>
    <xsd:attribute name="mode" type="xsf:xdSignatureRelationship"/>
  </xsd:complexType>
  <xsd:unique name="signedDataBlock_name_unique">
    <xsd:selector xpath="." />
    <xsd:field xpath="@name" />
  </xsd:unique>
</xsd:element>
```

Remarks

The **signedDataBlock** element is a new XSF element addition in Microsoft Office InfoPath 2003 Service Pack 1. See below for backward compatibility information with 1.0 form solutions.

For backward compatibility, the document signatures element defines the location of the digital signature XML [Document Object Model \(DOM\)](#) node within the form's underlying XML document.

The **documentSignatures** element is an optional element of the [xDocumentClass](#) element.

Example

The following is an example of the **documentSignatures** element:

```
<xsf:documentSignatures signatureLocation="my:myfields/my:subtr
```

The following is an example of the **documentSignatures** element used in Microsoft Office InfoPath 2003 Service Pack 1:

```
<xsf:documentSignatures>  
<xsf:signedDataBlock name="main"  
  data="my:myfields/my:subtree1 | my:myfields/my:subtree2"  
  signatureLocation="my:mifields/sig:signatures/main"  
  mode="countersign">  
  <xsf:message>By pressing the &quot;Sign&quot; button below, I agr  
</xsf:signedDataBlock>  
</xsf:documentSignatures>
```


documentVersionUpgrade Element

Defines how forms created with an older version of the [form template](#) can be upgraded to the latest version of the form template.

Type

xsd:complexType

Structure

| Name | Description |
|----------------------------------|---|
| useScriptHandler | (Optional element) Specifies that the upgrade will be handled using scripting code. |
| useTransform | (Optional element) Specifies that the upgrade will be handled by an XSL Transformation (XSLT) supplied by the newer version of the form template. |

Remarks

The **documentVersionUpgrade** element is an optional element of the [xDocumentClass](#) element.

Example

The following is an example of the **documentVersionUpgrade** element:

```
<xsf:documentVersionUpgrade>  
  <xsf:useTransform  
    transform="upgrade.xsl"  
    minVersionToUpgrade="0.0.0.0"  
    maxVersionToUpgrade="1.0.0.5"/>  
</xsf:documentVersionUpgrade>
```


domEventHandler Element

Defines an [event handler](#) for one or more specific XML [Document Object Model \(DOM\)](#) nodes.

Type

xsd:complexType

Structure

| Name | Description |
|-------------------------------|--|
| handlerObject | (Required attribute) Identifies the unique name of the event handler in the scripting code. |
| match | (Required attribute) Identifies the XML DOM node for which the event handler is declared. Value must be a valid XPath expression that identifies the XML node. |
| dataObject | (Optional attribute) Contains the name of the dataObject to be used in the event handler. |
| | Note Use of this attribute requires Microsoft Office InfoPath 2003 Service Pack 1. |

Remarks

The **domEventHandler** element is an optional element of the [domEventHandlers](#) element.

Example

The following is an example of the **domEventHandler** element:

```
<xsf:domEventHandlers>  
  <xsf:domEventHandler  
    match="TravelReport/Expenses"  
    handlerObject="TravelExpenses"/>  
</xsf:domEventHandlers>
```


domEventHandlers Element

Contains pointers to various script-based [event handlers](#) that react to changes in XML [Document Object Model \(DOM\)](#) nodes of a form's underlying XML document when the form is being filled out.

Type

xsd:complexType

Structure

| Name | Description |
|---------------------------------|--|
| domEventHandler | (Optional element) Defines an event handler object for one or more specific XML DOM nodes. |

Remarks

The **domEventHandlers** element is an optional element of the [xDocumentClass](#) element.

Example

The following is an example of the **domEventHandlers** element:

```
<xf:domEventHandlers>  
  <xf:domEventHandler  
    match="TravelReport/Expenses"  
    handlerObject="TravelExpenses"/>  
</xf:domEventHandlers>
```


editing Element

Contains information about the editing components used in the view.

Type

xsd:complexType

Structure

| Name | Description |
|---------------------------|---|
| xmlToEdit | (Optional element) Specifies an instance of an editing component. |

Remarks

The **editing** element is an optional element of the [view](#) element.

The editing components section of the Microsoft Office InfoPath 2003 [form definition \(.xsf\) file](#) defines how and when users are able to edit specified XML [Document Object Model \(DOM\)](#) nodes of a form's underlying XML document. Only one **editing** element is allowed per view. Each **editing** element can contain zero or more **xmlToEdit** elements.

Example

The following is an example of the **editing** element:

```
<xf:editing>
  <xf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xf:editWith caption="CD"
      xd:autogeneration="template"
      component="xCollection">
      <xf:fragmentToInsert>
        <xf:chooseFragment parent="CDCollection">
          <CD>
            <Title></Title>
            <Artist></Artist>
            <Tracks>
              <Track></Track>
              <Track></Track>
            </Tracks>
          </CD>
        </xf:chooseFragment>
      </xf:fragmentToInsert>
    </xf:editWith>
  </xf:xmlToEdit>
</xf:editing>
```


editWith Element

Specifies an instance of an editing component, and provides the corresponding parameters to determine its exact behavior.

Type

xsd:complexType

Structure

| Name | Description |
|---|---|
| <u>allowedFileTypes</u> | (Optional attribute) Specifies the file extensions of files that can be attached to the <u>form</u> . |
| <u>autoComplete</u> | (Optional attribute) Switches the auto-completion of fields on or off. |
| <u>caption</u> | (Optional attribute) Specifies an identifier for alternate forms of XML data to be used in the editing component. |
| <u>component</u> | (Required attribute) Specifies the name of the editing component that will be referenced within the <u>action</u> attribute of a <u>button</u> element. |
| <u>field</u> | (Optional attribute) Specifies a relative XPath expression from the XML <u>Document Object Model (DOM)</u> node specified by the <u>item</u> attribute of the <u>xmlToEdit</u> element. |
| <u>fragmentToInsert</u> | (Optional element) Contains alternate versions of XML data (fragments). |
| <u>proofing</u> | (Optional attribute) Switches the proofing features, such as the spelling checker, on or off. |
| <u>removeAncestors</u> | (Optional attribute) Specifies the number of ancestor (parent) elements to be removed when the last item is removed. |
| <u>type</u> | (Optional attribute) Specifies the type of editing for the fields that match the XPath expression specified by the <u>item</u> attribute of the <u>xmlToEdit</u> element. |
| <u>widgetIcon</u> | (Optional attribute) Specifies whether or not a modified widget icon will be shown for filtered items. |
| <u>useFilter</u> | (Optional attribute) Indicates the user wants a filter widget. |
| <u>filterDependency</u> | (Optional attribute) Specifies automatic reapplication of the filter when filter fields change. |

maxLength

(Optional attribute) Specifies the maximum number of characters allowed for text boxes.

Remarks

The **editWith** element is an optional element of the [xmlToEdit](#) element.

If the [viewContext](#) attribute of the **xmlToEdit** element is defined, the parameters of the **editWith** element are associated with the specified view context.

Example

The following is an example of the **editWith** element:

```
<xf:editing>
  <xf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xf:editWith caption="CD"
      xd:autogeneration="template"
      component="xCollection">
      <xf:fragmentToInsert>
        <xf:chooseFragment parent="CDCollection">
          <CD>
            <Title></Title>
            <Artist></Artist>
            <Tracks>
              <Track></Track>
              <Track></Track>
            </Tracks>
          </CD>
        </xf:chooseFragment>
      </xf:fragmentToInsert>
    </xf:editWith>
  </xf:xmlToEdit>
</xf:editing>
```

emailAdapter Element

Contains the information needed to submit an InfoPath form as an attachment to an e-mail, with a specified set of recipients, a subject, and an introduction.

Type

xsd:complexType

Child Elements

| Element | Description |
|---|--|
| <u>to</u> | Contains a list of addresses, separated by semicolons, to be added to the to line of the submitted e-mail. |
| <u>cc</u> | Contains a list of addresses, separated by semicolons, to be added to the cc line of the submitted e-mail. |
| <u>bcc</u> | Contains a list of addresses, separated by semicolons, to be added to the bcc line of the submitted e-mail. |
| <u>subject</u> | Contains the subject of the submitted e-mail. |
| <u>intro</u> | Contains the introduction of the submitted e-mail. |
| <u>attachmentFileName</u> | Contains the file name of the attachment to be submitted with the e-mail. |

Attributes

| Attribute | Type | Required | Description | Possible Values |
|--------------------------------------|--------------------------------|----------|---|--|
| <u>name</u> | <u>xdTitle</u> | Yes | Contains the name of the emailAdapter . | minLength = 1 maxLength = 255 pattern = ([^\p{Z}\p{Cc}\p{Cf}\p{Zl}\p{Zp}\p{Cc}][^\p{Z}\p{Cc}\p{Cf}] |
| <u>queryAllowed</u> | <u>xdYesNo</u> | No | Specifies whether the adapter can be used for querying the data source. Omitted for the emailAdapter , corresponding to a default value of "no". | <ul style="list-style-type: none"> • yes • no |
| <u>submitAllowed</u> | <u>xdYesNo</u> | No | Specifies whether the adapter can be used for submitting to the data source. Always set to "yes" for the emailAdapter . | <ul style="list-style-type: none"> • yes • no |

Definition

```
<xsd:element name="emailAdapter">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="to" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required">
            <xsd:attribute name="valueType" type="xsd:string" use="required" />
          </xsd:complexType>
        </xsd:element>
      <xsd:element name="cc" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required">
            <xsd:attribute name="valueType" type="xsd:string" use="required" />
          </xsd:complexType>
        </xsd:element>
      <xsd:element name="bcc" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required">
            <xsd:attribute name="valueType" type="xsd:string" use="required" />
          </xsd:complexType>
        </xsd:element>
      <xsd:element name="subject" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="value" type="xsd:string" use="required">
            <xsd:attribute name="valueType" type="xsd:string" use="required" />
          </xsd:complexType>
        </xsd:element>
      <xsd:element name="intro" minOccurs="0">
        <xsd:complexType>
```

```
<xsd:attribute name="value" type="xsd:string" use="required">
</xsd:complexType>
</xsd:element>
<xsd:element name="attachmentFileName" minOccurs="0">
  <xsd:complexType>
    <xsd:attribute name="value" type="xsd:string" use="required">
      <xsd:attribute name="valueType" type="xsd:expression" use="optional" />
    </xsd:complexType>
  </xsd:element>
</xsd:all>
<xsd:attribute name="name" type="xsd:string" use="required"></xsd:attribute>
<xsd:attribute name="queryAllowed" type="xsd:boolean" use="optional" />
<xsd:attribute name="submitAllowed" type="xsd:boolean" use="optional" />
</xsd:complexType>
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **emailAdapter** element:

```
<xsf:emailAdapter name="Submit" submitAllowed="yes">  
  <xsf:to value="someone@example.com" valueType="literal"/>  
  <xsf:cc value="my:ccNames" valueType="expression"/>  
  <xsf:bcc value="someoneelse@example.com" valueType="literal"/>  
  <xsf:subject value="My report" valueType="literal"/>  
  <xsf:intro value="See below"/>  
  <xsf:attachmentFileName value="Status Report" valueType="literal"/>  
</xsf:emailAdapter>
```


errorCondition Element

Defines a custom validation (or error condition) for a specific XML [Document Object Model \(DOM\)](#) node in a form's underlying XML document.

Type

xsd:complexType

Structure

| Name | Description |
|--|---|
| <u>expression</u> | (Required attribute) An <u>XPath</u> expression (relative to the expressionContext attribute, if specified) that must be evaluated to validate the XML DOM node specified in the match attribute. |
| <u>expressionContext</u> | (Optional attribute) Specifies the XML DOM node on which the expression specified in the expression attribute is rooted. |
| <u>match</u> | (Required attribute) Identifies the XML DOM nodes on which the custom validation is declared. |
| <u>showErrorOn</u> | (Optional attribute) Identifies XML DOM nodes (within the context of the expression context XML DOM node) on which the error should be displayed when the form is filled out. |
| <u>errorMessage</u> | (Required element) Specifies the error message to be returned if the value of the specified XML DOM node is considered to be invalid. |

Remarks

The **errorCondition** element is an optional element of the [customValidation](#) element.

Example

The following is an example of the **errorCondition** element:

```
<xsf:customValidation>
  <xsf:errorCondition
    match="/exp:expenseReport"
    expressionContext="exp:reportDate"
    expression="msxsl:string-compare(., ../exp:startDate) < 0 and ../ex]
    showErrorOn=".">
    <xsf:errorMessage
      type="modeless"
      shortMessage="The report date occurs before the end of the expe
      The report date occurs before the end of the expense period. Veri:
    </xsf:errorMessage>
  </xsf:errorCondition>
</xsf:customValidation>
```


errorMessage Element

Specifies the error message that should be returned if the value of the specified XML [Document Object Model \(DOM\)](#) node is considered to be invalid.

Type

xsf:xdErrorMessage

Structure

| Name | Description |
|------------------------------|--|
| shortMessage | (Required attribute) Identifies the short error message to return in case of invalid data. |
| type | (Optional attribute) Identifies the type of error message to return. |

Remarks

The **errorMessage** element is a required element of the [errorCondition](#), [override](#), and [submit](#) elements.

You can supply a detailed error message as the value of the **errorMessage** element. The detailed error message is displayed by clicking the shortcut menu's **Full error description** button when error information is displayed and when the error message **type** attribute is set to "modal".

Example

The following is an example of the **errorMessage** element:

```
<xsf:customValidation>
  <xsf:errorCondition
    match="/exp:expenseReport"
    expressionContext="exp:reportDate"
    expression="msxsl:string-compare(., ../exp:startDate) < 0 and ../ex]
    showErrorOn=".">
    <xsf:errorMessage
      type="modeless"
      shortMessage="The report date occurs before the end of the expe
      The report date occurs before the end of the expense period. Veri:
    </xsf:errorMessage>
  </xsf:errorCondition>
</xsf:customValidation>
```


exitRuleSet Element

An element that stops further **rule** processing of the entire [ruleSet](#).

Type

xsd:complexType

Parent Elements

| Element | Description |
|----------------------|--|
| rule | Defines an action invoked after an event has occurred in the form. |

Child Elements

None.

Attributes

None.

Definition

```
<xsd:element name="exitRuleSet" minOccurs="0">  
  <xsd:complexType/>  
</xsd:element>
```

Remarks

The **exitRuleSet** element must be the last child element of the **rule** element. The **ruleSet** element processing will be halted only if the **rule** element is executed.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **exitRuleSet** element:

```
<xf:rule caption="Receipts" condition="my:expense > 75">
  <xf:dialogBoxMessageAction>Don't forget your receipts!</xf:dialogBoxMessageAction>
  <xf:switchViewAction view="Approval View"/>
  <xf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression="sum(my:expenses/my:expense) * my:taxRate"/>
  <xf:queryAction adapter="Exchange Rates"/>
  <xf:submitAction adapter="Exchange Rates"/>
  <xf:openNewDocumentAction solutionURI="uri:microsoft-ExpenseForm"/>
  <xf:closeDocumentAction promptToSaveChanges="yes"/>
  <xf:dialogBoxExpressionAction>my:group/my:field1</xf:dialogBoxExpressionAction>
  <xf:exitRuleSet/>
</xf:rule>
```


exportToExcel Element

Specifies whether the user can use the [form's](#) menus, toolbars, or keyboard shortcuts to export the form to a Microsoft Office Excel 2003 workbook.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|--------------------|----------------------|----------|---|--|
| ui | xsfxdEnabledDisabled | required | Sets whether the user can export the contents of the form to an Excel workbook. | <ul style="list-style-type: none">• enabled• disabled |

Definition

```
<xsd:element name="exportToExcel" >  
  <xsd:complexType>  
    <xsd:attribute name="ui" type="xsd:boolean" use="required" />  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **exportToExcel** element is an optional element of the **featureRestrictions** element. If this element is not included in the **form definition file (.xsf)**, the user can use the form's menus, toolbars, or keyboard shortcuts to export the form to an Excel workbook.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **exportToExcel** element:

```
<xsf:featureRestrictions>  
  <xsf:save ui="disabled"/>  
  <xsf:sendMail ui="disabled"/>  
  <xsf:exportToWeb ui="disabled"/>  
  <xsf:exportToExcel ui="disabled"/>  
  <xsf:print ui="enabled"/>  
  <xsf:autoRecovery feature="disabled"/>  
</xsf:featureRestrictions>
```


exportToWeb Element

Specifies whether the user can use the [form's](#) menus, toolbars, or keyboard shortcuts to export the form to a Web page.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|--------------------|------------------------|----------|--|--|
| ui | xsfd:xdEnabledDisabled | required | Sets whether the user can export the contents of the form to a Web page. | <ul style="list-style-type: none">• enabled• disabled |

Definition

```
<xsd:element name="exportToWeb">  
  <xsd:complexType>  
    <xsd:attribute name="ui" type="xsd:boolean" use="required" />  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **exportToWeb** element is an optional element of the **featureRestrictions** element. If this element is not included in the **form definition file (.xsf)**, the user can use the form's menus, toolbars, or keyboard shortcuts to export the form to a Web page.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **exportToWeb** element:

```
<xsf:featureRestrictions>  
  <xsf:save ui="disabled"/>  
  <xsf:sendMail ui="disabled"/>  
  <xsf:exportToWeb ui="disabled"/>  
  <xsf:exportToExcel ui="disabled"/>  
  <xsf:print ui="enabled"/>  
  <xsf:autoRecovery feature="disabled"/>  
</xsf:featureRestrictions>
```


extension Element

Contains [open content model](#) information.

Type

xsd:complexType

Structure

| Name | Description |
|----------------------|---|
| name | (Required attribute) A unique name identifying the extension being specified. |

Remarks

The **extension** element is an optional element of the [extensions](#) element.

Note Use of the **extension** element is reserved. Microsoft Office InfoPath 2003 ignores any content within the **extension** element.

Example

The following is an example of the **extension** element:

```
<xsf:extensions>
  <xsf:extension
    name="someValue"
    anyAttributesHere="someValue">
    ...open content model here...
  </xsf:extension>
</xsf:extensions>
```


extensions Element

Includes minor upgrades to the Microsoft Office InfoPath 2003 [form definition \(.xsf\) file](#) that can be used by specific future versions of InfoPath or by specific forms.

Type

xsd:complexType

Structure

| Name | Description |
|---------------------------|---|
| extension | (Optional element) Contains open content model information. |

Remarks

The **extensions** element is an optional element of the [xDocumentClass](#) element.

The **extensions** element contains zero or more **extension** elements, each of which has an open content model.

Note Use of the **extension** element is reserved. Microsoft Office InfoPath 2003 ignores any content within the **extension** element.

Example

The following is an example of the **extensions** element:

```
<xsf:extensions>
  <xsf:extension
    name="someValue"
    anyAttributesHere="someValue">
    ...open content model here...
  </xsf:extension>
</xsf:extensions>
```


externalView Element

Defines a [view](#) that cannot be edited in Microsoft Office InfoPath.

Type

xsd:complexType

Child Elements

| Element | Description |
|--------------------------|---|
| mainpane | Specifies the XSL Transformation (XSLT) to be applied to the view . |

Attributes

| Attribute | Type | Required | Description | Possible Values |
|----------------------|----------------|----------|---|---|
| name | xsf:xdViewName | Yes | Contains the name of the externalView element. | minlength = 1 maxlength = 255 pattern = ([^\p{Z}\p{C}\#\&}">< (([\p{Zl}\p{Zp}]\p{C}/ <])*([\p{Z}\p{C}\#\& <]))? |

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Definition

```
<xsd:element name="externalView">  
<xsd:complexType>  
  <xsd:sequence>  
    <xsd:element ref="xsf:mainpane" />  
  </xsd:sequence>  
  <xsd:attribute name="name" type="xsf:xdViewName" use="required"  
</xsd:complexType>  
</xsd:element>
```

Example

The following is an example of the **externalView** element:

```
<xsf:externalViews>  
  <xsf:externalView name="Sales Doc">  
    <xsf:mainpane transform="myWordView.xsl" />  
  </xsf:externalView>  
</xsf:externalViews>
```


externalViews Element

Contains one or more [externalView](#) elements.

Type

xsd:complexType

Child Elements

| Element | Description |
|------------------------------|--|
| externalView | Defines a view that cannot be edited in Microsoft Office InfoPath. |

Attributes

None.

Definition

```
<xsd:element name="externalViews" >
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:externalView" minOccurs="1" maxOccurs=
    </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="externalViews_name_unique">
    <xsd:selector xpath="/xsf:externalView" />
    <xsd:field xpath="@default" />
  </xsd:unique>
  <xsd:keyref name="external_views_printView" refer="xsf:externalVi
    <xsd:selector xpath="." />
    <xsd:field xpath="@default" />
  </xsd:keyref>
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **externalViews** element:

```
<xsf:externalViews>  
  <xsf:externalView name="Sales Doc">  
    <xsf:mainpane transform="myWordView.xsl" />  
  </xsf:externalView>  
</xsf:externalViews>
```


featureRestrictions Element

Specifies whether the user can use the [form's](#) menus, toolbars, or keyboard shortcuts to save the form, export the form, print the form, or send the form as an e-mail attachment.

Type

xsd:complexType

Child Elements

| Element | Description |
|--------------------------------------|--|
| <u>save</u> | (Optional element) Specifies whether the user can use the form's menus, toolbars, or keyboard shortcuts to save the form. |
| <u>exportToWeb</u> | (Optional element) Specifies whether the user can use the form's menus, toolbars, or keyboard shortcuts to export the form to a Web page. |
| <u>exportToExcel</u> | (Optional element) Specifies whether the user can use the form's menus, toolbars, or keyboard shortcuts to export the form to an Microsoft Office Excel 2003 workbook. |
| <u>print</u> | (Optional element) Specifies whether the user can use the form's menus, toolbars, or keyboard shortcuts to print the form. |
| <u>sendMail</u> | (Optional element) Specifies whether the user can use the form's menus, toolbars, or keyboard shortcuts to send the form as an e-mail attachment. |
| <u>autoRecovery</u> | (Optional element) Specifies whether the form will save AutoRecover information and whether the AutoRecover setting can be changed by the user. |

Attributes

None.

Definition

```
<xsd:element name="featureRestrictions">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="save" minOccurs="0">
        <xsd:complexType>
          <xsd:attribute name="ui" type="xsd:boolean" use="required"/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element ref="xsd:exportToWeb" minOccurs="0" />
      <xsd:element ref="xsd:exportToExcel" minOccurs="0" />
      <xsd:element ref="xsd:print" minOccurs="0" />
      <xsd:element ref="xsd:sendMail" minOccurs="0" />
      <xsd:element ref="xsd:autoRecovery" minOccurs="0" />
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

Remarks

Disabling any of the **featureRestrictions** element's child elements does not disable the use of form code to save the form, export the form, print the form, or send the form as an email attachment.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **featureRestrictions** element:

```
<xsf:featureRestrictions>  
  <xsf:save ui="disabled"/>  
  <xsf:sendMail ui="disabled"/>  
  <xsf:exportToWeb ui="disabled"/>  
  <xsf:exportToExcel ui="disabled"/>  
  <xsf:print ui="enabled"/>  
  <xsf:autoRecovery feature="disabled"/>  
</xsf:featureRestrictions>
```


field Element

Defines one field for form library columns.

Type

xsd:complexType

Structure

| Name | Description |
|------------------------------------|---|
| <u>aggregation</u> | (Optional attribute) Specifies how the XML Document Object Model (DOM) nodes returned from an XPath expression in the node attribute should be aggregated to obtain a single value for the document. Can either be an aggregation action or an indication of the particular element in the collection. |
| <u>columnName</u> | (Required attribute) Identifies the column name in the SQL table (underlying the form list view). |
| <u>maxLength</u> | (Optional attribute) Defines the length of the field in number of bytes. |
| <u>name</u> | (Required attribute) Identifies the friendly name of the field to be used on the form list view. |
| <u>node</u> | (Required attribute) Defines the XPath expression needed to extract the value of the specified property from the form's underlying XML document. |
| <u>required</u> | (Optional attribute) Specifies whether this field accepts null values. |
| <u>type</u> | (Required attribute) Identifies the standard XML Schema data type. |
| <u>viewable</u> | (Optional attribute) Specifies whether this field should be added to the default view. Possible values "yes" and "no". The default value is "no". |

Remarks

The **field** element is an optional element of the [fields](#) element.

Example

The following is an example of the **field** element:

```
<xsf:listProperties>
  <xsf:fields>
    <xsf:field
      type="xsd:date"
      name="TravelDate"
      columnName="TravelDate"
      required="yes"
      viewable="yes"
      node="TravelReport/Header/travelDate"
      aggregation="first"/>
  </xsf:fields>
</xsf:listProperties>
```

field Element (sharepointListAdapter Element)

Contains field mapping information for each field in a SharePoint list and the corresponding name used in InfoPath.

Type

xsd:ComplexType

Parent Elements

| Element | Description |
|---------------------------------------|--|
| sharepointListAdapter | Contains the data adapter information to query a SharePoint list or library. |

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|---------------------------------------|--------------------------------|----------|--|--|
| <u>sharepointName</u> | xsd:string | Yes | Contains the name of a string field in a SharePoint list. | |
| <u>infopathName</u> | xsd:string | Yes | Contains the corresponding InfoPath field name for the sharepointName . | string |
| <u>isLookup</u> | <u>xdYesNo</u> | No | Specifies whether a field in a SharePoint list is a lookup field. The default is "no". | <ul style="list-style-type: none">• yes• no |

Definition

```
<xsd:element name="field" minOccurs="0" maxOccurs="unbounded":  
  <xsd:complexType>  
    <xsd:attribute name="sharepointName" type="xsd:string" use="requ  
    <xsd:attribute name="infopathName" type="xsd:string" use="requir  
    <xsd:attribute name="isLookup" type="xsd:boolean" use="optiona  
  </xsd:complexType>  
</xsd:element>
```

Remarks

Each field returned from a SharePoint list or library by the **sharepointListAdapter** data adapter will have a **field** element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **field** element:

```
<xsf:field  
  sharepointName="xd__x007b_D00F1DBD_..."  
  infopathName="Title_1"  
  isLookup="no">  
</xsf:field>
```

fields Element

Defines a set of one or more [field](#) elements.

Type

xsd:complexType

Structure

| Name | Description |
|-----------------------|--|
| field | (Optional element) Defines one field for form library columns. |

Remarks

The **fields** element is an optional element of the [listProperties](#) element.

Example

The following is an example of the **fields** element:

```
<xsf:listProperties>
  <xsf:fields>
    <xsf:field
      type="xsd:date"
      name="TravelDate"
      columnName="TravelDate"
      required="yes"
      viewable="yes"
      node="TravelReport/Header/travelDate"
      aggregation="first"/>
  </xsf:fields>
</xsf:listProperties>
```

file Element

Identifies a file as part of a Microsoft Office InfoPath 2003 form.

Type

xsd:complexType

Structure

| Name | Description |
|--------------------------------|--|
| name | (Required attribute) Specifies the name of the file. |
| fileProperties | (Optional element) Defines the properties of a file. |

Remarks

The **file** element is an optional element of the [files](#) element.

Example

The following is an example of the **file** element:

```
<xsf:package>
  <xsf:files>
    <xsf:file name="view_1.xsl">
      <xsf:fileProperties>
        <xsf:property
          name="lang"
          type="string"
          value="1033"/>
      </xsf:fileProperties>
    </xsf:file>
  </xsf:files>
</xsf:package>
```


fileName Element

Specifies the file name or an expression that returns a file name when the form is submitted using the [davAdapter](#).

Type

xsd:complexType

Parent Elements

| Element | Description |
|----------------------------|---|
| davAdapter | The parent element that contains all the information necessary to submit files to a server that is running Microsoft Windows SharePoint Services or a Web based Distributed Authoring and Versioning (WebDAV) server. |

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|---------------------------|-------------------------|----------|--|-----------------|
| value | xsd:string | Yes | Contains the name of the file, once submitted to the server. | string |
| valueType | xsf:xdExpressionLiteral | No | Specifies whether the name of the file should be interpreted as an XPath expression or as a literal string. <ul style="list-style-type: none">• expression• literal | |

Definition

```
<xsd:element name="fileName" >
  <xsd:complexType>
    <xsd:attribute name="value" type="xsd:string" use="required"/></xsd:
    <xsd:attribute name="valueType" type="xsd:string" use="optional" /></xsd:
  </xsd:complexType>
</xsd:element>
```

Remarks

The extension ".xml" is appended to the file name if a file name extension is not specified.

The following system reserved characters will be replaced by an underscore ("_") character when a form is submitted:

\\ : * ? " < > |

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **fileName** element:

```
<xsf:fileName value="my:myFields/my:fileName" valueType="expres
```


fileNew Element

Provides a reference to an .xml file containing sample data to be loaded when a user chooses to create a new form based on the [form template](#).

Type

xsd:complexType

Structure

| Name | Description |
|------------------------------------|--|
| initialXmlDocument | (Required element) Contains a reference to the XML template file that is used for the creation of a new form based on the form template. |

Remarks

The **fileNew** element is an optional element of the [xDocumentClass](#) element.

The **fileNew** element of the Microsoft Office InfoPath 2003 [form definition \(.xsf\) file](#) defines the name and location of an XML template file that is used when a user clicks **Fill Out a Form** on the **File** menu. The XML template file contains the sample data that is loaded when the user chooses to create a new form based on the form template.

Example

The following is an example of the **fileNew** element:

```
<xsf:fileNew>  
  <xsf:initialXmlDocument  
    caption="Travel Report"  
    href="TravelReportTemplate.xml"/>  
</xsf:fileNew>
```

fileProperties Element

Defines the properties of a file.

Type

xsd:complexType

Structure

| Name | Description |
|--------------------------|--|
| property | (Optional element) Defines one specific property for the specified file. |

Remarks

The **fileProperties** element is an optional element of the [file](#) element.

Example

The following is an example of the **fileProperties** element:

```
<xsf:package>
  <xsf:files>
    <xsf:file name="view_1.xsl">
      <xsf:fileProperties>
        <xsf:property
          name="lang"
          type="string"
          value="1033"/>
      </xsf:fileProperties>
    </xsf:file>
  </xsf:files>
</xsf:package>
```

files Element

Identifies a list of files that are used by a Microsoft Office InfoPath 2003 form.

Type

xsd:complexType

Structure

| Name | Description |
|----------------------|---|
| file | (Optional element) Identifies a file as part of an InfoPath form. |

Remarks

The **files** element is a required element of the [package](#) element.

The **files** element may include specific property names and values for each file.

Example

The following is an example of the **files** element:

```
<xsf:package>
  <xsf:files>
    <xsf:file name="view_1.xsl">
      <xsf:fileProperties>
        <xsf:property
          name="lang"
          type="string"
          value="1033"/>
      </xsf:fileProperties>
    </xsf:file>
  </xsf:files>
</xsf:package>
```


folderURL Element

Specifies the [Uniform Resource Locator \(URL\)](#) of a [Web-based Distributed Authoring and Versioning \(WebDAV\)](#) server or a server that is running Microsoft Windows SharePoint Services.

Type

xsd:complexType

Parent Elements

| Element | Description |
|----------------------------|--|
| davAdapter | The parent element, which contains all information necessary to submit Infopath forms to a Web-based Distributed Authoring and Versioning (WebDAV) server or a server that is running SharePoint Services. |

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|-----------------------|------------|----------|--|-----------------|
| value | xsd:string | Yes | The URL of the Web-based Distributed Authoring and Versioning (WebDAV) server or the server that is running SharePoint Services. | string |

Definition

```
<xsd:element name="folderURL">  
  <xsd:complexType>  
    <xsd:attribute name="value" type="xsd:string" use="required"/></xsd:  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The URL must begin with "http://" or "https://". Other common protocols will cause an error.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **folderURL** element:

```
<folderURL value="http://some_server/some_doc_lib"/>
```

footer Element

Specifies the footer text.

Type

xsd:simpleType

Child Elements

None.

Attributes

None.

Definition

```
<xsd:element name="footer" >  
  <xsd:complexType mixed="true">  
    <xsd:sequence>  
      <xsd:any minOccurs="0" maxOccurs="unbounded" processConter  
    </xsd:sequence>  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **footer** element is a child element of the [printSettings](#) element. The **footer** text should not exceed 255 characters.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **footer** element:

```
<xf:printSettings
  orientation="landscape"
  header="&Pqsdf"
  footer="&D"
  printerName="\\printserver\printer"
  paperSource="Auto Select"
  paperSize="Envelope DL"
  topMargin="0.8"
  leftMargin="0.8"
  bottomMargin="0.8"
  rightMargin="0.8"
  marginUnitsType="in"
  copies="2"
  collate="no"
  pageRangeStart="1"
  pageRangeEnd="1" >
  <xf:header>
    <font>
      <div>&Pqsdf</div>
    </font>
  </xf:header>
  <xf:footer>
    <font>
      <div>&D</div>
    </font>
  </xf:footer>
</xf:printSettings>
```

fragmentToInsert Element

Contains alternate versions of XML data (fragments).

Type

xsd:complexType

Structure

| Name | Description |
|--------------------------------|---|
| chooseFragment | (Required element) Specifies an XML fragment. |

Remarks

The **fragmentToInsert** element is an optional element of the [editWith](#) element.

XML fragments are sources of XML data that can be used within an associated editing component. More than one **chooseFragment** element can be defined within a **fragmentToInsert** element.

The **fragmentToInsert** element is used by the xCollection, xOptional, and xReplace editing components.

Example

The following is an example of the **fragmentToInsert** element:

```
<xsf:editing>
  <xsf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xsf:editWith caption="CD"
      xd:autogeneration="template"
      component="xCollection">
      <xsf:fragmentToInsert>
        <xsf:chooseFragment parent="CDCollection">
          <CD>
            <Title></Title>
            <Artist></Artist>
            <Tracks>
              <Track></Track>
              <Track></Track>
            </Tracks>
          </CD>
        </xsf:chooseFragment>
      </xsf:fragmentToInsert>
    </xsf:editWith>
  </xsf:xmlToEdit>
</xsf:editing>
```


getUserNameFromData Element

Allows user names to be determined by an [XPath](#) query into the data in the main data source or a secondary data source and to be associated with a role.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|----------------------------|------------|----------|---|-----------------|
| dataObject | xsd:string | No | The name of the secondary data source where the user name can be found. | string |
| select | xsd:string | Yes | An XPath query expression returning one or more data nodes containing the user names. | string |
| memberOf | xsd:string | Yes | Specifies the role to be associated with a user whose user name is returned by the XPath query. | string |

Definition

```
<xsd:element name="getUserNameFromData">
  <xsd:complexType>
    <xsd:attribute name="dataObject" type="xsd:string" use="optional"/>
    <xsd:attribute name="select" type="xsd:string" use="required"/></xsd:attribute>
    <xsd:attribute name="memberOf" type="xsd:string" use="required"/></xsd:attribute>
  </xsd:complexType>
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **getUserUserNameFromData** element:

```
<xsf:roles initiator="A" default="C" hideStatusBarDisplay="yes">
  <xsf:role name="A" />
  <xsf:role name="B" />
  <xsf:role name="C" />

  <xsf:membership>
    <xsf:getUserUserNameFromData dataObject="catalog" select="/dfs:
    <xsf:userName name="Domain\username1" memberOf="A" />
    <xsf:userName name="Domain\username2" memberOf="B" />
    <xsf:group name="Domain\username3" memberOf="C" />
  </xsf:membership>
</xsf:roles>
```

group Element

Associates a group with a particular role.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|--------------------------|------------|----------|---|-----------------|
| name | xsd:string | Yes | Specifies users in the form of "domain\groupname" or "groupname". | string |
| memberOf | xsd:string | Yes | Specifies the role to be associated with the group. | string |

Definition

```
<xsd:element name="group">  
  <xsd:complexType>  
    <xsd:attribute name="name" type="xsd:string" use="required"/></xsd:  
    <xsd:attribute name="memberOf" type="xsd:string" use="required"  
  </xsd:complexType>  
</xsd:element>
```

Remarks

Users can be assigned roles through grouping that can be managed externally, without requiring a form update. Groups can be set up through the Active Directory directory service. Permissions specifying who can access the membership information can be set for groups.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **group** element:

```
<xsf:roles initiator="A" default="C" hideStatusBarDisplay="yes">
  <xsf:role name="A"/>
  <xsf:role name="B"/>
  <xsf:role name="C"/>

  <xsf:membership>
    <xsf:getUserNameFromData dataObject="catalog"
      select="/dfs:myFields/dfs:dataFields/d:UserA" memberOf="B"/>
    <xsf:userName name="Domain\username1" memberOf="A"/>
    <xsf:userName name="Domain\username2" memberOf="B"/>
    <xsf:group name="Domain\username3" memberOf="C"/>
  </xsf:membership>
</xsf:roles>
```

header Element

Specifies the header text.

Type

xsd:simpleType

Child Elements

None.

Attributes

None.

Definition

```
<xsd:element name="header" >  
  <xsd:complexType mixed="true">  
    <xsd:sequence>  
      <xsd:any minOccurs="0" maxOccurs="unbounded" processConter  
    </xsd:sequence>  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **header** element is a child element of the [printSettings](#) element. The **header** text should not exceed 255 characters.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **header** element:

```
<xsf:printSettings
  orientation="landscape"
  header="&Pqsdf"
  footer="&D"
  printerName="\\printserver\printer"
  paperSource="Auto Select"
  paperSize="Envelope DL"
  topMargin="0.8"
  leftMargin="0.8"
  bottomMargin="0.8"
  rightMargin="0.8"
  marginUnitsType="in"
  copies="2"
  collate="no"
  pageRangeStart="1"
  pageRangeEnd="1" >
  <xsf:header>
    <font>
      <div>&Pqsdf</div>
    </font>
  </xsf:header>
  <xsf:footer>
    <font>
      <div>&D</div>
    </font>
  </xsf:footer>
</xsf:printSettings>
```


hwsAdapter Element

Defines the Microsoft BizTalk 2004 Human Workflow Services (HWS) data adapter, which can be used to start or extend an activity flow and respond to a task.

Type

xsd:complexType

Child Elements

| Element | Description |
|------------------------------|---|
| hwsOperation | Defines the HWS operation type, such as adding an action to a new activity flow, adding an action to an existing activity flow, and responding to a task. |

Attributes

| Attribute | Type | Required | Description | Possible Values |
|-------------------------------|-------------------------|----------|--|--|
| name | xdTitle | Yes | Contains the name of the data adapter. | minLength = 1 maxLength = 255 pattern = ([^p{Z}\p{Cc}\p{Cf}\p (([^p{Zl}\p{Zp}\p{Cc} ([^p{Z}\p{Cc}\p{Cf}\p |
| wsdlUrl | xsd:string | Yes | Contains the Uniform Resource Locator (URL) of the HWS Web service. | string |
| queryAllowed | xdYesNo | No | Specifies whether the adapter can be used for querying the data source. Can be omitted for the hwsAdapter , corresponding to a default value of "no". | <ul style="list-style-type: none"> • yes • no |
| submitAllowed | xdYesNo | No | Specifies whether the adapter can be used for submitting to the data | <ul style="list-style-type: none"> • yes • no |

source.
Always set to
"yes" for the
hwsAdapter.

Definition

```
<xsd:element name="hwsAdapter">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="xsf:hwsOperation"/>
    </xsd:choice>
    <xsd:attribute name="name" type="xsf:xdTitle" use="required"/></x
    <xsd:attribute name="wsdlUrl" type="xsd:string" use="required"><
    <xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="opt
    <xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="opt
  </xsd:complexType>
</xsd:element>
```

Remarks

Upon submit, the **hwsAdapter** automatically creates a globally unique identifier (GUID), as required by the HWS Web service; encodes the XML file; and updates the [processing instructions](#) of the XML instance file.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **hwsAdapter** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval"/>
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate"/>
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response"/>
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response"/>
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond"/>
  </xsf:allowedTasks>
</xsf:hwsWorkflow>
```

```
<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID="
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1"/>
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/a:some/b:thing" dataObject="Aux1"/>
```

```
</xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```


hwsOperation Element

Defines the Microsoft BizTalk 2004 Human Workflow Services (HWS) operation type, such as adding an action to a new activity flow, adding an action to an existing activity flow, and responding to a task.

Type

xsd:complexType

Child Elements

| Element | Description |
|-----------------------|---|
| input | Specifies the XML form file, which is encoded and submitted with the HWS operation. |

Attributes

| Attribute | Type | Required | Description | Possible Values |
|----------------------------|------------|----------|---|---|
| type | xsd:string | Yes | Specifies the HWS operation type. | <ul style="list-style-type: none">• addActionToNewActivityFlow• addActionToActivityFlow• sendTaskResponse |
| typeID | xsd:string | Yes | Contains the globally unique identifier (GUID) for the operation. | string |
| serviceUrl | xsd:string | Yes | Specifies the Uniform Resource Locator (URL) location of the HWS Web service. | string |

Definition

```
<xsd:element name="hwsOperation">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="xsf:input"/>
    </xsd:choice>
    <xsd:attribute name="type" type="xsd:string" use="required"></xsd:attribute>
    <xsd:attribute name="typeID" type="xsd:string" use="required"></xsd:attribute>
    <xsd:attribute name="serviceUrl" type="xsd:string" use="required"></xsd:attribute>
  </xsd:complexType>
</xsd:element>
```

Remarks

Upon submit, the **hwsAdapter** element automatically creates a globally unique identifier (GUID), as required by the HWS Web service; encodes the XML file; and updates the [processing instructions](#) of the XML instance file with the new GUID.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **hwsOperation** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval"/>
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate"/>
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response"/>
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response"/>
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond"/>
  </xsf:allowedTasks>
</xsf:hwsWorkflow>

<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID=
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1"/>
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/a:some/b:thing" dataObject="Aux1"/>
```

```
</xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```


hwsWorkflow Element

Contains the information to enable the **Workflow** task pane and to enable individual actions and tasks associated with a Microsoft BizTalk 2004 Human Workflow Services (HWS) server.

Type

xsd:complexType

Child Elements

| Element | Description |
|---------------------------------------|---|
| <u>location</u> | (Required) Contains the <u>Uniform Resource Locator (URL)</u> of the HWS Web service. |
| <u>allowedActions</u> | Contains the HWS actions enabled for the form. |
| <u>allowedTasks</u> | Contains the HWS tasks enabled for the form. |

Attributes

| Attribute | Type | Required | Description | Possible Values |
|---------------------------------|-------------------------|----------|---|--|
| taskpaneVisible | xdYesNo | No | Specifies whether the Workflow task pane is visible. Default value is "yes". | <ul style="list-style-type: none">• yes• no |

Definition

```
<xsd:element name="hwsWorkflow">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:location" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="xsf:allowedActions" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="xsf:allowedTasks" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="taskpaneVisible" type="xsf:xdYesNo"/>
  </xsd:complexType>
  <xsd:unique name="hws_actiontask_name">
    <xsd:selector xpath="/xsf:allowedActions/xsf:action|/xsf:allowedTasks/xsf:allowedTask"/>
    <xsd:field xpath="@name"/>
  </xsd:unique>
</xsd:element>
```

Remarks

For every action and task included in the **hwsWorkflow** element, a button is enabled on the **Workflow** task pane; however, the button event must be scripted manually for each action and task.

InfoPath does not support multiple actions in a form and does not support HWS activity models (predefined sets of actions).

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **hwsWorkflow** element:

```
<xf:hwsWorkflow taskpaneVisible="yes">
  <xf:location url="http://www.contoso.com/hwsservice/hwsservice.as
  <xf:allowedActions>
    <xf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval"/>
    <xf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate"/>
  </xf:allowedActions>
  <xf:allowedTasks>
    <xf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response"/>
    <xf:task name="getVPAapproval" taskTypeID="436"
      caption="Send Response"/>
    <xf:task name="delegateToManager" taskTypeID="420"
      caption="Respond"/>
  </xf:allowedTasks>
</xf:hwsWorkflow>
```


importParameters Element

Contains all the parameters that define how the import (merge) forms feature works for the form.

Type

xsd:complexType

Structure

| Name | Description |
|---|--|
| <u>enabled</u> | (Required attribute) Specifies whether form merging is enabled for the form. |
| <u>importSource</u> | (Optional element) Specifies all parameters to be used when merging a form of a specific XML Schema into a destination form. |
| <u>useScriptHandler</u> | (Optional attribute) Specifies whether to use the event handler defined for the OnMergeRequest event when importing (merging) forms. |

Remarks

The **importParameters** element is an optional element of the [xDocumentClass](#) element.

Example

The following is an example of the **importParameters** element:

```
<xsf:importParameters
  enabled="yes"
  useScriptHandler="yes">
  <xsf:importSource
    name="MySource"
    schema="MySchema.xsd"
    transform="schematransform.xslt"/>
</xsf:importParameters>
```


importSource Element

Specifies all parameters to be used when merging a form of a specific [XML Schema](#) into a destination form.

Type

xsd:complexType

Structure

| Name | Description |
|----------------------------------|--|
| <u>name</u> | (Required attribute) Identifies the name of the source form as defined in the <u>processing instruction</u> of that form's underlying XML document. |
| <u>schema</u> | (Required attribute) Identifies the XML Schema file that should be used during the merge operation to validate the form being merged. |
| <u>transform</u> | (Required attribute) Identifies the <u>.xslt</u> file that should be used during the merge operation when the source form (the one that is being merged in) matches the XML Schema specified in the corresponding schema attribute. |

Remarks

The **importSource** element is an optional element of the [importParameters](#) element.

If the **importSource** element is not defined, the default .xslt file is used for all transformations during a merge operation.

Example

The following is an example of the **importSource** element:

```
<xsf:importParameters
  enabled="yes"
  <xsf:importSource
    name=""
    schema="MySchema.xsd"
    transform="schematransform.xslt"/>
</xsf:importParameters>
```


initialXmlDocument Element

Contains a reference to the [XML template file](#) that is used for the creation of a new form based on the [form template](#).

Type

xsd:complexType

Structure

| Name | Description |
|---------------------------------------|---|
| <u>caption</u> | (Required attribute) Defines the text string to be used as the name of the form in the Template Gallery and in the most recently used list. |
| <u>customCategory</u> | (Optional element) Specifies the category that the <u>form template</u> appears under in the Fill Out a Form task pane. |
| <u>href</u> | (Required attribute) Specifies the <u>Uniform Resource Locator (URL)</u> of the XML template file to be used when a user clicks Fill Out a Form on the File menu. |

Remarks

The **initialXmlDocument** element is a required element of the [fileNew](#) element.

Example

The following is an example of the **initialXmlDocument** element:

```
<xsf:fileNew>  
  <xsf:initialXmlDocument  
    caption="Travel Report"  
    href="TravelReportTemplate.xml"/>  
</xsf:fileNew>
```


input Element

Contains the substitution information for parts of the input Simple Object Access Protocol (SOAP) message to the Web service.

Type

xsd:complexType

Structure

| Name | Description |
|------------------------------|--|
| source | (Required attribute) Contains the name of the resource file in the form template that contains the XML Schema for the input SOAP message of the selected operation of the Web service. |
| partFragment | (Optional element) Defines one substitution group for a specific part of the input SOAP message. |

Remarks

The **input** element is an optional element of the [operation](#) element.

Specified parts in the SOAP message are replaced when the form template is filled out with data from within the form. It is used when a [secondary data source](#) is populated from a Web service call and Microsoft Office InfoPath 2003 needs some input arguments to make the Web service calls.

Example

The following is an example of the **input** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval" />
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate" />
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response" />
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response" />
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond" />
  </xsf:allowedTasks>
</xsf:hwsWorkflow>

<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID=
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1" />
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/a:some/b:thing" dataObject="Aux1" />
```

```
</xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```

intro Element

Contains the introduction for the e-mail message when the form is submitted by using the [emailAdapter](#) element.

Type

xsd:complexType

Parent Elements

| Element | Description |
|------------------------------|---|
| emailAdapter | Parent element that contains the information needed to submit an InfoPath form as an attachment to an e-mail message, with a specified set of recipients, a subject, and an introduction. |

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|-----------------------|------------|----------|---|-----------------|
| value | xsd:string | Yes | Contains the value of the intro element. | string |

Definition

```
<xsd:element name="intro" minOccurs="0">  
  <xsd:complexType>  
    <xsd:attribute name="value" type="xsd:string" use="required"/></xsd:  
  </xsd:complexType>  
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **intro** element:

```
<xsf:emailAdapter name="Submit" submitAllowed="yes">  
  <xsf:to value="someone@example.com" valueType="literal"/>  
  <xsf:cc value="my:ccNames" valueType="expression"/>  
  <xsf:bcc value="someoneelse@example.com" valueType="literal"/>  
  <xsf:subject value="My report" valueType="literal"/>  
  <xsf:intro value="See below"/>  
  <xsf:attachmentFileName value="Status Report" valueType="literal"/>  
</xsf:emailAdapter>
```


listProperties Element

Identifies the properties that should be on a list view of all forms belonging to the [form template](#).

Type

xsd:complexType

Structure

| Name | Description |
|------------------------|---|
| fields | (Optional element) Defines a set of one or more field elements. |

Remarks

The **listProperties** element is an optional element of the [xDocumentClass](#) element.

XML documents belonging to a form can be placed in a single folder or form library. Depending on the underlying support in the file system or server, this information can be used to create meaningful list views on a set of forms. For example, when Microsoft Office InfoPath 2003 forms are saved to a Windows SharePoint Services form library that is based on an InfoPath form template, form properties specified in this section are automatically promoted and made available to the default view of the form library.

Example

The following is an example of the **listProperties** element:

```
<xsf:listProperties>
  <xsf:fields>
    <xsf:field
      type="xsd:date"
      name="TravelDate"
      columnName="TravelDate"
      required="yes"
      viewable="yes"
      node="TravelReport/Header/travelDate"
      aggregation="first"/>
  </xsf:fields>
</xsf:listProperties>
```


location Element

Contains the [Uniform Resource Locator \(URL\)](#) of the Microsoft BizTalk 2004 Human Workflow Services (HWS) Web service.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|---------------------|------------|----------|--------------------------------------|-----------------|
| url | xsd:string | Yes | The location of the HWS Web service. | string |

Definition

```
<xsd:element name="location">  
  <xsd:complexType>  
    <xsd:attribute name="url" type="xsd:string" use="required"></xsd:  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **location** element is a child element of the [hwsWorkflow](#) element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **location** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.as
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval"/>
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate"/>
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response"/>
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response"/>
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond"/>
  </xsf:allowedTasks>
</xsf:hwsWorkflow>
```


mainpane Element

Determines what is displayed in the main pane.

Type

xsd:complexType

Structure

| Name | Description |
|---------------------------|---|
| transform | (Required attribute) Specifies the relative URL to the XSL Transformation (XSLT) that is used for the view. |

Remarks

The **mainpane** element is a required element of the [view](#) element.

The **mainpane** element is an optional element of the [externalView](#) element.

The main pane is the main [form area](#), as opposed to secondary user interface areas such as the task pane.

Example

The following is an example of the **mainpane** element:

```
<xsf:views default="View">
  <xsf:view name="View" caption="View">
    <xsf:mainpane transform="view1.xsl"></xsf:mainpane>
    ...
  </xsf:view>
</xsf:views>
```

The following is an example of the **mainpane** element used within the [externalView](#) element:

```
<xsf:externalViews>
  <xsf:externalView name="Sales Doc">
    <xsf:mainpane transform="myWordView.xsl" />
  </xsf:externalView>
</xsf:externalViews>
```


masterDetail Element

Defines the XML fragments that form a master and detail relationship in a view's [repeating tables](#) or [repeating sections](#).

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|-----------------------------------|------------|----------|--|-----------------|
| master | xsd:string | No | Contains the XPath of the XML fragment that is bound to a master table or section. | string |
| masterViewContext | xsd:string | No | Specifies a string that identifies an HTML element in the view. | string |
| masterKey | xsd:string | No | Contains the XPath of the field in the master XML fragment that forms the relationship to the detail XML fragment. | string |
| detailKey | xsd:string | No | Contains the XPath of the field in the detail XML fragment that forms the relationship to the master XML fragment. | string |

Definition

```
<xsd:element name="masterDetail" >
  <xsd:complexType>
    <xsd:attribute name="master" type="xsd:string"></xsd:attribute>
    <xsd:attribute name="masterViewContext" type="xsd:string"></xsd:attribute>
    <xsd:attribute name="masterKey" type="xsd:string"></xsd:attribute>
    <xsd:attribute name="detailKey" type="xsd:string"></xsd:attribute>
  </xsd:complexType>
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **masterDetail** element:

```
<xsf:editWith caption="group2"
  xd:autogeneration="template"
  component="xCollection">
  <xsf:masterDetail
    masterViewContext="CTRL1_5"
    master="my:group2"
    masterKey="my:field1"
    detailKey="my:field3">
  </xsf:masterDetail>
</xsf:editWith>
```


membership Element

Associates a user or group of users with a role.

Type

xsd:complexType

Child Elements

| Element | Description |
|-------------------------------------|---|
| getUserNameFromData | Allows user names to be determined by an XPath query into the data in the main data source or a secondary data source and to be associated with a role. |
| userName | Defines a user to a particular role. |
| group | Defines which group a user belongs to. |

Attributes

None.

Definition

```
<xsd:element name="membership">
  <xsd:complexType>
    <xsd:choice minOccurs="1" maxOccurs="unbounded">
      <xsd:element ref="xsf:getUserNameFromData"/>
      <xsd:element ref="xsf:userName"/>
      <xsd:element ref="xsf:group"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

Remarks

InfoPath associates a role with the current user based on the membership order in the [form definition file \(.xsf\)](#). Developers can modify this order by hand in the form definition file. The **membership** element must be a child of the [roles](#) element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **membership** element:

```
<xsf:roles initiator="A" default="C" hideStatusBarDisplay="yes">
```

```
  <xsf:role name="A"/>
```

```
  <xsf:role name="B"/>
```

```
  <xsf:role name="C"/>
```

```
  <xsf:membership>
```

```
    <xsf:getUserNameFromData dataObject="catalog" select="/dfs:my
```

```
    <xsf:userName name="Domain\username1" memberOf="A"/>
```

```
    <xsf:userName name="Domain\username2" memberOf="B"/>
```

```
    <xsf:group name="Domain\username3" memberOf="C"/>
```

```
  </xsf:membership>
```

```
</xsf:roles>
```

menu Element

Contains information about the custom menus used in the view.

Type

xsd:complexType

Structure

| Name | Description |
|-------------------------|---|
| button | (Optional element) Defines a button that has an associated action. |
| caption | (Required attribute) Used as the caption for a menu. |
| menu | (Optional element) Contains information about the menus used within a menu (cascading menus). |

Remarks

The **menu** element is an optional element of the [view](#), [toolbar](#), and [menuArea](#) elements. Multiple menus can be declared for a form, and each menu can contain multiple **menu** or **button** elements.

Note To create cascading menus, a **menu** element can be nested inside another **menu** element.

Example

The following is an example of the **menu** element:

```
<xsf:menuArea name="msoInsertMenu">
  <xsf:menu caption="&Section">
    <xsf:button action="xCollection::insert" xmlToEdit="CD_10"
      caption="CD" showIf="always"></xsf:button>
    <xsf:button action="xCollection::insert" xmlToEdit="Track_14"
      caption="Track" showIf="always"></xsf:button>
    <xsf:button action="xOptional::insert" xmlToEdit="Label_16"
      caption="Label"></xsf:button>
  </xsf:menu>
</xsf:menuArea>
```

menuArea Element

Contains information about the Microsoft Office InfoPath 2003 built-in menus used in the view.

Type

xsd:complexType

Structure

| Name | Description |
|------------------------|--|
| button | (Optional element) Defines a button that has an associated action. |
| name | (Required attribute) Corresponds to one of the built-in InfoPath top-level menus. |
| menu | (Optional element) Contains information about the menus used within the menu area. |

Remarks

The **menuArea** element is an optional element of the [view](#) element.

The **menuArea** element is analogous to the top-level menus found in InfoPath when filling out a form. Multiple buttons or menus can be declared within a **menuArea** element. Each **button** element creates an additional menu item within the corresponding built-in menu, specified by the **name** attribute of the **menuArea** element, and has an action (or command) associated with it. If a **menu** element is nested within a **menuArea** element, this creates a cascading menu off of the built-in menu.

Example

The following is an example of the **menuArea** element:

```
<xf:menuArea name="msoInsertMenu">
  <xf:menu caption="&Section">
    <xf:button action="xCollection::insert" xmlToEdit="CD_10"
      caption="CD" showIf="always"></xf:button>
    <xf:button action="xCollection::insert" xmlToEdit="Track_14"
      caption="Track" showIf="always"></xf:button>
    <xf:button action="xOptional::insert" xmlToEdit="Label_16"
      caption="Label"></xf:button>
  </xf:menu>
</xf:menuArea>
```

message Element

Contains the signature confirmation message that is shown when a signature is applied to the form or section of the form.

Type

xsf:xdSignedDataBlockMessage

Parent Elements

| Element | Description |
|---------------------------------|--|
| signedDataBlock | Defines a node set in the form's underlying XML document, to which a digital signature can be applied. |

Child Elements

None.

Attributes

None.

Definition

```
<xsd:element name="message" type="xsf:xdSignedDataBlockMessage"/>
```

Remarks

The confirmation message is limited to 255 characters.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **message** element:

```
<xsf:documentSignatures>
  <xsf:signedDataBlock name="main"
    data="my:myfields/my:subtree1 | my:myfields/my:subtree2"
    signatureLocation="my:mifields/sig:signatures/main"
    mode="countersign">
    <xsf:message>By pressing the &quot;Sign&quot; button below, I agree
      to the terms of this document.
    </xsf:message>
  </xsf:signedDataBlock>
</xsf:documentSignatures>
```


onLoad Element

Contains one [ruleSet](#) element that is invoked when the [form](#) is opened.

Type

xsd:complexType

Child Elements

| Element | Description |
|-------------------------------|--|
| ruleSetAction | Defines the ruleSet action element to be invoked. |

Attributes

None.

Definition

```
<xsd:element name="onLoad">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:ruleSetAction" minOccurs="1" maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:keyref name="load_ruleSetAction" refer="xsf:ruleset_name_key" />
  <xsd:selector xpath="./xsf:ruleSetAction"/>
  <xsd:field xpath="@ruleSet"/>
</xsd:keyref>
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **onLoad** element:

```
<xsf:onLoad>  
  <xsf:ruleSetAction ruleSet="RuleSet4"/>  
</xsf:onLoad>
```


openNewDocumentAction Element

Defines a [form](#) create action.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|-----------------------------|------------|----------|--|-----------------|
| solutionURI | xsd:anyURI | Yes | The Uniform Resource Identifier (URI) of the solution on which the new form will be based. | anyURI |

Definition

```
<xsd:element name="openNewDocumentAction">  
  <xsd:complexType>  
    <xsd:attribute name="solutionURI" type="xsd:anyURI" use="requi  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **openNewDocumentAction** element is a child element of the [rule](#) element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **openNewDocumentAction** element:

```
<xf:rule caption="Receipts" condition="my:expense > 75">
  <xf:dialogBoxMessageAction>Don't forget your receipts!</xf:dialogBoxMessageAction>
  <xf:switchViewAction view="Approval View"/>
  <xf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression="sum(my:expenses/my:expense) * my:taxRate"/>
  <xf:queryAction adapter="Exchange Rates"/>
  <xf:submitAction adapter="Exchange Rates"/>
  <xf:openNewDocumentAction solutionURI="uri:microsoft-Expenses">
  <xf:closeDocumentAction promptToSaveChanges="yes"/>
  <xf:dialogBoxExpressionAction>my:group/my:field1</xf:dialogBoxExpressionAction>
  <xf:exitRuleSet/>
</xf:rule>
```


operation Element

Defines the operation (method) of the Web service to be used for retrieving and submitting data.

Type

xsd:complexType

Structure

| Name | Description |
|-----------------------------------|--|
| <u>name</u> | (Required attribute) Contains the unique name of the Web service method. |
| <u>serviceUrl</u> | (Required attribute) Contains the Web service <u>Uniform Resource Locator (URL)</u> to which the request should be sent. |
| <u>soapAction</u> | (Required attribute) Contains the value of the SOAPAction attribute in the Simple Object Access Protocol (SOAP) request message. |
| <u>input</u> | (Optional element) Contains the substitution information for parts of the input SOAP message to the Web service. |

Remarks

The **operation** element is a required element of the [webServiceAdapter](#) element.

Example

The following is an example of the **operation** element:

```
<xsf:query>
  <xsf:webServiceAdapter
    wsdlUrl="http://localhost/infopathwebservicessample/infopathwebs
    queryAllowed="yes"
    submitAllowed="no">
    <xsf:operation
      name="getOrders"
      soapAction="http://tempuri.org/getOrders"
      serviceUrl="http://localhost/infopathwebservicessample/infopathv
    <xsf:input
      source="Submit.xml">
    </xsf:input>
    </xsf:operation>
  </xsf:webServiceAdapter>
</xsf:query>
```


override Element

Defines one overriding error message for [XML Schema](#) data type errors for an individual XML [Document Object Model \(DOM\)](#) node.

Type

xsd:complexType

Structure

| Name | Description |
|------------------------------|--|
| errorMessage | (Required element) Specifies the error message that should be returned if the value of the specified XML DOM node is considered to be invalid. |
| match | (Required attribute) Identifies the XML DOM node for which the error message override is defined. |

Remarks

The **override** element is an optional element of the [schemaErrorMessage](#) element.

Example

The following is an example of the **override** element:

```
<xsf:schemaErrorMessages>
  <xsf:override
    match="/sampleData/number">
    <xsf:errorMessage
      shortMessage="Invalid Number.">
      The value entered must be a valid number.
    </xsf:errorMessage>
  </xsf:override>
</xsf:schemaErrorMessages>
```

package Element

Contains information about all of the files used in a Microsoft Office InfoPath 2003 form.

Type

xsd:complexType

Structure

| Name | Description |
|-----------------------|--|
| files | (Required element) Identifies a list of files that are used by an InfoPath form. |

Remarks

The **package** element is a required element of the [xDocumentClass](#) element.

Example

The following is an example of the **package** element:

```
<xsf:package>
  <xsf:files>
    <xsf:file name="view_1.xsl">
      <xsf:fileProperties>
        <xsf:property
          name="lang"
          type="string"
          value="1033"/>
      </xsf:fileProperties>
    </xsf:file>
  </xsf:files>
</xsf:package>
```


partFragment Element

Defines one substitution group for a specific part of the input Simple Object Access Protocol (SOAP) message.

Type

xsd:complexType

Structure

| Name | Description |
|-------------------------------------|--|
| <u>match</u> | (Required attribute) Contains an <u>XPath</u> expression that identifies the elements and attributes inside the input SOAP message that are to be replaced at run time. |
| <u>replaceWith</u> | (Required attribute) Contains an XPath expression that identifies the values in the source document that should be used to replace parts of the input SOAP message. |
| <u>sendAsString</u> | (Optional attribute) Specifies that the data is submitted as a string through the <u>webServiceAdapter</u> data adapter. |
| <u>dataObject</u> | (Optional attribute) Specifies the name of the data object to use when submitting a <u>partFragment</u> element to a Microsoft Biztalk 2004 Human Workflow Services (HWS) server. |
| <u>filter</u> | (Optional attribute) Specifies the <u>XPath</u> expression of the structured XML subtree when submitting a subset of the <u>XML data</u> . |

Remarks

The **partFragment** element is an optional element of the [input](#) element.
Multiple **partFragment** elements are allowed.

Example

The following is an example of the **partFragment** element:

```
<xf:query>
  <xf:webServiceAdapter
    wsdlUrl="http://localhost/infopathwebservicessample/infopathwebs
    queryAllowed="yes"
    submitAllowed="no">
    <xf:operation
      name="getOrders"
      soapAction="http://tempuri.org/getOrders"
      serviceUrl="http://localhost/infopathwebservicessample/infopathv
    <xf:input
      source="Submit.xml">
      <xf:partFragment
        match="/dfs:myFields/dfs:dataFields/s0:IsPrime/s0:inValue"
        replaceWith="/dfs:myFields/dfs:dataFields/s0:IsPrime" />
      </xf:input>
    </xf:operation>
  </xf:webServiceAdapter>
</xf:query>
```

permissions Element

Asserts the permissions the InfoPath form requires for the ActiveX controls in the view.

Type

xsd:complexType

Child Elements

| Element | Description |
|--------------------------------|---|
| allowedControl | Specifies the ActiveX controls that are allowed to be instantiated. |

Attributes

None.

Definition

```
<xsd:element name="permissions">  
  <xsd:complexType>  
    <xsd:choice minOccurs="0" maxOccurs="unbounded">  
      <xsd:element ref="xsf:allowedControl"/>  
    </xsd:choice>  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **permissions** element also provides installation information for ActiveX controls that are not already installed on the user's computer.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **permissions** element:

```
<xsf:permissions>  
  <xsf:allowedControl  
    cabFile="{84F32C01-78D8-4B93-8ED4-106DA70224C2}.cab"  
    clsid="{84F32C01-78D8-4B93-8ED4-106DA70224C2}"  
    version="1,0,0,1"/>  
  <xsf:allowedControl  
    clsid="{F08DF954-8592-11D1-B16A-00C0F0283630}"/>  
</xsf:permissions>
```


print Element

Specifies whether the user can use the [form's](#) menus, toolbars, or keyboard shortcuts to print the form.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|--------------------|------------------------|----------|---|--|
| ui | xsfd:xdEnabledDisabled | Yes | Sets whether the user can print the form. | <ul style="list-style-type: none">• enabled• disabled |

Definition

```
<xsd:element name="print">  
  <xsd:complexType>  
    <xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="requi  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **print** element is an optional element of the [featureRestrictions](#) element. If this element is not included in the [form definition file \(.xsf\)](#), the user can use the form's menus, toolbars, or keyboard shortcuts to print the form.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **print** element:

```
<xsf:featureRestrictions>  
  <xsf:save ui="disabled"/>  
  <xsf:sendMail ui="disabled"/>  
  <xsf:exportToWeb ui="disabled"/>  
  <xsf:exportToExcel ui="disabled"/>  
  <xsf:print ui="enabled"/>  
  <xsf:autoRecovery feature="disabled"/>  
</xsf:featureRestrictions>
```

printSettings Element

Specifies the printer settings used when printing the view.

Type

xsd:complexType

Structure

| Name | Description |
|--|---|
| <u>bottomMargin</u> | (Optional attribute) Specifies the bottom margin. |
| <u>collate</u> | (Optional attribute) Specifies whether the paper is collated. |
| <u>copies</u> | (Optional attribute) Specifies the number of copies. |
| <u>footer</u> | (Optional attribute) Specifies the footer text. |
| <u>header</u> | (Optional attribute) Specifies the header text. |
| <u>leftMargin</u> | (Optional attribute) Specifies the left margin. |
| <u>marginUnitsType</u> | (Optional attribute) Specifies the margin unit size. |
| <u>orientation</u> | (Optional attribute) Specifies the orientation. |
| <u>pageRangeEnd</u> | (Optional attribute) Specifies the last page to be printed. |
| <u>pageRangeStart</u> | (Optional attribute) Specifies the first page to be printed. |
| <u>paperSize</u> | (Optional attribute) Specifies the size of the paper. |
| <u>paperSource</u> | (Optional attribute) Specifies the source of the paper. |
| <u>printerName</u> | (Optional attribute) Specifies the name of the printer. |
| <u>printerSpecificSettings</u> | (Optional attribute) Specifies printer specific settings. |
| <u>rightMargin</u> | (Optional attribute) Specifies the right margin. |
| <u>topMargin</u> | (Optional attribute) Specifies the top margin. |

Remarks

The **printSettings** element is an optional element of the [view](#) element.

Example

The following is an example of the **printSettings** element:

```
<xsf:printSettings
  orientation="landscape"
  header="&Pqsdf"
  footer="&D"
  printerName="\\printserver\printer"
  paperSource="Auto Select"
  paperSize="Envelope DL"
  topMargin="0.8"
  leftMargin="0.8"
  bottomMargin="0.8"
  rightMargin="0.8"
  marginUnitsType="in"
  copies="2"
  collate="no"
  pageRangeStart="1"
  pageRangeEnd="1" >
  <xsf:header>
    <font>
      <div>&Pqsdf</div>
    </font>
  </xsf:header>
  <xsf:footer>
    <font>
      <div>&D</div>
    </font>
  </xsf:footer>
</xsf:printSettings>
```


property Element

Defines one specific property for the specified file.

Type

xsd:complexType

Structure

| Name | Description |
|-----------------------|---|
| name | (Required attribute) Defines the name of the property. |
| type | (Required attribute) Defines the type of the property. |
| value | (Required attribute) For simple properties, specifies a value for the property. For complex and multi-valued properties, the specified value is defined as a container XML tree using an open content model . |

Remarks

The **property** element is an optional element of the [fileProperties](#) element.

Example

The following is an example of the **property** element:

```
<xsf:package>
  <xsf:files>
    <xsf:file name="view_1.xsl">
      <xsf:fileProperties>
        <xsf:property
          name="lang"
          type="string"
          value="1033"/>
      </xsf:fileProperties>
    </xsf:file>
  </xsf:files>
</xsf:package>
```


query Element

Associates a [data adapter](#) with a data object or a form's underlying XML document.

Type

xsd:complexType

Structure

| Name | Description |
|---------------------------------------|--|
| adoAdapter | (Optional element) Defines an ActiveX Data Objects (ADO) data adapter that retrieves data from an ADO data source for the specified data object. |
| webServiceAdapter | (Optional element) Defines a Web service data adapter that retrieves data from a Web service for the specified data object. |
| xmlFileAdapter | (Optional element) Defines an .xml file data adapter that retrieves data from an .xml file for the specified data object. |
| queryAction | (Optional element) Defines a data connection query action. |
| sharepointListAdapter | (Optional element) Contains the data adapter information to query a SharePoint list or library. |

Remarks

The **query** element is an optional element of the [xDocumentClass](#) element and the [dataObject](#) element.

Only one data adapter can be specified in the **query** element.

Example

The following is an example of the **query** element:

```
<xsf:query>
  <xsf:adoAdapter
    connectionString="Provider=Microsoft.Jet.OLEDB.4.0;
      Password=&quot;&quot;;User ID=Admin;
      Data Source=infwind.mdb;Mode=Share Deny None;
      Extended Properties=&quot;&quot;;..."
    commandText="select [EmployeeID],[LastName],[FirstName]
      from [Employees] as [Employees]"
    queryAllowed="yes"
    submitAllowed="yes">
  </xsf:adoAdapter>
</xsf:query>
```


query Element (dataObject Element)

Associates a [data adapter](#) with a data object.

Type

xsd:complexType

Parent Elements

| Element | Description |
|----------------------------|---|
| dataObject | Defines a secondary data object that is used in a Microsoft Office InfoPath form. |

Child Elements

| Element | Description |
|--|---|
| <u>adoAdapter</u> | (Optional element) Defines an <u>ActiveX Data Objects (ADO)</u> data adapter that retrieves data from an ADO data source for the specified data object. |
| <u>webServiceAdapter</u> | (Optional element) Defines a Web service data adapter that retrieves data from a Web service for the specified data object. |
| <u>xmlFileAdapter</u> | (Optional element) Defines an .xml file data adapter that retrieves data from an .xml file for the specified data object. |
| <u>sharepointListAdapter</u> | (Optional element) Contains the data adapter information to query a SharePoint list or library. |

Attributes

None.

Definition

```
<xsd:element name="query" >
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="xsf:adoAdapter" />
      <xsd:element ref="xsf:webServiceAdapter" />
      <xsd:element ref="xsf:xmlFileAdapter" />
      <xsd:element ref="xsf:sharepointListAdapter" />
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **query** element:

```
<xf:dataObjects>
  <xf:dataObject
    name="EmployeeNames"
    schema="EmployeeNames.xsd"
    initOnLoad="yes">
    <xf:query>
      <xf:adoAdapter
        connectionString="Provider=Microsoft.Jet.OLEDB.4.0;
        Password='';User ID=Admin;
        Data Source=infwind.mdb;Mode=Share Deny None;
        Extended Properties='';..."
        commandText="select [EmployeeID],[LastName],[FirstName]
        from [Employees] as [Employees]"
        queryAllowed="yes"
        submitAllowed="yes">
      </xf:adoAdapter>
    </xf:query>
  </xf:dataObject>
</xf:dataObjects>
```

queryAction Element

Defines a data connection [query](#) action.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|-------------------------|------------|----------|--|-----------------|
| adapter | xsd:string | Yes | Contains the name of the data string adapter to query. | |

Definition

```
<xsd:element name="queryAction">  
  <xsd:complexType>  
    <xsd:attribute name="adapter" type="xsd:string" use="required"></  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **queryAction** element is a child element of the [rule](#) element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **queryAction** element:

```
<xf:rule caption="Receipts" condition="my:expense > 75">
  <xf:dialogBoxMessageAction>Don't forget your receipts!</xf:dialogBoxMessageAction>
  <xf:switchViewAction view="Approval View"/>
  <xf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression="sum(my:expenses/my:expense) * my:taxRate"/>
  <xf:queryAction adapter="Exchange Rates"/>
  <xf:submitAction adapter="Exchange Rates"/>
  <xf:openNewDocumentAction solutionURI="uri:microsoft-ExpenseForm"/>
  <xf:closeDocumentAction promptToSaveChanges="yes"/>
  <xf:dialogBoxExpressionAction>my:group/my:field1</xf:dialogBoxExpressionAction>
  <xf:exitRuleSet/>
</xf:rule>
```


role Element

Defines role.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|----------------------|----------------------------|----------|----------------------------|--|
| name | xdRoleName | Yes | Used to identify the role. | Same restriction as xdViewName . |

Definition

```
<xsd:element name="role">  
  <xsd:complexType>  
    <xsd:attribute name="name" type="xsf:xdRoleName" use="required"/>  
  </xsd:complexType>  
</xsd:element>
```

Remarks

Each time a role is created for a form, a corresponding **role** element is created in the [form definition file \(.xsf\)](#). You can assign users to roles by using user names, groups, and **XPath** user names. Users can be selected for roles in one of four ways:

Initiator— role for new documents.

User name— NT domain users (intranet or trusted only).

XPath user name— user names stored in XML data (intranet or trusted only).

Group— Active Directory directory service groups (intranet or trusted only).

InfoPath makes the user assignment by following the membership order in the form definition file.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **role** element:

```
<roles initiator="xsd:string" default="xsd:string" hideStatusBarDispla;
  <role name="xsf:xdRoleName"/>
  <membership>
    <getUserNameFromData dataObject="xsd:string" select="xsd:
    <userName name="xsd:string" memberOf="xsd:string" />
    <userName name="xsd:string" memberOf="xsd:string" />
    <group name="xsd:string" memberOf="xsd:string" />
  </membership>
</roles>
```

roles Element

Defines roles.

Type

xsd:complexType

Child Elements

| Element | Description |
|----------------------------|--|
| role | Defines role. |
| membership | Maps a user or group of users to a role. |

Attributes

| Attribute | Type | Required | Description | Possible Values |
|---|------------|----------|--|--|
| <u>default</u> | xsd:string | Yes | Specifies the name identifier of the role that is the default role. | string |
| <u>initiator</u> | xsd:string | No | Specifies the name identifier of the role chosen to be the initiator role. | string |
| <u>hideStatusBarDisplay</u> | xsd:YesNo | No | Specifies whether the current role should be displayed in the status bar. | <ul style="list-style-type: none">• yes• no |

Definition

```
<xsd:element name="roles">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:role" minOccurs="1" maxOccurs="unbound" />
      <xsd:element ref="xsf:membership" minOccurs="0" maxOccurs="unbound" />
    </xsd:sequence>
    <xsd:attribute name="default" type="xsd:string" use="required"></xsd:attribute>
    <xsd:attribute name="initiator" type="xsd:string" use="optional"></xsd:attribute>
    <xsd:attribute name="hideStatusBarDisplay" type="xsf:xdYesNo" use="optional"></xsd:attribute>
  </xsd:complexType>
  <!-- role names must be unique -->
  <xsd:unique name="roles_name_unique">
    <xsd:selector xpath="/xsf:role" />
    <xsd:field xpath="@name" />
  </xsd:unique>
  <!-- fields must reference existing role -->
  <xsd:key name="role_name_key">
    <xsd:selector xpath="/xsf:role" />
    <xsd:field xpath="@name" />
  </xsd:key>
  <xsd:keyref name="role_default" refer="xsf:role_name_key">
    <xsd:selector xpath="." />
    <xsd:field xpath="@default" />
  </xsd:keyref>
  <xsd:keyref name="role_initiator" refer="xsf:role_name_key">
    <xsd:selector xpath="." />
    <xsd:field xpath="@initiator" />
  </xsd:keyref>
  <xsd:keyref name="role_membership" refer="xsf:role_name_key">
```

```
<xsd:selector xpath="/xsf:membership/*" />  
<xsd:field xpath="@memberOf" />  
</xsd:keyref>  
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **roles** element:

```
<xsf:roles initiator="A" default="C" hideStatusBarDisplay="yes">
  <xsf:role name="A"/>
  <xsf:role name="B"/>
  <xsf:role name="C"/>

  <xsf:membership>
    <xsf:getUserNameFromData dataObject="catalog" select="/dfs:my
    <xsf:userName name="Domain\username1" memberOf="A" />
    <xsf:userName name="Domain\username2" memberOf="B" />
    <xsf:group name="Domain\username3" memberOf="C" />
  </xsf:membership>
</xsf:roles>
```


rule Element

Defines an action invoked after an event has occurred in the [form](#).

Type

xsd:complexType

Child Elements

| Element | Description |
|--|--|
| <u>assignmentAction</u> | Defines an action to set the value of a <u>field</u> . |
| <u>closeDocumentAction</u> | Defines a form close action. |
| <u>dialogBoxExpressionAction</u> | Defines an <u>XPath</u> expression to be displayed in a dialog box. |
| <u>dialogBoxMessageAction</u> | Defines a literal message to be displayed in a dialog box. |
| <u>exitRuleSet</u> | An element which, if present at the end of the rule, stops further rule processing of the entire <u>ruleSet</u> when the rule is executed. |
| <u>openNewDocumentAction</u> | Defines a form create action. |
| <u>queryAction</u> | Defines a data connection <u>query</u> action. |
| <u>submitAction</u> | Defines a form <u>submit</u> action. |
| <u>switchViewAction</u> | Defines a <u>view</u> switch action. |

Attributes

| Attribute | Type | Required | Description | Possible Values |
|----------------------------------|--------------------------------|----------|--|--|
| <u>caption</u> | xsd:string | Yes | Contains the name of the rule as it appears in the user interface. | string |
| <u>condition</u> | xsd:string | No | Defines the XPath expression, evaluated as a Boolean value, that determines whether the associated action will be invoked. | string |
| <u>isEnabled</u> | <u>xdYesNo</u> | No | Specifies whether the rule is enabled for the form. The default value is "yes". | <ul style="list-style-type: none">• yes• no |

Definition

```
<xsd:element name="rule">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="xsf:dialogBoxMessageAction"/>
        <xsd:element ref="xsf:dialogBoxExpressionAction"/>
        <xsd:element ref="xsf:switchViewAction"/>
        <xsd:element ref="xsf:assignmentAction"/>
        <xsd:element ref="xsf:queryAction"/>
        <xsd:element name="submitAction">
          <xsd:complexType>
            <xsd:attribute name="adapter" type="xsf:xdTitle" use="require
            </xsd:complexType>
          </xsd:element>
        <xsd:element ref="xsf:openNewDocumentAction"/>
        <xsd:element ref="xsf:closeDocumentAction"/>
      </xsd:choice>
      <xsd:element name="exitRuleSet" minOccurs="0">
        <xsd:complexType />
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="caption" type="xsd:string" use="required"></
    <xsd:attribute name="condition" type="xsd:string" use="optional">
    <xsd:attribute name="isEnabled" type="xsf:xdYesNo" use="optiona
  </xsd:complexType>
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **rule** element:

```
<xsf:ruleSets>
  <xsf:ruleSet name="RuleSet1">
    <xsf:rule caption="Receipts" condition="my:expense > 75">
      <xsf:dialogBoxMessageAction>Don't forget receipts!</xsf:dialogBo
      <xsf:openNewDocumentAction solution="urn:approvalForm"/>
      <xsf:exitRuleSet/>
    </xsf:rule>
    <xsf:rule caption="Always Submit" isEnabled="no">
      <xsf:submitAction adapter="Expense Database"/>
    </xsf:rule>
  </xsf:ruleSet>

  <xsf:ruleSet name="RuleSet2">
    <xsf:rule caption="Look up contact">
      <xsf:queryAction adapter="Contacts"/>
    </xsf:rule>
  </xsf:ruleSet>
</xsf:ruleSets>
```


ruleSet Element

Contains one or more [rule](#) elements.

Type

xsd:complexType

Child Elements

| Element | Description |
|----------------------|--|
| rule | Defines an action invoked after an event has occurred in the form. |

Attributes

| Attribute | Type | Required | Description | Possible Values |
|----------------------|------------|----------|--|-----------------|
| name | xsd:string | Yes | Contains the name of the ruleSet element. | string |

Definition

```
<xsd:element name="ruleSet">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:rule" minOccurs="1" maxOccurs="unbound" />
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **ruleSet** element:

```
<xsf:ruleSets>
  <xsf:ruleSet name="RuleSet1">
    <xsf:rule caption="Receipts" condition="my:expense > 75">
      <xsf:dialogBoxMessageAction>Don't forget receipts!</xsf:dialogBoxMessageAction>
      <xsf:openNewDocumentAction solution="urn:approvalForm"/>
      <xsf:exitRuleSet/>
    </xsf:rule>
    <xsf:rule caption="Always Submit" isEnabled="no">
      <xsf:submitAction adapter="Expense Database"/>
    </xsf:rule>
  </xsf:ruleSet>

  <xsf:ruleSet name="RuleSet2">
    <xsf:rule caption="Look up contact">
      <xsf:queryAction adapter="Contacts"/>
    </xsf:rule>
  </xsf:ruleSet>
</xsf:ruleSets>
```

ruleSetAction Element

Defines the [ruleSet](#) action element to be invoked.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|-------------------------|------------|----------|--|-----------------|
| ruleSet | xsd:string | Yes | Contains the name of the ruleSet to be invoked. | string |

Definition

```
<xsd:element name="ruleSetAction">  
  <xsd:complexType>  
    <xsd:attribute name="ruleSet" type="xsd:string" use="required"/></:  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **ruleSetAction** element is a child of the [submit](#), [domEventHandler](#), [onLoad](#), and [button](#) elements.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **ruleSetAction** element:

```
<xsf:onLoad>  
  <xsf:ruleSetAction ruleSet="RuleSet4"/>  
</xsf:onLoad>
```

ruleSets Element

Contains one or more [ruleSet](#) elements.

Type

xsd:complexType

Child Elements

| Element | Description |
|-------------------------|---|
| ruleSet | Contains one or more rule elements. |

Attributes

None.

Definition

```
<xsd:element name="ruleSets">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="xsf:ruleSet" minOccurs="0" maxOccurs="unbo
    </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="ruleSets_name_unique">
    <xsd:selector xpath="./xsf:ruleSet"/>
    <xsd:field xpath="@name"/>
  </xsd:unique>
</xsd:element>
```

Remarks

The **ruleSets** element is an optional element of the [xDocumentClass](#) element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **ruleSets** element:

```
<xf:ruleSets>
  <xf:ruleSet name="RuleSet1">
    <xf:rule caption="Receipts" condition="my:expense > 75">
      <xf:dialogBoxMessageAction>Don't forget receipts!</xf:dialogBo
      <xf:openNewDocumentAction solution="urn:approvalForm"/>
      <xf:exitRuleSet/>
    </xf:rule>
    <xf:rule caption="Always Submit" isEnabled="no">
      <xf:submitAction adapter="Expense Database"/>
    </xf:rule>
  </xf:ruleSet>

  <xf:ruleSet name="RuleSet2">
    <xf:rule caption="Look up contact">
      <xf:queryAction adapter="Contacts"/>
    </xf:rule>
  </xf:ruleSet>
</xf:ruleSets>
```


save Element

Shows whether the **Save using custom code** check box option is enabled.

Type

xsd:complexType

Child Elements

| Element | Description |
|----------------------------------|--|
| useScriptHandler | Specifies that the submit, save, or version upgrade operation will be handled using form code. |

Attributes

None.

Definition

```
<xsd:element name="save">  
  <xsd:complexType>  
    <xsd:choice minOccurs="0" maxOccurs="1">  
      <xsd:element ref="xsf:useScriptHandler"/>  
    </xsd:choice>  
  </xsd:complexType>  
</xsd:element>
```

Remarks

If the **Save using custom code** check box option is enabled, the `<xsf:save><xsf:useScriptHandler/></xsf:save>` elements will be added to the [form definition file \(.xsf\)](#) as a child of the **xDocumentClass** element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **save** element:

```
<xsf:save>  
  <xsf:useScriptHandler/>  
</xsf:save>
```


save Element (featureRestrictions Element)

Specifies whether the user can use the [form's](#) menus, toolbars, or keyboard shortcuts to save the form.

Type

xsd:complexType

Parent Elements

| Element | Description |
|-------------------------------------|---|
| featureRestrictions | Specifies whether the user can use the form's menus, toolbars, or keyboard shortcuts to save the form, export the form, print the form, or send the form as an e-mail attachment. |

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|--------------------|-----------------------|----------|--|--|
| ui | xsf:xdEnabledDisabled | Yes | Sets whether the user can save the form. | <ul style="list-style-type: none">• enabled• disabled |

Definition

```
<xsd:element name="save" minOccurs="0">  
  <xsd:complexType>  
    <xsd:attribute name="ui" type="xsd:boolean" use="required" />  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **save** element is an optional element of the [featureRestrictions](#) element. If this element is not included in the [form definition file \(.xsf\)](#), then the user can use the form's menus, toolbars, or keyboard shortcuts to save the form.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **save** element:

```
<xsf:featureRestrictions>  
  <xsf:save ui="disabled"/>  
  <xsf:sendMail ui="disabled"/>  
  <xsf:exportToWeb ui="disabled"/>  
  <xsf:exportToExcel ui="disabled"/>  
  <xsf:print ui="enabled"/>  
  <xsf:autoRecovery feature="disabled"/>  
</xsf:featureRestrictions>
```


schemaErrorMessages Element

Contains custom error messages used to override [XML Schema](#) data type errors.

Type

xsd:complexType

Structure

| Name | Description |
|--------------------------|---|
| override | (Optional element) Defines one overriding error message for XML Schema data type errors for an individual XML Document Object Model (DOM) node. |

Remarks

The **schemaErrorMessages** element is an optional element of the [xDocumentClass](#) element.

Example

The following is an example of the **schemaErrorMessages** element:

```
<xsf:schemaErrorMessages>
  <xsf:override
    match="/sampleData/number">
    <xsf:errorMessage
      shortMessage="Invalid Number.">
      The value entered must be a valid number.
    </xsf:errorMessage>
  </xsf:override>
</xsf:schemaErrorMessages>
```


script Element

Defines the source scripting file containing all the data-level script content referenced in the form.

Type

xsd:complexType

Structure

| Name | Description |
|---------------------|--|
| src | (Required attribute) Provides a relative URL within the form template to the specified script source file. |

Remarks

The **script** element is an optional element of the [scripts](#) element.

Example

The following is an example of the **script** element:

```
<xsf:scripts language="jscript">  
  <xsf:script src="myscripts.js"/>  
</xsf:scripts>
```


scripts Element

Defines the source of all business logic scripts used at the document level in the form.

Type

xsd:complexType

Structure

| Name | Description |
|--------------------------------------|---|
| language | (Required attribute) Defines the script language used in the business logic source files. |
| script | (Optional element) Defines the source scripting file containing all the data-level script content referenced in the form. |
| enforceScriptTimeout | (Optional element) Specifies whether to enable or disable a time-out period for scripts. |

Remarks

The **scripts** element is an optional element of the [xDocumentClass](#) element.

There can potentially be more than one **script** element in the **scripts** element; however, they should all be written in the same scripting language. All defined script sources are concatenated and loaded into a single script engine environment when the form is being filled out. This means that duplicate functions and property names are resolved by default by the script engine, and the last declaration is the one that is used. Form developers must ensure unique names across scripting source files that are used in a form.

In order to add encoded jscript files to a form, you need to edit the [form definition file \(.xsf\)](#) to change the **language** property in the **scripts** element. After the property has been updated, the **scripts** section in the .xsf file will look like the following:

```
<xsf:scripts language="jscript.encode">  
  <xsf:script src="scriptenc.js"></xsf:script>  
</xsf:scripts>
```

The **scripts** element is not present in the .xsf file if the form uses managed code.

Example

The following is an example of the **scripts** element:

```
<xsf:scripts language="jscript" enforceScriptTimeout="no">  
  <xsf:script src="internal.js" />  
  <xsf:script src="script.js" />  
</xsf:scripts>
```


sendMail Element

Specifies whether the user can use the [form's](#) menus, toolbars, or keyboard shortcuts to send the form as an email attachment.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|--------------------|-----------------------------------|----------|---|--|
| ui | xdEnabledDisabled | required | Sets whether the user can send the form as an email attachment. | <ul style="list-style-type: none">• enabled• disabled |

Definition

```
<xsd:element name="sendMail" >  
  <xsd:complexType>  
    <xsd:attribute name="ui" type="xsd:boolean" use="required" />  
  </xsd:complexType>  
</xsd:element>
```

Remarks

The **sendMail** element is an optional element of the [featureRestrictions](#) element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **sendMail** element:

```
<xsf:featureRestrictions>  
  <xsf:save ui="disabled"/>  
  <xsf:sendMail ui="disabled"/>  
  <xsf:exportToWeb ui="disabled"/>  
  <xsf:exportToExcel ui="disabled"/>  
  <xsf:print ui="enabled"/>  
  <xsf:autoRecovery feature="disabled"/>  
</xsf:featureRestrictions>
```


sharepointListAdapter Element

Contains the data adapter information to query a SharePoint list or library.

Type

xsd:complexType

Child Elements

| Element | Description |
|-----------------------|--|
| field | Contains field mapping information for each field in a SharePoint list and the corresponding name used in Microsoft Office InfoPath 2003 Service Pack 1. |

Attributes

| Attribute | Type | Required | Description | Possible |
|--------------------------------|-------------------------|----------|---|--|
| name | xdTitle | Yes | Contains the name of the sharepointListAdapter and is used as the secondary data source name. | minLength maxLength pattern = ([\p{Z}\p{ ([\p{Zl}\p{ ([\p{Z}\p{ |
| siteUrl | xsd:string | Yes | Contains the Uniform Resource Locator (URL) of the SharePoint site. | string |
| sharepointGuid | xsd:string | Yes | Contains the GUID of the SharePoint list. | string |
| infopathGroup | xsd:string | Yes | Contains the name of the group under which all fields in the SharePoint list will be stored. | string |
| queryAllowed | xdYesNo | No | Specifies whether the adapter can be used for querying the data source. Always set to "yes" for the sharepointListAdapter . | <ul style="list-style-type: none"> • yes • no |
| submitAllowed | xdYesNo | No | Specifies whether the adapter can be used for submitting to the data source. Omitted for the sharepointListAdapter , corresponding to a default value of "no". | <ul style="list-style-type: none"> • yes • no |

Definition

```
<xsd:element name="sharepointListAdapter">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="field" minOccurs="0" maxOccurs="unbound"
        <xsd:complexType>
          <xsd:attribute name="sharepointName" type="xsd:string" use="required"
            <xsd:attribute name="infopathName" type="xsd:string" use="required"
              <xsd:attribute name="isLookup" type="xsd:boolean" use="optional"
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string" use="required"
        <xsd:attribute name="siteUrl" type="xsd:string" use="required"
        <xsd:attribute name="sharepointGuid" type="xsd:string" use="required"
        <xsd:attribute name="infopathGroup" type="xsd:string" use="required"
        <xsd:attribute name="queryAllowed" type="xsd:boolean" use="optional"
        <xsd:attribute name="submitAllowed" type="xsd:boolean" use="optional"
      </xsd:complexType>
    </xsd:element>
```

Remarks

Each field in the SharePoint list that is returned by the data adapter will have a **field** element. The **sharepointListAdapter** can only be used as a [secondary data source](#) and does not support a submit action.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **sharepointListAdapter** element:

```
<xf:sharepointListAdapter
  name="Status Report library"
  siteUrl="http://xyzco/reports/"
  sharepointGuid="{ABD2E239-0EE7-48F4-B506-C38A1728E195}"
  infopathGroup="XyzReportsLibrary"
  queryAllowed="yes">
  <xf:field
    sharepointName="File_x0020_Type"
    infopathName="Type"></xf:field>
  <xf:field
    sharepointName="xd__x007b_D00F1DBD_..."
    infopathName="Title_1"></xf:field>
</xf:sharepointListAdapter>
```


signedDataBlock Element

Defines a node set in the [form's](#) underlying XML document to which a [digital signature](#) can be applied.

Type

xsd:complexType

Child Elements

| Element | Description |
|-------------------------|---|
| message | The signature confirmation message shown when a digital signature is applied to the form or to a section of the form. |

Attributes

| Attribute | Type | Required | Description |
|--|--|----------|--|
| <u>name</u> | <u>xdSignedDataBlockName</u> | Yes | Contains the name of the signedData element. |
| <u>data</u> | xsd:string | Yes | Contains an <u>XPath</u> match expression that defines the resource set to which signature will be applied. |
| <u>signatureLocation</u> | xsd:string | Yes | Contains an XPath expression that points to the XML Document Object Model (DOM) node within the form underlying the X document that is used for storing the digital signature. |
| <u>mode</u> | <u>xdSignatureRelationEnum</u> | Yes | Specifies the signature relationship for the signedData element. The default is "signature" |

Definition

```
<xsd:element name="signedDataBlock">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="message" type="xsf:xdSignedDataBlockMes
    </xsd:sequence>
    <xsd:attribute name="name" type="xsf:xdSignedDataBlockName" u
    <xsd:attribute name="data" type="xsd:string" use="required"></xsd
    <xsd:attribute name="signatureLocation" type="xsd:string" use="re
    <xsd:attribute name="mode" type="xsf:xdSignatureRelationEnum"
  </xsd:complexType>
  <xsd:unique name="signedDataBlock_name_unique">
    <xsd:selector xpath="."/>
    <xsd:field xpath="@name"/>
  </xsd:unique>
</xsd:element>
```

Remarks

The node set must be the union of connected subtrees. If a nonleaf node is included in a **signedDataBlock**, then all children must be included.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **signedDataBlock** element:

```
<xsf:documentSignatures>  
  <xsf:signedDataBlock name="main"  
    data="my:myfields/my:subtree1 | my:myfields/my:subtree2"  
    signatureLocation="my:mifields/sig:signatures/main"  
    mode="countersign">  
    <xsf:message>By pressing the &quot;Sign&quot; button below, I agr  
      to the terms of this document.</xsf:message>  
  </xsf:signedDataBlock>  
</xsf:documentSignatures>
```


solutionProperties Element

Contains design-time information about a Microsoft Office InfoPath 2003 form.

Type

xsd:complexType

Structure

| Name | Description |
|--|---|
| <u>allowCustomization</u> | (Optional attribute) Indicates whether the form can be modified or customized. |
| <u>automaticallyCreateNodes</u> | (Optional attribute) Indicates whether XML <u>Document Object Model (DOM)</u> nodes will be automatically generated when controls are inserted in the view in <u>design mode</u> . |
| <u>fullyEditableNamespace</u> | (Optional attribute) Identifies the <u>namespace</u> of an <u>XML Schema</u> in the <u>form template</u> that can be entirely modified in InfoPath design mode. |
| <u>lastOpenView</u> | (Optional attribute) Identifies the name of the <u>view</u> that was last open in InfoPath when designing a form. |
| <u>lastVersionNeedingTransform</u> | (Optional attribute) Identifies, temporarily, the value of the <u>maxToVersionUpgrade</u> attribute in the <u>documentVersionUpgrade</u> element for upgrade with an <u>.xslt</u> file if scripting code is being used for the upgrade. |
| <u>scriptLanguage</u> | (Optional attribute) Identifies the name of the scripting language used to implement the business logic of the form. |
| <u>publishSaveUrl</u> | (Optional attribute) Contains the location of the saved <u>form template</u> if different from the value of the <u>publishUrl</u> attribute. |

Remarks

The **solutionProperties** element is an optional element of the [applicationParameters](#) element.

Example

The following is an example of the **solutionProperties** element:

```
<xsf: applicationParameters application="InfoPath Design Mode">  
  <xsf: solutionProperties  
    allowCustomization="no"  
    lastOpenView="view1"  
    scriptLanguage="JScript"  
    automaticallyCreateNodes="no"  
    lastVersionNeedingTransform="1.1.0.10"  
    fullyEditableNamespace="urn:names?pace1:mynames"/>  
</xsf:applicationParameters>
```


subject Element

Contains the subject line of the e-mail message when the form is submitted using the [emailAdapter](#) element.

Type

xsd:complexType

Parent Elements

| Element | Description |
|------------------------------|---|
| emailAdapter | Parent element that contains the information needed to submit an InfoPath form as an attachment to an e-mail message, with a specified set of recipients, a subject, and an introduction. |

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|---------------------------|-------------------------------------|----------|--|-----------------|
| value | xsd:string | Yes | Contains the value of the subject element. | string |
| valueType | xdExpressionLiteral | No | Specifies whether the value attribute is interpreted as an XPath expression or a literal string. <ul style="list-style-type: none">• expression• literal | |

Definition

```
<xsd:element name="subject" minOccurs="0">  
  <xsd:complexType>  
    <xsd:attribute name="value" type="xsd:string" use="required"/></xsd:  
    <xsd:attribute name="valueType" type="xsd:string" use="optional" />  
  </xsd:complexType>  
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **subject** element:

```
<xsf:emailAdapter name="Submit" submitAllowed="yes">  
  <xsf:to value="someone@example.com" valueType="literal"/>  
  <xsf:cc value="my:ccNames" valueType="expression"/>  
  <xsf:bcc value="someoneelse@example.com" valueType="literal"/>  
  <xsf:subject value="My report" valueType="literal"/>  
  <xsf:intro value="See below"/>  
  <xsf:attachmentFileName value="Status Report" valueType="literal"/>  
</xsf:emailAdapter>
```


submit Element

Contains information about the submission functionality of a form.

Type

xsd:complexType

Structure

| Name | Description |
|--|--|
| <u>caption</u> | (Optional attribute) Defines the name of the submit button and corresponding menu item that will appear on the File menu when a user is filling out the form. |
| <u>disableMenuItem</u> | (Optional attribute) Specifies whether the menu item for the submit operation should be disabled. |
| <u>errorMessage</u> | (Optional element) Specifies the text to be used in the error message. If the showStatusDialog attribute of the submit element is set to "no", this element will be ignored. |
| <u>onAfterSubmit</u> | (Optional attribute) Specifies whether the form should be closed, kept open, or if a new form should be created after a successful submission. |
| <u>showSignatureReminder</u> | (Optional attribute) Specifies whether a dialog box should be displayed to prompt the user to digitally sign the form before submitting it. |
| <u>showStatusDialog</u> | (Optional attribute) Specifies whether the status dialog box should be shown after the submit operation. Values include "yes" and "no". The default values is "yes". |
| <u>successMessage</u> | (Optional element) Specifies the text to be used to notify the user that the submission was successful. |
| <u>useHttpHandler</u> | (Optional element) Specifies that the form is to be submitted to the specified Uniform Resource Locator (URL) using the specified HTTP method. |
| <u>useScriptHandler</u> | (Optional element) Specifies that the form is |

to be submitted by scripting code in the associated script file. Submit code must be written in the OnSubmitRequest [event handler](#) in the form's primary scripting file.

[**useQueryAdapter**](#)

(Optional element) Specifies that the form is to be submitted to the same [data adapter](#) as specified in the [query](#) element.

[**webServiceAdapter**](#)

(Optional element) Specifies that the form is to be submitted to a Web service adapter.

[**submitAction**](#)

(Optional element) Specifies the data adapter used to submit the [form](#).

Remarks

The **submit** element is an optional element of the [xDocumentClass](#) element.

Example

The following is an example of the **submit** element:

```
<xf:submit
  caption="Submit"
  disableMenuItem="no"
  onAfterSubmit="KeepOpen"
  showStatusDialog="yes"
  showSignatureReminder="yes">
  <xf:useScriptHandler/>
  <xf:successMessage>Submit was successful.</xf:successMessage>
  <xf:errorMessage>Submit was not successful.</xf:errorMessage>
</xf:submit>
```


submitAction Element (rule Element)

Defines a [form submit](#) action.

Type

xsd:complexType

Parent Elements

| Element | Description |
|----------------------|--|
| rule | Defines an action invoked after an event has occurred in the form. |

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|-------------------------|-------------|----------|--|---|
| adapter | xsf:xdTitle | Yes | Contains the name of the data adapter to submit. | minLength = 1 maxLength = 255 pattern = ([\p{Z}\p{Cc}\p{Cf}\p{Cn}]) (([\p{Zl}\p{Zp}\p{Cc}])*) ([\p{Z}\p{Cc}\p{Cf}\p{Cn}])? |

Definition

```
<xsd:element name="submitAction">  
  <xsd:complexType>  
    <xsd:attribute name="adapter" type="xsf:xdTitle" use="required"><  
  </xsd:complexType>  
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **submitAction** element:

```
<xf:rule caption="Receipts" condition="my:expense > 75">
  <xf:dialogBoxMessageAction>Don't forget your receipts!</xf:dialogBoxMessageAction>
  <xf:switchViewAction view="Approval View"/>
  <xf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression="sum(my:expenses/my:expense) * my:taxRate"/>
  <xf:queryAction adapter="Exchange Rates"/>
  <xf:submitAction adapter="Exchange Rates"/>
  <xf:openNewDocumentAction solutionURI="uri:microsoft-ExpenseForm"/>
  <xf:closeDocumentAction promptToSaveChanges="yes"/>
  <xf:dialogBoxExpressionAction>my:group/my:field1</xf:dialogBoxExpressionAction>
  <xf:exitRuleSet/>
</xf:rule>
```


submitAction Element (submit Element)

Specifies the data adapter used to submit the [form](#).

Type

xsd:complexType

Parent Elements

| Element | Description |
|------------------------|--|
| submit | Contains information about the submission functionality of a form. |

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|-----------|-------------|----------|--|---|
| adapter | xsf:xdTitle | Yes | The name of the data adapter used for submitting the form. | minLength = 1 maxLength = 255 pattern = ([\p{Z}\p{Cc}\p{Cf}\p{Cn}]) (([\p{Zl}\p{Zp}\p{Cc}])*) ([\p{Z}\p{Cc}\p{Cf}\p{Cn}])? |

Definition

```
<xsd:element name="submitAction" minOccurs="0" >
  <xsd:complexType>
    <xsd:attribute name="adapter" type="xsf:xdTitle" use="required"><
  </xsd:complexType>
  <xsd:keyref name="submitAdapter_name_keyref" refer="xsf:adapter
    <xsd:selector xpath="." />
    <xsd:field xpath="@adapter" />
  </xsd:keyref>
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **submitAction** element:

```
<xsf:submit caption="Submit" disableMenuItem="no" onAfterSubmit="onAfterSubmit"
  <xsf:submitAction adapter="dav"/>
</xsf:submit>
```

```
<xsf:dataAdapters>
  <xsf:davAdapter name="dav" folderUrl="http://some_server/some_dir" />
</xsf:dataAdapters>
```

successMessage Element

Specifies the text to be used to notify the user that the submission was successful.

Type

xsd:string

Remarks

The **successMessage** element is an optional element of the [submit](#) element.

The **successMessage** element does not contain any attributes or child elements.

The **successMessage** element is ignored if the **showStatusDialog** attribute of the **submit** element is set to "no".

Example

The following is an example of the **successMessage** element:

```
<xf:submit
  caption="Submit"
  disableMenuItem="no"
  onAfterSubmit="KeepOpen"
  showStatusDialog="yes"
  showSignatureReminder="yes">
  <xf:useScriptHandler/>
  <xf:successMessage>Submit was successful.</xf:successMessage>
  <xf:errorMessage>Submit was not successful.</xf:errorMessage>
</xf:submit>
```


switchViewAction Element

Defines a [view](#) switch action.

Type

xsd:complexType

Parent Elements

| Element | Description |
|----------------------|--|
| rule | Defines an action invoked after an event has occurred in the form. |

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|----------------------|----------------|----------|--------------------------------|--|
| view | xsf:xdViewName | Yes | Contains the name of the view. | minLength = 1 maxLength = 255 pattern = ([\p{Z}\p{C}\/\#&"><]) (((\p{Z}\p{C}\/\#&"><])*(\p{Z}\p{C}\/\#&"><]))? |

Definition

```
<xsd:element name="switchViewAction">
  <xsd:complexType>
    <xsd:attribute name="view" type="xsf:xdViewName" use="required"/>
  </xsd:complexType>
  <xsd:keyref name="switchViewAction_view_keyref" refer="xsf:view">
    <xsd:selector xpath="." />
    <xsd:field xpath="@view"/>
  </xsd:keyref>
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **switchViewAction** element:

```
<xf:rule caption="Receipts" condition="my:expense > 75">
  <xf:dialogBoxMessageAction>Don't forget your receipts!</xf:dialogBoxMessageAction>
  <xf:switchViewAction view="Approval View"/>
  <xf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression="sum(my:expenses/my:expense) * my:taxRate"/>
  <xf:queryAction adapter="Exchange Rates"/>
  <xf:submitAction adapter="Exchange Rates"/>
  <xf:openNewDocumentAction solutionURI="uri:microsoft-ExpenseForm"/>
  <xf:closeDocumentAction promptToSaveChanges="yes"/>
  <xf:dialogBoxExpressionAction>my:group/my:field1</xf:dialogBoxExpressionAction>
  <xf:exitRuleSet/>
</xf:rule>
```


task Element

Contains the Microsoft BizTalk Server 2004 Human Workflow Services (HWS) task information enabled for the [form](#).

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|-----------------------------------|------------------|----------|--|---|
| <u>name</u> | xsf:xdHWSname | Yes | The unique name of the task, as specified in the HWS Workflow designer; used for the onClick event of the button in the HWS Workflow task pane. | Cannot contain the following characters: \ / " [] : < > + = ; , ? * @ |
| <u>taskTypeID</u> | xsd:string | Yes | The unique ID for the task. | string |
| <u>caption</u> | xsf:xdHWSCaption | No | The label for the corresponding button in the HWS Workflow task pane to start or respond to a task. | minLength = 1 maxLength = 255 |

Definition

```
<xsd:element name="task">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsf:xdHWSname"
    <xsd:attribute name="taskTypeID" type="xsd:string" u
    <xsd:attribute name="caption" type="xsf:xdHWSCapti
  </xsd:complexType>
</xsd:element>
```

Remarks

The **task** element is an optional element of the [allowedTasks](#) element.

Each task enabled for the form must have a corresponding **task** element in the **allowedTasks** section of the [form definition file \(.xsf\)](#).

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **task** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval"/>
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate"/>
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response"/>
    <xsf:task name="getVPAApproval" taskTypeID="436"
      caption="Send Response"/>
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond"/>
  </xsf:allowedTasks>
</xsf:hwsWorkflow>
```

```
<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID="
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1"/>
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/a:some/b:thing" dataObject="Aux1"/>
```

```
</xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```


taskpane Element

Defines a [custom task pane](#) to be used in a Microsoft Office InfoPath 2003 form.

Type

xsd:complexType

Structure

| Name | Description |
|-------------------------|--|
| caption | (Required attribute) Defines the caption used in the task pane drop-down list box. |
| href | (Required attribute) Specifies the relative or absolute Uniform Resource Locator (URL) to an .html file. |

Remarks

The **taskpane** element is an optional element of the [xDocumentClass](#) element.

The task pane is a modeless panel that appears to the right of the main [form area](#). Task panes contain commands specific to completing tasks that are related to a single form. InfoPath supports the use of a single custom task pane.

To get multiple task pane behavior, you can include multiple .htm files in the form template and use the [Navigate](#) method of the [HTMLTaskPane](#) object in the Microsoft Office InfoPath 2003 object model to navigate to different .htm files.

Example

The following is an example of the **taskpane** element:

```
<xsf:xDocumentClass>
  ...
  <xsf:taskpane
    caption="Design Issues Help"
    href="DesignIssuesHelp.htm"/>
  ...
</xsf:xDocumentClass>
```


to Element

Contains the recipient information for the **to** line of the e-mail message when the form is submitted using the [emailAdapter](#) element.

Type

xsd:complexType

Parent Elements

| Element | Description |
|------------------------------|---|
| emailAdapter | Parent element that contains the information needed to submit an InfoPath form as an attachment to an e-mail message, with a specified set of recipients, a subject, and an introduction. |

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|---------------------------|-------------------------|----------|--|--|
| value | xsd:string | Yes | Contains the value of the to element. | string |
| valueType | xsf:xdExpressionLiteral | No | Specifies whether the value attribute is interpreted as an XPath expression or as a literal string. | <ul style="list-style-type: none">• expression• literal |

Definition

```
<xsd:element name="to" minOccurs="0">  
  <xsd:complexType>  
    <xsd:attribute name="value" type="xsd:string" use="required"/></xsd:  
    <xsd:attribute name="valueType" type="xsd:xdExpressionLiteral" u  
  </xsd:complexType>  
</xsd:element>
```

Remarks

Multiple addresses on the **to** line must be separated by semicolons.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **to** element:

```
<xsf:emailAdapter name="Submit" submitAllowed="yes">  
  <xsf:to value="someone@example.com" valueType="literal"/>  
  <xsf:cc value="my:ccNames" valueType="expression"/>  
  <xsf:bcc value="someoneelse@example.com" valueType="literal"/>  
  <xsf:subject value="My report" valueType="literal"/>  
  <xsf:intro value="See below"/>  
  <xsf:attachmentFileName value="Status Report" valueType="literal"/>  
</xsf:emailAdapter>
```

toolbar Element

Contains information about the toolbars used in the view.

Type

xsd:complexType

Structure

| Name | Description |
|--------------------------------|---|
| <u>button</u> | (Optional element) Defines a button that has an associated action. |
| <u>caption</u> | (Required attribute) Used as the title of the toolbar, when the toolbar is not docked to the user interface. |
| <u>name</u> | (Required attribute) Identifies the corresponding toolbar within different views. Must be unique within a given view. |
| <u>menu</u> | (Optional element) Contains information about the menus used within the toolbar. |

Remarks

The **toolbar** element is an optional element of the [view](#) element. There can be multiple toolbars declared for a view, and each toolbar can have multiple **menu** or **button** elements.

Example

The following is an example of the **toolbar** element:

```
<xsf:toolbar caption="CD Collection Toolbar"
  name="CD Collection Toolbar">
  <xsf:button action="xCollection::insert" xmlToEdit="CD_10"
    caption="New CD" showIf="always"></xsf:button>
  <xsf:button action="xCollection::insert" xmlToEdit="Track_14"
    caption="New Track" showIf="always"></xsf:button>
  <xsf:button action="xOptional::insert" xmlToEdit="Label_16"
    caption="New Label" showIf="always"></xsf:button>
  <xsf:menu caption="Remove">
    <xsf:button action="xCollection::remove" xmlToEdit="CD_10"
      caption="CD" showIf="always"></xsf:button>
    <xsf:button action="xCollection::remove" xmlToEdit="Track_14"
      caption="Track" showIf="always"></xsf:button>
    <xsf:button action="xOptional::remove" xmlToEdit="Label_16"
      caption="Label" showIf="always"></xsf:button>
  </xsf:menu>
</xsf:toolbar>
```


unboundControls Element

Defines the unbound button controls that are used in the view.

Type

xsd:complexType

Structure

| Name | Description |
|------------------------|--|
| button | (Optional element) Defines a button that has an associated action. |

Remarks

The **unboundControls** element is an optional element of the [view](#) element.

Unbound controls in Microsoft Office InfoPath 2003 are the buttons that are dragged from the **Controls** task pane onto the view in [design mode](#). The name assigned to the button control is used in the [name](#) attribute of the button element, and when a user clicks on the button, the scripting code associated with the button will be called.

Example

The following is an example of the **unboundControls** element:

```
<xsf:view>
  <xsf:unboundControls>
    <xsf:button name="MyButton"></xsf:button>
  </xsf:unboundControls>
  ...
</xsf:view>
```

The following script handler, included in the script file, will be called when the button is clicked:

```
function MyButton::OnClick()
{
  // Write your code here.
}
```


useHttpHandler Element

Specifies that the form is to be submitted to the specified [Uniform Resource Locator \(URL\)](#) using the specified HTTP method.

Type

xsd:complexType

Structure

| Name | Description |
|-------------------------------|---|
| <u>href</u> | (Required attribute) Specifies the URL to which the form should be submitted. |
| <u>method</u> | (Required attribute) Specifies the HTTP method to use for the submit operation. |

Remarks

The **useHttpHandler** element is an optional element of the [submit](#) element.

Example

The following is an example of the **useHttpHandler** element:

```
<xf:submit
  caption="Submit"
  disableMenuItem="no"
  onAfterSubmit="KeepOpen"
  showStatusDialog="yes"
  showSignatureReminder="yes">
  <xf:useHttpHandler>
    href="http://MyServer/InfoPathScripts/MyScript.asp"
    method="POST"
  </xf:useHttpHandler>
  <xf:successMessage>Submit was successful.</xf:successMessage>
  <xf:errorMessage>Submit was not successful.</xf:errorMessage>
</xf:submit>
```


useQueryAdapter Element

Specifies that the form is to be submitted to the same [data adapter](#) as specified in the [query](#) element.

Type

xsd:complexType

Remarks

The **useQueryAdapter** element is an optional element of the [submit](#) element.

The **query** element used is the one that is the child element of the [xDocumentClass](#) element, not the **query** element that is a child of the [dataObject](#) element.

The **useQueryAdapter** element does not contain any attributes or child elements.

Example

The following is an example of the **useQueryAdapter** element:

```
<xf:submit
  caption="Submit"
  disableMenuItem="no"
  onAfterSubmit="KeepOpen"
  showStatusDialog="yes"
  showSignatureReminder="yes">
  <xf:useQueryAdapter/>
  <xf:successMessage>Submit was successful.</xf:successMessage>
  <xf:errorMessage>Submit was not successful.</xf:errorMessage>
</xf:submit>
```

userName Element

Associates a user with a particular role.

Type

xsd:complexType

Child Elements

None.

Attributes

| Attribute | Type | Required | Description | Possible Values |
|--------------------------|------------|----------|--|-----------------|
| name | xsd:string | Yes | Specifies the name of a user for inclusion in the membership list of a role. | string |
| memberOf | xsd:string | Yes | Specifies the role to be associated with the user. | string |

Definition

```
<xsd:element name="userName">  
  <xsd:complexType>  
    <xsd:attribute name="name" type="xsd:string" use="required"/></xsd:  
    <xsd:attribute name="memberOf" type="xsd:string" use="required"  
  </xsd:complexType>  
</xsd:element>
```

Remarks

InfoPath requires that a user be associated with only one role at a time.

If a user creates a new InfoPath document from a form template, that user is assigned to the role specified in the optional [initiator](#) attribute of the [roles](#) element. If no role has been designated as the initiator role, InfoPath assigns the user to the role specified in the **memberOf** attribute of the first entry that corresponds to the user in the membership list of the **roles** element. An entry in the membership list corresponds to a user if any one of the following conditions is true:

The name of the user matches the value of the **name** attribute of a **userName** element in the membership list.

The user is included in a group identified by a [group](#) element in the membership list.

The name of the user matches a name returned by a [getUserNameFromData](#) element in the membership list.

If no entry for the user is found in the membership list, InfoPath associates the user with the role specified in the [default](#) attribute of the **roles** element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **userName** element:

```
<xsf:roles initiator="A" default="C" hideStatusBarDisplay="yes">
  <xsf:role name="A"/>
  <xsf:role name="B"/>
  <xsf:role name="C"/>

  <xsf:membership>
    <xsf:getUserNameFromData dataObject="catalog" select="/dfs:my
    <xsf:userName name="Domain\username1" memberOf="A"/>
    <xsf:userName name="Domain\username2" memberOf="B"/>
    <xsf:group name="Domain\username3" memberOf="C"/>
  </xsf:membership>
</xsf:roles>
```


useScriptHandler Element

Specifies that the submit, save, or version upgrade operation will be handled using form code.

Type

xsd:element

Remarks

The **useScriptHandler** element is an optional element of the [submit](#) element, the [save](#) element and the [documentVersionUpgrade](#) element. The **useScriptHandler** element contains no attributes or child elements.

When used with the **submit** element, it declares that the form data will be processed by script code in the associated form file. The submit form code must be written in the OnSubmitRequest [event handler](#) in the form's primary script file.

When used with the **save** element, it declares that the save operation of the form data will be processed by form code in the associated form code file. The code must be written in the OnSaveRequest event handler in the form's primary scripting file.

When used with the **documentVersionUpgrade** element, it declares that the upgrade of older forms (created with an older version of the [form template](#)) will be processed by form code in the associated scripting file. The form upgrade code must be written in the OnVersionUpgrade event handler in the form's primary script file.

Example

The following is an example of the **useScriptHandler** element:

```
<xsf:documentVersionUpgrade>  
  <xsf:useScriptHandler/>  
</xsf:documentVersionUpgrade>
```


useTransform Element

Specifies that the upgrade will be handled by an [XSL Transformation \(XSLT\)](#) supplied by the newer version of the [form template](#).

Type

xsd:complexType

Structure

| Name | Description |
|--|---|
| <u>maxToVersionUpgrade</u> | (Optional attribute) Inclusive value for the latest form that needs to be upgraded. |
| <u>minVersionToUpgrade</u> | (Required attribute) Inclusive value for the oldest form that can be upgraded. |
| <u>transform</u> | (Required attribute) Specifies the XSLT file name relative to the form template. |

Remarks

The **useTransform** element is an optional element of the [documentVersionUpgrade](#) element.

When a user fills out a form, Microsoft Office InfoPath 2003 automatically runs the specified XSLT on the form's underlying XML document and uses the output as the XML data to be edited, if the version of the form is greater than or equal to the **minVersionToUpgrade** attribute and the version is less than or equal to the **maxVersionToUpgrade** attribute.

Example

The following is an example of the **useTransform** element:

```
<xsf:documentVersionUpgrade>  
  <xsf:useTransform  
    transform="upgrade.xsl"  
    minVersionToUpgrade="0.0.0.0"  
    maxVersionToUpgrade="1.0.0.5"/>  
</xsf:documentVersionUpgrade>
```


view Element

Contains information about a Microsoft Office InfoPath 2003 view.

Type

xsd:complexType

Structure

| Name | Description |
|--|--|
| <u>caption</u> | (Optional attribute) Provides the display caption for the view within the view list. |
| <u>designMode</u> | (Optional attribute) Determines whether the view can be opened in design mode. |
| <u>editing</u> | (Optional element) Contains information about the editing components used in the view. |
| <u>mainpane</u> | (Required element) Determines what is displayed in the main pane (<u>form area</u>). |
| <u>menu</u> | (Optional element) Contains information about the custom menus used in the view. |
| <u>menuArea</u> | (Optional element) Contains information about the custom menu items that can be added to the InfoPath built-in menus. |
| <u>name</u> | (Required attribute) Identifies the view for object model calls when switching views, and for specifying the default view. |
| <u>printSettings</u> | (Optional element) Specifies the printer settings used when printing the view. |
| <u>printView</u> | (Optional attribute) Specifies the name of another view to use for printing this view. |
| <u>toolbar</u> | (Optional element) Contains information about the toolbars used in the view. |
| <u>unboundControls</u> | (Optional element) Defines the unbound button controls that are used in the view. |
| <u>showMenuItem</u> | (Optional element) Provides the display of the menu item in the View menu with a check mark when enabled. |

Remarks

The **view** element is a required element of the [views](#) element.

Example

The following is an example of the **view** element:

```
<xsf:views default="View">  
  <xsf:view name="View" caption="View">  
    <xsf:mainpane transform="view1.xsl"></xsf:mainpane>  
    ...  
  </xsf:view>  
</xsf:views>
```

views Element

Defines all of the views that have been implemented in a Microsoft Office InfoPath 2003 form.

Type

xsd:complexType

Structure

| Name | Description |
|-------------------------|--|
| default | (Optional attribute) Specifies the name of the view chosen to be the default view. |
| view | (Required element) Contains information about an InfoPath view. |

Remarks

The **views** element is a required element of the [xDocumentClass](#) element.

Example

The following is an example of the **views** element:

```
<xsf:views default="View">  
  <xsf:view name="View" caption="View">  
    <xsf:mainpane transform="view1.xsl"></xsf:mainpane>  
    ...  
  </xsf:view>  
</xsf:views>
```


webServiceAdapter Element

Defines a Web service [data adapter](#) that retrieves data from a Web service for the specified data object.

Type

xsd:complexType

Structure

| Name | Description |
|--------------------------------------|--|
| <u>queryAllowed</u> | (Optional attribute) Specifies whether data can be retrieved from the data source through the query method of the adapter object. |
| <u>submitAllowed</u> | (Optional attribute) Specifies whether data can be submitted to the data source through the submit method of the adapter object. |
| <u>wsdlUrl</u> | (Required attribute) Contains the <u>Uniform Resource Locator (URL)</u> of the Web Services Description Language (WSDL) file that describes the Web service specification. |
| <u>operation</u> | (Required element) Defines the operation (method) of the Web service to be used for retrieving and submitting data. |
| <u>useDataSet</u> | (Optional attribute) Specifies whether the adapter will support an ADO.Net DataSet. Default is "no". |

Remarks

The **webServiceAdapter** element is an optional element of the [query](#) element.

The **webServiceAdapter** element can also be used to define a Web service adapter used to submit the main or secondary form data.

Example

The following is an example of the **webServiceAdapter** element:

```
<xf:query>
  <xf:webServiceAdapter
    wsdlUrl="http://localhost/infopathwebservicessample/infopathwebs
    queryAllowed="yes"
    submitAllowed="no"
    useDataSet="no">
    <xf:operation
      name="getOrders"
      soapAction="http://tempuri.org/getOrders"
      serviceUrl="http://localhost/infopathwebservicessample/infopathv
      <xf:input
        source="Submit.xml">
      </xf:input>
    </xf:operation>
  </xf:webServiceAdapter>
</xf:query>
```


xDocumentClass Element

The root element of the [form definition \(.xsf\) file](#) file. Contains all other [elements](#) and [attributes](#) of the .xsf file.

Type

xsd:complexType

Structure

| Name | Description |
|--|---|
| <u>author</u> | (Optional attribute) Identifies the author of the form. |
| <u>dataFormSolution</u> | (Optional attribute) Identifies the form as a database form. |
| <u>description</u> | (Optional attribute) A brief description of the form. |
| <u>name</u> | (Optional attribute) Provides a unique, <u>Uniform Resource Name (URN)</u> -based name for the form that the .xsf file defines. |
| <u>productVersion</u> | (Optional attribute) Identifies the version number of InfoPath with which the form has been created or for which a particular form is intended. |
| <u>publishUrl</u> | (Optional attribute) Identifies where the form was published and where the form template should download updates from. |
| <u>requireFullTrust</u> | (Optional attribute) Allows the form to run as a fully trusted form when a form template is registered or signed with a certificate. |
| <u>solutionFormatVersion</u> | (Required attribute) Identifies the version number that represents the format of the .xsf file. |
| <u>solutionVersion</u> | (Optional attribute) Identifies the version number of the form. |
| <u>xmlns</u> | (Required attribute) Defines the <u>xsf</u> namespace. |
| <u>applicationParameters</u> | (Optional element) Contains form-specific properties that describe how a form should be used in InfoPath <u>design mode</u> . |
| <u>customValidation</u> | (Optional element) Defines rule-based custom validation on top of all validation |

| | |
|---|--|
| | enforced through the XML Schema . |
| dataObjects | (Optional element) Defines all secondary data objects used in an InfoPath form. |
| documentSchemas | (Optional element) Defines the XML Schemas that the InfoPath form is designed to handle. |
| documentSignatures | (Optional element) Defines the location of the digital signature XML Document Object Model (DOM) node within the form's underlying XML document. |
| documentVersionUpgrade | (Optional element) Defines how forms created with an older version of the form template can be upgraded to the latest version of the form template. |
| domEventHandlers | (Optional element) Contains pointers to various script-based event handlers that react to changes in XML DOM nodes of a form's underlying XML document while the form is being filled out. |
| extensions | (Optional element) Includes minor upgrades to the InfoPath .xsf file that can be used by future releases of InfoPath or by specific forms. |
| fileNew | (Optional element) Provides a reference to an .xml file containing sample data, to be loaded when a user chooses to create a new form based on the form template. |
| importParameters | (Optional element) Contains all the parameters that define how the import (merge) forms feature works for the form. |
| listProperties | (Optional element) Identifies the properties that should be on a list view of all forms belonging to a form template. |
| package | (Required element) Contains information about all of the files used in an InfoPath form. |

| | |
|--|--|
| <u>query</u> | (Optional element) Associates a <u>data adapter</u> with the form's underlying XML document. |
| <u>schemaErrorMessages</u> | (Optional element) Contains custom error messages used to override XML Schema data type errors. |
| <u>scripts</u> | (Optional element) Defines the source of all business logic scripts used at the document level in the form. |
| <u>submit</u> | (Optional element) Contains information about the submission functionality of a form. |
| <u>taskpane</u> | (Optional element) Defines a <u>custom task pane</u> to be used in an InfoPath form. |
| <u>views</u> | (Required element) Defines all of the views that have been implemented in an InfoPath form. |
| <u>trustLevel</u> | (Optional attribute) Specifies the trust level of a form. |
| <u>trustSetting</u> | (Optional attribute) Specifies the trust setting of a form. |

Remarks

The **xDocumentClass** element is a required element and must be present in the InfoPath .xsf file.

The attributes that are contained within the **xDocumentClass** element are collectively called the global metadata section. The global metadata section of the .xsf file is a required section that contains global information about the InfoPath form.

Example

The following is an example of the **xDocumentClass** element:

xsf:xDocumentClass

```
xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solut
name="urn:schemas-microsoft-com:myTravelReport"
author="Frank Miller"
description="Travel Report form for entering travel reports, issues, e
dataFormSolution="yes"
solutionVersion = "1.0.0.1"
productVersion="11.0.5106"
solutionFormatVersion="1.0.0.0">
dataFormSolution="yes"
requireFullTrust="yes"
trustLevel="restricted"
trustSetting="manual"
publishUrl="http://www.contoso.com/InfoPathTemplates/MyTempla
```

...

</xsf:xDocumentClass

xmlFileAdapter Element

Defines an .xml file [data adapter](#) that retrieves data from an .xml file for the specified data object.

Type

xsd:complexType

Structure

| Name | Description |
|-------------------------|--|
| fileUrl | (Required attribute) Contains the Uniform Resource Locator (URL) of the .xml file. |
| name | (Optional attribute) Contains the name of the xmlFileAdapter element. |

Remarks

The **xmlFileAdapter** element is an optional element of the [query](#) element.

Example

The following is an example of the **xmlFileAdapter** element:

```
<xsf:query>  
  <xsf:xmlFileAdapter fileUrl="currencies.xml" />  
</xsf:query>
```


xmlToEdit Element

Specifies an instance of an editing component.

Type

xsd:complexType

Structure

| Name | Description |
|------------------------------------|---|
| <u>container</u> | (Optional attribute) Specifies an XPath expression that determines the context in which the control will be selectable and its actions enabled. |
| <u>editWith</u> | (Optional element) Defines an instance of an editing component, and provides the corresponding parameters to determine its exact behavior. |
| <u>item</u> | (Required attribute) Specifies an XPath expression that determines the XML Document Object Model (DOM) nodes to be edited using the editing component defined in the editWith element. |
| <u>name</u> | (Required attribute) Used in the xmlToEdit attribute of the button element to associate actions of the associated editing component with buttons defined in menus and toolbars. |
| <u>viewContext</u> | (Optional attribute) Specifies a string that identifies an HTML element in the view. |

Remarks

The **xmlToEdit** element is an optional element of the [editing](#) element.

xmlToEdit elements are used to define the editing components that can be used in a form. The **xmlToEdit** elements can contain multiple **editWith** elements that specify the editing components that will be used to edit various types of XML DOM nodes.

Example

The following is an example of the **xmlToEdit** element:

```
<xf:editing>
  <xf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xf:editWith caption="CD"
      xd:autogeneration="template"
      component="xCollection">
      <xf:fragmentToInsert>
        <xf:chooseFragment parent="CDCollection">
          <CD>
            <Title></Title>
            <Artist></Artist>
            <Tracks>
              <Track></Track>
              <Track></Track>
            </Tracks>
          </CD>
        </xf:chooseFragment>
      </xf:fragmentToInsert>
    </xf:editWith>
  </xf:xmlToEdit>
</xf:editing>
```

action Attribute

Specifies an action of an editing component, using the syntax "NameOfEditingComponent::NameOfAction".

Type

xsd:NMTOKEN

Remarks

The **action** attribute is an optional attribute of the [button](#) element, but is required for buttons used with editing components.

The following are the editing actions that may be used as the value of the **action** attribute.

| Name | Description |
|---------------------------|--|
| xCollection::insert | Inserts an item. |
| xCollection::insertBefore | Inserts an item before the current selection. |
| xCollection::insertAfter | Inserts an item after the current selection. |
| xCollection::remove | Removes an item. |
| xCollection::removeAll | Removes all items. |
| xOptional::insert | Inserts an optional item. |
| xOptional::remove | Removes an optional item. |
| xReplace::replace | Replaces an existing item. |
| xFileAttachment::attach | Opens a dialog box to browse for files to attach to a form. |
| xFileAttachment::open | Opens a file attached to a form. |
| xFileAttachment::saveAs | Opens the a dialog box to browse for a location where a file is to be saved. |
| xFileAttachment::remove | Removes the file from the form. |

Example

The following is an example of the **action** attribute as it is used in the **button** element:

```
<xsf:menuArea name="msoInsertMenu">  
  <xsf:menu caption="&Section">  
    <xsf:button action="xCollection::insert" xmlToEdit="CD_10"  
      caption="CD" showIf="always"></xsf:button>  
    <xsf:button action="xCollection::insert" xmlToEdit="Track_14"  
      caption="Track" showIf="always"></xsf:button>  
    <xsf:button action="xOptional::insert" xmlToEdit="Label_16"  
      caption="Label"></xsf:button>  
  </xsf:menu>  
</xsf:menuArea>
```


actionTypeID Attribute

Contains a unique ID for the [action](#) element.

Type

xsd:string

Parent Elements

| Element | Description |
|------------------------|---|
| action | Contains the information for an action. |

Definition

```
<xsd:attribute name="actionTypeID" type="xsd:string" use="required"
```

Remarks

The **actionTypeID** attribute is an attribute of the **action** element.

The value of the **actionTypeID** attribute must be unique in the [form](#).

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **actionTypeID** attribute as it is used in the **action** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval" />
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate" />
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response" />
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response" />
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond" />
  </xsf:allowedTasks>
</xsf:hwsWorkflow>

<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes" queryAllowed="no">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID="
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1" />
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
```

```
        replaceWith="/a:some/b:thing" dataObject="Aux1"/>  
    </xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```

adapter Attribute (queryAction Element)

Contains the adapter name for the data connection used for the [queryAction](#) element.

Type

xsd:string

Parent Elements

| Element | Description |
|-----------------------------|---|
| queryAction | Defines a data connection query action. |

Definition

```
<xsd:attribute name="adapter" type="xsd:string" use="required" ></xs
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **adapter** attribute as it is used in the **queryAction** element:

```
<ksf:rule caption="Receipts" condition="my:expense > 75">
  <ksf:dialogBoxMessageAction>Don't forget your receipts!</ksf:dialogBoxMessageAction>
  <ksf:switchViewAction view="Approval View"/>
  <ksf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression ="sum(my:expenses/my:expense) * my:taxRate"/>
  <ksf:queryAction adapter="Exchange Rates"/>
  <ksf:submitAction adapter="Exchange Rates"/>
  <ksf:openNewDocumentAction solutionURI="uri:microsoft-Expense1">
  <ksf:closeDocumentAction promptToSaveChanges="yes"/>
  <ksf:dialogBoxExpressionAction>my:group/my:field1</ksf:dialogBoxExpressionAction>
  <ksf:exitRuleSet/>
</ksf:rule>
```


adapter Attribute (submitAction Element)

Contains the name of the data adapter to submit.

Type

xsf:xdTitle

Parent Elements

| Element | Description |
|------------------------------|--|
| submitAction | Defines a submit action for a form . |

Definition

```
<xsd:attribute name="adapter" type="xsf:xdTitle" use="required" ></x
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **adapter** attribute as it is used in the **submitAction** element:

```
<xf:rule caption="Receipts" condition="my:expense > 75">
  <xf:dialogBoxMessageAction>Don't forget your receipts!</xf:dialogBoxMessageAction>
  <xf:switchViewAction view="Approval View"/>
  <xf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression="sum(my:expenses/my:expense) * my:taxRate"/>
  <xf:queryAction adapter="Exchange Rates"/>
  <xf:submitAction adapter="Exchange Rates"/>
  <xf:openNewDocumentAction solutionURI="uri:microsoft-Expense1">
  <xf:closeDocumentAction promptToSaveChanges="yes"/>
  <xf:dialogBoxExpressionAction>my:group/my:field1</xf:dialogBoxExpressionAction>
  <xf:exitRuleSet/>
</xf:rule>
```


adapter Attribute (submitAction Element)

Contains the name of the data adapter used for submitting the [form](#).

Type

xsd:complexType

Parent Elements

| Element | Description |
|------------------------------|--|
| submitAction | Specifies the data adapter used to submit the form . |

Definition

```
<xsd:element name="submitAction" minOccurs="0" >
  <xsd:complexType>
    <xsd:attribute name="adapter" type="xsf:xdTitle" use="required"><
  </xsd:complexType>
  <xsd:keyref name="submitAdapter_name_keyref" refer="xsf:adapter
    <xsd:selector xpath="." />
    <xsd:field xpath="@adapter" />
  </xsd:keyref>
</xsd:element>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **adapter** attribute as it is used in the **submitAction** element:

```
<xf:submit caption="Submit" disableMenuItem="no" onAfterSubmit="onAfterSubmit"
  <xf:submitAction adapter="dav"/>
</xf:submit>
<xf:dataAdapters>
  <xf:davAdapter name="dav" folderUrl="http://some_server/some_dir" />
</xf:dataAdapters>
```


aggregation Attribute

Specifies how the nodes returned from an [XPath](#) expression in the [node](#) attribute should be aggregated to obtain a single value for the document.

Type

xsd:simpleType

Remarks

The **aggregation** attribute is an optional attribute of the [field](#) element.

The **aggregation** attribute can either be an aggregation action or an indication of the particular element in the collection. It uses the following aggregation actions: sum, count, average, min, max, first, last, merge, and plaintext.

Example

The following is an example of the **aggregation** attribute as it is used in the **field** element:

```
<xsf:listProperties>
  <xsf:fields>
    <xsf:field
      type="xsd:date"
      name="TravelDate"
      columnName="TravelDate"
      required="yes"
      viewable="yes"
      node="TravelReport/Header/travelDate"
      aggregation="first"/>
  </xsf:fields>
</xsf:listProperties>
```


allowCustomization Attribute

Identifies whether the Microsoft Office InfoPath 2003 form can be modified or customized.

Type

xsf:xdYesNo

Remarks

The **allowCustomization** attribute is an optional attribute of the [solutionProperties](#) element.

If the **allowCustomization** attribute is set to "no", an error is generated when a user tries to modify the [form template](#) after opening it in InfoPath. Allowed values are "yes" and "no". The default value is "yes".

Example

The following is an example of the **allowCustomization** attribute as it is used in the **solutionProperties** element:

```
<xsf: applicationParameters application="InfoPath Design Mode">  
  <xsf: solutionProperties  
    allowCustomization="no"  
    lastOpenView="view1"  
    scriptLanguage="JScript"  
    automaticallyCreateNodes="no"  
    lastVersionNeedingTransform="1.1.0.10"  
    fullyEditableNamespace="urn:namespace1:mynames"/>  
</xsf:applicationParameters>
```


allowedFileTypes attribute

Specifies the file name extensions of files that can be attached to the [form](#).

Type

xsd:string

Parent Elements

| Element | Description |
|--------------------------|---|
| editWith | Specifies an instance of an editing component, and provides the corresponding parameters to determine its exact behavior. |

Definition

```
<xsd:attribute name="allowedFileTypes" type="xsd:string" use="optio
```

Remarks

The file extensions listed in the **allowedFileTypes** attribute value are separated by commas.

Example

```
<xsf:xmlToEdit name="some_name" item="/my:myFields/my:field1'  
<xsf:editWith allowedFileTypes="doc, xls" component="xFileAttach  
</xsf:xmlToEdit>
```

```
<menuArea name="msoStructuralEditingContextMenu">  
<button action="xFileAttachment::attach" xmlToEdit="some_name" c  
<button action="xFileAttachment::open" xmlToEdit="some_name" ca  
<button action="xFileAttachment::saveAs" xmlToEdit="some_name"  
<button action="xFileAttachment::remove" xmlToEdit="some_name"  
</menuArea>
```

application Attribute

Identifies the name of the application used to design the Microsoft Office InfoPath 2003 form.

Type

xsd:string

Remarks

The **application** attribute is a required attribute of the [applicationParameters](#) element.

The only value supported is "InfoPath Design Mode".

Example

The following is an example of the **application** attribute as it is used in the **applicationParameters** element:

```
<xsf: applicationParameters application="InfoPath Design Mode">  
  <xsf: solutionProperties  
    allowCustomization="no"  
    lastOpenView="view1"  
    scriptLanguage="JScript"  
    automaticallyCreateNodes="no"  
    lastVersionNeedingTransform="1.1.0.10"  
    fullyEditableNamespace="urn:namespace1:mynames"/>  
</xsf:applicationParameters>
```

attribute Attribute

Specifies the name of the attribute to be inserted.

Type

xsd:string

Remarks

The **attribute** attribute is a required attribute of the [attributeData](#) element.

Example

The following is an example of the **attribute** attribute as it is used in the **attributeData** element:

```
<xsf:editWith component="xOptional">  
  <xsf:fragmentToInsert>  
    <xsf:chooseFragment parent="report">  
      <xsf:attributeData attribute="author" value="author name"/>  
    </xsf:chooseFragment>  
  </xsf:fragmentToInsert>  
</xsf:editWith>
```

author Attribute

Identifies the author of the form.

Type

xsd:string

Remarks

The **author** attribute is an optional attribute of the [xDocumentClass](#) element.

Example

The following is an example of the **author** attribute as it is used in the **xDocumentClass** element:

```
<xsf:xDocumentClass
  xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solut
  name="urn:microsoft-com:myTravelReport"
  author="AuthorName"
  description="Travel Report form for entering travel reports, issues, e
  dataFormSolution="yes"
  solutionVersion = "1.0.0.1"
  productVersion="11.0.5106"
  solutionFormatVersion="1.0.0.0">
  ...
</xsf:xDocumentClass>
```

autoComplete Attribute

Switches the auto-completion of fields on or off.

Type

xsf:xdYesNo

Remarks

The **autoComplete** attribute is an optional attribute of the [editWith](#) element.

Values include "yes" and "no". The default value is "yes".

Example

The following is an example of the **autoComplete** attribute as it is used in the **editWith** element:

```
<xsf:editing>
  <xsf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xsf:editWith caption="CD"
      autoComplete="yes"
      component="xCollection">
      <xsf:fragmentToInsert>
        <xsf:chooseFragment parent="CDCollection">
          <CD>
            <Title></Title>
            <Artist></Artist>
            <Tracks>
              <Track></Track>
              <Track></Track>
            </Tracks>
          </CD>
        </xsf:chooseFragment>
      </xsf:fragmentToInsert>
    </xsf:editWith>
  </xsf:xmlToEdit>
</xsf:editing>
```


automaticallyCreateNodes Attribute

Identifies whether XML [Document Object Model \(DOM\)](#) nodes will be automatically generated when controls are inserted in the view in [design mode](#).

Type

xsf:xdYesNo

Remarks

The **automaticallyCreateNodes** attribute is an optional attribute of the [solutionProperties](#) element.

The **automaticallyCreateNodes** attribute corresponds to the **Automatically create data source** check box at the bottom of the **Controls** task pane.

Example

The following is an example of the **automaticallyCreateNodes** attribute as it is used in the **solutionProperties** element:

```
<xsf: applicationParameters application="InfoPath Design Mode">  
  <xsf: solutionProperties  
    allowCustomization="no"  
    lastOpenView="view1"  
    scriptLanguage="JScript"  
    automaticallyCreateNodes="no"  
    lastVersionNeedingTransform="1.1.0.10"  
    fullyEditableNamespace="urn:namespace1:mynames"/>  
</xsf:applicationParameters>
```

bottomMargin Attribute

Specifies the bottom margin when printing a view.

Type

xsd:string

Remarks

The **bottomMargin** attribute is an optional attribute of the [printSettings](#) element.

The **bottomMargin** attribute must be greater than or equal to zero.

Example

The following is an example of the **bottomMargin** attribute as it is used in the **printSettings** element:

```
<xsf:view name="View" caption="View">
  <xsf:printSettings
    header="Header text goes here."
    footer="Footer text goes here."
    orientation="portrait"
    marginUnitsType="in"
    topMargin="1"
    leftMargin="2"
    rightMargin="2"
    bottomMargin="1"
  </xsf:printSettings>
  ...
</xsf:view>
```

cabFile Attribute

Specifies the name of the CAB file.

Type

xsd:string

Parent Elements

| Element | Description |
|--------------------------------|---|
| allowedControl | Specifies the ActiveX controls that are allowed to be instantiated. |

Definition

```
<xsd:attribute name="cabFile" type="xsd:string" use="optional" ></xs
```

Remarks

The **cabFile** attribute must refer to a file within the InfoPath package. If the **cabFile** attribute is not specified, the control must already be registered on the local computer or the form will not open.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **cabFile** attribute as it is used in the **allowedControl** element:

```
<xsf:permissions>
  <xsf:allowedControl
    cabFile="{84F32C01-78D8-4B93-8ED4-106DA70224C2}.cab"
    clsid="{84F32C01-78D8-4B93-8ED4-106DA70224C2}"
    version="1,0,0,1" />
  <xsf:allowedControl
    clsid="{F08DF954-8592-11D1-B16A-00C0F0283630}" />
</xsf:permissions>
```

canInitiateWorkflow Attribute

Specifies whether the [action](#) element can be used to initiate or extend an activity flow.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|------------------------|---|
| action | Contains the information for an action. |

Definition

`<xsd:attribute name="canInitiateWorkflow" type="xsd:boolean" use="optional" />`

Remarks

The **canInitiateWorkflow** attribute is a required attribute of the **action** element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **canInitiateWorkflow** attribute as it is used in the **action** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval" />
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate" />
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response" />
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response" />
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond" />
  </xsf:allowedTasks>
</xsf:hwsWorkflow>
```

```
<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes" queryAllowed="no">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeId=
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1" />
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
```

```
        replaceWith="/a:some/b:thing" dataObject="Aux1" />  
    </xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```

caption Attribute (action Element)

Specifies a label for the button in the Workflow task pane corresponding to the [action](#) element.

Type

xsf:xdHWSCaption

Parent Elements

| Element | Description |
|------------------------|---|
| action | Contains the information for an action. |

Definition

```
<xsd:attribute name="caption" type="xsf:xdHWSCaption" use="option
```

Remarks

The **caption** attribute is an optional attribute of the **action** element.

If a **caption** is not specified, the value of the **name** attribute will be used as the caption for the button in the Workflow task pane.

The **caption** attribute is also used to extend an activity flow.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **caption** attribute as it is used in the **action** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval" />
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate" />
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response" />
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response" />
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond" />
  </xsf:allowedTasks>
</xsf:hwsWorkflow>

<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes" queryAllowed="no">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID="
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1" />
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
```

```
        replaceWith="/a:some/b:thing" dataObject="Aux1"/>  
    </xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```

caption Attribute (button Element)

Provides the caption displayed on the button.

Type

xsf:xdTitle

Remarks

The **caption** attribute is an optional attribute of the [button](#) element.

Example

The following is an example of the **caption** attribute as it is used in the **button** element:

```
<xsf:menuArea name="msoInsertMenu">
  <xsf:menu caption="&Section">
    <xsf:button action="xCollection::insert" xmlToEdit="CD_10"
      caption="CD" showIf="always"></xsf:button>
    <xsf:button action="xCollection::insert" xmlToEdit="Track_14"
      caption="Track" showIf="always"></xsf:button>
    <xsf:button action="xOptional::insert" xmlToEdit="Label_16"
      caption="Label"></xsf:button>
  </xsf:menu>
</xsf:menuArea>
```

caption Attribute (editWith Element)

Specifies an identifier for alternate forms of XML data to be used in the editing component.

Type

xsf:xdTitle

Remarks

The **caption** attribute is an optional attribute of the [editWith](#) element.

Example

The following is an example of the **caption** attribute as it is used in the **editWith** element:

```
<xf:editing>
  <xf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xf:editWith caption="CD"
      autoComplete="yes"
      component="xCollection">
      <xf:fragmentToInsert>
        <xf:chooseFragment parent="CDCollection">
          <CD>
            <Title></Title>
            <Artist></Artist>
            <Tracks>
              <Track></Track>
              <Track></Track>
            </Tracks>
          </CD>
        </xf:chooseFragment>
      </xf:fragmentToInsert>
    </xf:editWith>
  </xf:xmlToEdit>
</xf:editing>
```

caption Attribute (initialXmlDocument Element)

Defines the text string to be used as the name of the form in the Template Gallery and in the most recently used list.

Type

xsf:xdTitle

Remarks

The **caption** attribute is a required attribute of the [initialXmlDocument](#) element.

If this attribute is not specified, the name of the form is used as the caption. The maximum size for the caption text string is 255 characters.

Example

The following is an example of the **caption** attribute as it is used in the **initialXmlDocument** element:

```
<xsf:fileNew>  
  <xsf:initialXmlDocument  
    caption="Travel Report"  
    href="TravelReportTemplate.xml"/>  
</xsf:fileNew>
```

caption Attribute (menu Element)

Used as the caption for a menu.

Type

xsf:xdTitle

Remarks

The **caption** attribute is a required attribute of the [menu](#) element.

When used for a menu that is nested within a view, the caption is the top-level menu caption; when used for a menu that is nested within a toolbar, the caption is for the button on a drop-down menu; and when used for a menu that is nested inside another menu (cascading menu), the caption is the sub-menu caption.

Example

The following is an example of the **caption** attribute as it is used in the **menu** element:

```
<xsf:menuArea name="msoInsertMenu">
  <xsf:menu caption="&Section">
    <xsf:button action="xCollection::insert" xmlToEdit="CD_10"
      caption="CD" showIf="always"></xsf:button>
    <xsf:button action="xCollection::insert" xmlToEdit="Track_14"
      caption="Track" showIf="always"></xsf:button>
    <xsf:button action="xOptional::insert" xmlToEdit="Label_16"
      caption="Label"></xsf:button>
  </xsf:menu>
</xsf:menuArea>
```


caption Attribute (rule Element)

Contains the name of a [rule](#) as it appears in Microsoft Office InfoPath.

Type

xsd:string

Parent Elements

| Element | Description |
|----------------------|--|
| rule | Defines an action invoked after an event has occurred in the form. |

Definition

```
<xsd:attribute name="caption" type="xsd:string" use="required" ></xs
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **caption** attribute as it is used in the **rule** element:

```
<xsf:ruleSets>
  <xsf:ruleSet name="RuleSet1">
    <xsf:rule caption="Receipts" condition="my:expense > 75">
      <xsf:dialogBoxMessageAction>Don't forget receipts!</xsf:dialogBo
      <xsf:openNewDocumentAction solution="urn:approvalForm"/>
      <xsf:exitRuleSet/>
    </xsf:rule>
    <xsf:rule caption="Always Submit" isEnabled="no">
      <xsf:submitAction adapter="Expense Database"/>
    </xsf:rule>
  </xsf:ruleSet>

  <xsf:ruleSet name="RuleSet2">
    <xsf:rule caption="Look up contact">
      <xsf:queryAction adapter="Contacts"/>
    </xsf:rule>
  </xsf:ruleSet>
</xsf:ruleSets>
```

caption Attribute (submit Element)

Defines the name of the submit button and corresponding menu item that will appear on the **File** menu when a user is filling out a form.

Type

xsd:string

Remarks

The **caption** attribute is an optional attribute of the [submit](#) element.

The default value is "Submit", but any text string can be used. If the **caption** attribute is missing or is set to an empty string, the default value is used. The "&" characters can be used to create a keyboard shortcut for the submit menu item.

Example

The following is an example of the **caption** attribute as it is used in the **submit** element:

```
<xf:submit
  caption="Submit"
  disableMenuItem="no"
  onAfterSubmit="KeepOpen"
  showStatusDialog="yes"
  showSignatureReminder="yes">
  <xf:useScriptHandler/>
  <xf:successMessage>Submit was successful.</xf:successMessage>
  <xf:errorMessage>Submit was not successful.</xf:errorMessage>
</xf:submit>
```


caption Attribute (task Element)

Specifies a label for the button in the **Workflow** task pane corresponding to the [task](#) element.

Type

xsf:xdHWSCaption

Parent Elements

| Element | Description |
|----------------------|---|
| task | The Human Workflow Services (HWS) task information enabled for the form . |

Definition

```
<xsd:attribute name="caption" type="xsf:xdHWSCaption" use="option
```

Remarks

The **caption** attribute is an optional attribute of the **task** element.

If a **caption** is not specified, the value of the **name** attribute will be used as the caption for the button in the **Workflow** task pane.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **caption** attribute as it is used in the **task** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval" />
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate" />
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response" />
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response" />
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond" />
  </xsf:allowedTasks>
</xsf:hwsWorkflow>

<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes" queryAllowed="no">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID=
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1" />
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
```

```
        replaceWith="/a:some/b:thing" dataObject="Aux1" />  
    </xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```

caption Attribute (taskpane Element)

Defines the caption used in the task pane drop-down list box.

Type

xsd:string

Remarks

The **caption** attribute is a required attribute of the [taskpane](#) element.

If the **caption** attribute contains an empty value, then the default caption of "Form Tasks" is used.

Example

The following is an example of the **caption** attribute as it is used in the **taskpane** element:

```
<xsf:xDocumentClass>  
  ...  
  <xsf:taskpane  
    caption="My Custom Task Pane"  
    href="MyCustomTaskPane.htm"/>  
  ...  
</xsf:xDocumentClass>
```

caption Attribute (toolbar Element)

Used as the title of the toolbar, when the toolbar is not docked to the user interface.

Type

xsf:xdTitle

Remarks

The **caption** attribute is a required attribute of the [toolbar](#) element.

Note The **caption** attribute is also used for the name of the toolbar in the **Customize Commands** dialog box for a control.

Example

The following is an example of the **caption** attribute as it is used in the **toolbar** element:

```
<xsf:toolbar caption="CD Collection Toolbar"
  name="CD Collection Toolbar">
  <xsf:button action="xCollection::insert" xmlToEdit="CD_10"
    caption="New CD" showIf="always"></xsf:button>
  <xsf:button action="xCollection::insert" xmlToEdit="Track_14"
    caption="New Track" showIf="always"></xsf:button>
  <xsf:button action="xOptional::insert" xmlToEdit="Label_16"
    caption="New Label" showIf="always"></xsf:button>
  <xsf:menu caption="Remove">
    <xsf:button action="xCollection::remove" xmlToEdit="CD_10"
      caption="CD" showIf="always"></xsf:button>
    <xsf:button action="xCollection::remove" xmlToEdit="Track_14"
      caption="Track" showIf="always"></xsf:button>
    <xsf:button action="xOptional::remove" xmlToEdit="Label_16"
      caption="Label" showIf="always"></xsf:button>
  </xsf:menu>
</xsf:toolbar>
```

caption Attribute (view Element)

Provides the display caption for the view within the view list.

Type

xsf:xdViewName

Remarks

The **caption** attribute is an optional attribute of the [view](#) element.

If not specified, the caption of the view defaults to the value of the [name](#) attribute.

Example

The following is an example of the **caption** attribute as it is used in the **view** element:

```
<xsf:views default="View">  
  <xsf:view name="View" caption="View">  
    <xsf:mainpane transform="view1.xsl"></xsf:mainpane>  
    ...  
  </xsf:view>  
</xsf:views>
```

clsid Attribute

Specifies the COM class ID (CLSID) of an ActiveX control.

Type

xsd:string

Parent Elements

| Element | Description |
|--------------------------------|---|
| allowedControl | Specifies the ActiveX controls that are allowed to be instantiated. |

Definition

```
<xsd:attribute name="clsid" type="xsd:string" use="required" ></xsd:
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

When the view contains an **OBJECT** tag, the control will be instantiated only if the CLSID is listed as an **allowedControl** element in the **permissions** element. If an ActiveX control with this CLSID is not present on the system and can't be installed, the form will not open.

Example

The following is an example of the **clsid** attribute as it is used in the **allowedControl** element:

```
<xsf:permissions>
  <xsf:allowedControl
    cabFile="{84F32C01-78D8-4B93-8ED4-106DA70224C2}.cab"
    clsid="{84F32C01-78D8-4B93-8ED4-106DA70224C2}"
    version="1,0,0,1" />
  <xsf:allowedControl
    clsid="{F08DF954-8592-11D1-B16A-00C0F0283630}" />
</xsf:permissions>
```

collate attribute

Specifies whether the paper is collated.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|-------------------------------|---|
| printSettings | Specifies the printer settings used when printing a view. |

Definition

```
<xsd:attribute name="collate" type="xsd:YesNo" ></xsd:attribute>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **collate** attribute as it is used in the **printSettings** element:

```
<xf:printSettings
  orientation="landscape"
  header="&Pqsdf"
  footer="&D"
  printerName="\\printserver\printer"
  paperSource="Auto Select"
  paperSize="Envelope DL"
  topMargin="0.8"
  leftMargin="0.8"
  bottomMargin="0.8"
  rightMargin="0.8"
  marginUnitsType="in"
  copies="2"
  collate="no"
  pageRangeStart="1"
  pageRangeEnd="1" >
  <xf:header>
    <font>
      <div>&Pqsdf</div>
    </font>
  </xf:header>
  <xf:footer>
    <font>
      <div>&D</div>
    </font>
  </xf:footer>
</xf:printSettings>
```

columnName Attribute

Identifies the column name in the SQL table (underlying the form list view).

Type

xsf:xdTitle

Remarks

The **columnName** attribute is a required attribute of the [field](#) element.

Example

The following is an example of the **columnName** attribute as it is used in the **field** element:

```
<xsf:listProperties>
  <xsf:fields>
    <xsf:field
      type="xsd:date"
      name="TravelDate"
      columnName="TravelDate"
      required="yes"
      viewable="yes"
      node="TravelReport/Header/travelDate"
      aggregation="first"/>
  </xsf:fields>
</xsf:listProperties>
```


commandText Attribute

A string property that contains the [ActiveX Data Objects \(ADO\)](#) SQL command text to be used for querying the data from the specified data source.

Type

xsd:string

Remarks

The **commandText** attribute is a required attribute of the [adoAdapter](#) element.

Example

The following is an example of the **commandText** attribute as it is used in the **adoAdapter** element:

```
<xsf:query>
  <xsf:adoAdapter
    connectionString="Provider=Microsoft.Jet.OLEDB.4.0;
      Password=&quot;&quot;;User ID=Admin;
      Data Source=infwind.mdb;Mode=Share Deny None;
      Extended Properties=&quot;&quot;;..."
    commandText="select [EmployeeID],[LastName],[FirstName]
      from [Employees] as [Employees]"
    queryAllowed="yes"
    submitAllowed="yes">
  </xsf:adoAdapter>
</xsf:query>
```

component Attribute

Specifies the name of the editing component that will be referenced within the [action](#) attribute of a [button](#) element.

Type

xsd:enumeration

Remarks

The **component** attribute is a required attribute of the [editWith](#) element.

Valid component names include xCollection, xOptional, xReplace, xTextList, xField, xImage, xFileAttachment.

Example

The following is an example of the **component** attribute as it is used in the **editWith** element:

```
<xsf:editing>
  <xsf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xsf:editWith caption="CD"
      autoComplete="yes"
      component="xCollection">
      <xsf:fragmentToInsert>
        <xsf:chooseFragment parent="CDCollection">
          <CD>
            <Title></Title>
            <Artist></Artist>
            <Tracks>
              <Track></Track>
              <Track></Track>
            </Tracks>
          </CD>
        </xsf:chooseFragment>
      </xsf:fragmentToInsert>
    </xsf:editWith>
  </xsf:xmlToEdit>
</xsf:editing>
```

The following is an example of the **component** attribute as it is used in the **editWith** element, where the value of the **component** attribute is "xFileAttachment":

```
<xsf:xmlToEdit name="some_name" item="/my:myFields/my:field1"
```

```
<xsf:editWith allowedFileTypes="doc, xls" component="xFileAttach
</xsf:xmlToEdit>
```

```
<menuArea name="msoStructuralEditingContextMenu">
  <button action="xFileAttachment::attach" xmlToEdit="some_name"
  <button action="xFileAttachment::open" xmlToEdit="some_name" c
  <button action="xFileAttachment::saveAs" xmlToEdit="some_name
  <button action="xFileAttachment::remove" xmlToEdit="some_name
</menuArea>
```


condition Attribute

Defines an [XPath](#) expression evaluated as a **Boolean** value to determine whether the associated action will be invoked.

Type

xsd:string

Parent Elements

| Element | Description |
|----------------------|--|
| rule | Defines an action invoked after an event has occurred in the form. |

Definition

```
<xsd:attribute name="condition" type="xsd:string" use="optional" ></
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **condition** attribute as it is used in the **rule** element:

```
<xsf:ruleSets>
  <xsf:ruleSet name="RuleSet1">
    <xsf:rule caption="Receipts" condition="my:expense > 75">
      <xsf:dialogBoxMessageAction>Don't forget receipts!</xsf:dialogBo
      <xsf:openNewDocumentAction solution="urn:approvalForm"/>
      <xsf:exitRuleSet/>
    </xsf:rule>
    <xsf:rule caption="Always Submit" isEnabled="no">
      <xsf:submitAction adapter="Expense Database"/>
    </xsf:rule>
  </xsf:ruleSet>

  <xsf:ruleSet name="RuleSet2">
    <xsf:rule caption="Look up contact">
      <xsf:queryAction adapter="Contacts"/>
    </xsf:rule>
  </xsf:ruleSet>
</xsf:ruleSets>
```


connectionString Attribute

A string property that contains the [ActiveX Data Objects \(ADO\)](#) connection string to be used to connect to the data source.

Type

xsd:string

Remarks

The **connectionString** attribute is a required attribute of the [adoAdapter](#) element.

Example

The following is an example of the **connectionString** attribute as it is used in the **adoAdapter** element:

```
<xf:query>
  <xf:adoAdapter
    connectionString="Provider=Microsoft.Jet.OLEDB.4.0;
      Password=&quot;&quot;;User ID=Admin;
      Data Source=infwind.mdb;Mode=Share Deny None;
      Extended Properties=&quot;&quot;;..."
    commandText="select [EmployeeID],[LastName],[FirstName]
      from [Employees] as [Employees]"
    queryAllowed="yes"
    submitAllowed="yes">
  </xf:adoAdapter>
</xf:query>
```


container Attribute

Specifies an [XPath](#) expression that determines the context in which the control will be selectable and its actions enabled.

Type

xsd:string

Remarks

The **container** attribute is an optional attribute of the [xmlToEdit](#) element.

If the current context (view selection or insertion point) is within some HTML element that maps back to an XML [Document Object Model \(DOM\)](#) node that satisfies the container XPath expression, the control is enabled. Otherwise, all actions are disabled.

It does not suffice for the container XML DOM node to exist. Actions will be enabled only when the current selection is within an HTML element that maps to the container XML DOM node.

Note In Microsoft Office InfoPath 2003 [design mode](#), generated XPath expressions for the [item](#) and **container** attributes will always be in the form `item="/a/b/c"`, which provides the full path from the root element of the form's underlying XML document. But patterns such as `item="b/c"`, or with predicates as in `item="b[@p='23']/c[q]"` are allowed if the [form definition \(.xsf\) file](#) is edited manually.

Example

The following is an example of the **container** attribute as it is used in the **xmlToEdit** element:

```
<xsf:editing>
  <xsf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xsf:editWith caption="CD"
      xd:autogeneration="template"
      component="xCollection">
      <xsf:fragmentToInsert>
        <xsf:chooseFragment parent="CDCollection">
          <CD>
            <Title></Title>
            <Artist></Artist>
            <Tracks>
              <Track></Track>
              <Track></Track>
            </Tracks>
          </CD>
        </xsf:chooseFragment>
      </xsf:fragmentToInsert>
    </xsf:editWith>
  </xsf:xmlToEdit>
</xsf:editing>
```

copies attribute

Specifies the number of copies to be printed.

Type

xsd:Integer

Parent Elements

| Element | Description |
|-------------------------------|---|
| printSettings | Specifies the printer settings used when printing the view. |

Usage

```
<xsf:printSettings
  orientation="landscape"
  header="&Pqsdf"
  footer="&D"
  printerName="\\printserver\printer"
  paperSource="Auto Select"
  paperSize="Envelope DL"
  topMargin="0.8"
  leftMargin="0.8"
  bottomMargin="0.8"
  rightMargin="0.8"
  marginUnitsType="in"
copies="2"
  collate="no"
  pageRangeStart="1"
  pageRangeEnd="1" >
  <xsf:header>
    <font>
      <div>&Pqsdf</div>
    </font>
  </xsf:header>
  <xsf:footer>
    <font>
      <div>&D</div>
    </font>
  </xsf:footer>
</xsf:printSettings>
```

Definition

```
<xsd:attribute name="copies" >  
  <xsd:simpleType>  
    <xsd:restriction base="xsd:integer">  
      <xsd:minInclusive value="1" />  
      <xsd:maxInclusive value="9999" />  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:attribute>
```


data Attribute

Contains an [XPath](#) match expression that defines the nodeset to which the signature will be applied.

Type

xsd:string

Parent Elements

| Element | Description |
|---------------------------------|---|
| signedDataBlock | Defines a nodeset in the form's underlying XML document to which a digital signature can be applied |

Definition

```
<xsd:attribute name="data" type="xsd:string" use="required" ></xsd:a
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **data** attribute as it is used in the **signedDataBlock** element:

```
<xf:documentSignatures>
  <xf:signedDataBlock name="main"
    data="my:myfields/my:subtree1 | my:myfields/my:subtree2"
    signatureLocation="my:mifields/sig:signatures/main"
    mode="countersign">
    <xf:message>By pressing the "Sign" button below, I agr
  </xf:signedDataBlock>
</xf:documentSignatures>
```


dataFormSolution Attribute

Identifies the form as a database form.

Type

xsf:xdYesNo

Remarks

The **dataFormSolution** attribute is an optional attribute of the [xDocumentClass](#) element.

Allowed values are "yes" and "no". The default value is "no". Forms based on [ActiveX Data Objects \(ADO\)](#) or Web services and that have a special query view should have this attribute set to "yes" to work properly in Microsoft Office InfoPath 2003.

Example

The following is an example of the **dataFormSolution** attribute as it is used in the **xDocumentClass** element:

```
<xsf:xDocumentClass
  xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solut
  name="urn:microsoft-com:myTravelReport"
  author="AuthorName"
  description="Travel Report form for entering travel reports, issues, e
  dataFormSolution="yes"
  solutionVersion = "1.0.0.1"
  productVersion="11.0.5106"
  solutionFormatVersion="1.0.0.0">
  ...
</xsf:xDocumentClass>
```


dataObject Attribute (domEventHandler Element)

Contains the name of the [dataObject](#) element to be used in the event handler.

Type

xsd:string

Parent Elements

| Element | Description |
|---------------------------------|--|
| domEventHandler | Defines an event handler for one or more specific XML Document Object Model (DOM) nodes. |

Definition

```
<xsd:attribute name="dataObject" type="xsd:string" use="optional" ><
```

Remarks

The **dataObject** attribute is an optional attribute of the **domEventHandler** element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **dataObject** attribute as it is used in the **domEventHandler** element:

```
<xsf:domEventHandlers>  
  <xsf:domEventHandler  
    match="TravelReport/Expenses"  
    handlerObject="TravelExpenses"  
    dataObject="dataObject1"/>  
</xsf:domEventHandlers>
```

dataObject Attribute (getUserNameFromData Element)

The name of the secondary data source where the user name name can be found.

Type

xsd:string

Parent Elements

| Element | Description |
|-------------------------------------|--|
| getUserNameFromData | Retrieves a user name by using an XPath query of the data in the main data source or in a secondary data source. |

Definition

```
<xsd:attribute name="dataObject" type="xsd:string" use="optional" ><
```

Remarks

For the main data source, this attribute should not be present. An empty string is invalid.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **dataObject** attribute as it is used in the **getUserUserNameFromData** element:

```
<xsf:roles initiator="A" default="C" hideStatusBarDisplay="yes">
  <xsf:role name="A" />
  <xsf:role name="B" />
  <xsf:role name="C" />

  <xsf:membership>
    <xsf:getUserNameFromData dataObject="catalog" select="/dfs:r
    <xsf:userName name="domain\username1" memberOf="A" />
    <xsf:userName name="domain\username2" memberOf="B" />
    <xsf:group name="domain\groupname1" memberOf="C" />
  </xsf:membership>
</xsf:roles>
```

dataObject Attribute (partFragment Element)

Specifies the **name** of the data object to use when submitting a [partFragment](#) element to a Microsoft Biztalk 2004 Human Workflow Services (HWS) server.

Type

xsd:string

Parent Elements

| Element | Description |
|------------------------------|---|
| partFragment | Defines one substitution group for a specific part of the input Simple Object Access Protocol (SOAP) message. |

Definition

```
<xsd:attribute name="dataObject" type="xsd:string" use="optional" ><
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **dataObject** attribute as it is used in the **partFragment** element:

```
<xsf:partFragment  
  match="ActionSection/Target"  
  replaceWith="/my:myFields/my:Target"  
  dataObject="hwsDataObject3"/>
```


default Attribute

Specifies the name identifier of the view chosen to be the default view.

Type

xsd:string

Remarks

The **default** attribute is an optional attribute of the [views](#) element.

If not specified, the default view is the first **view** element found within the **views** element. This view is loaded when an Microsoft Office InfoPath 2003 is initially opened.

Note In the case of forms that use [ActiveX Data Objects \(ADO\)](#) or a Web service as their primary data source, the default view is set by the **initialView** attribute in the [processing instruction](#) of the form's [XML template file](#). This attribute cannot be changed while in [design mode](#).

Example

The following is an example of the **default** attribute as it is used in the **views** element:

```
<xsf:views default="View">  
  <xsf:view name="View" caption="View">  
    <xsf:mainpane transform="view1.xsl"></xsf:mainpane>  
    ...  
  </xsf:view>  
</xsf:views>
```

default Attribute (roles Element)

Designates a particular role as the default role.

Type

xsd:string

Parent Elements

| Element | Description |
|-----------------------|----------------|
| roles | Defines roles. |

Definition

```
<xsd:attribute name="default" type="xsd:string" use="required" ></xs
```

Remarks

If a user of a form is not defined in the **membership** element of the roles element or if the role of a user cannot be determined, that user is implicitly assigned to the role specified by the **default** attribute. The **default** attribute is required for the **roles** element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **default** attribute as it is used in the **roles** element:

```
<xsf:roles initiator="A" default="C" hideStatusBarDisplay="yes">  
  <xsf:role name="A" />  
  <xsf:role name="B" />  
  <xsf:role name="C" />
```

```
<xsf:membership>  
  <xsf:getUserNameFromData dataObject="catalog" select="/dfs:my  
  <xsf:userName name="domain\username1" memberOf="A" />  
  <xsf:userName name="domain\username2" memberOf="B" />  
  <xsf:group name="domain\groupname1" memberOf="C" />  
</xsf:membership>  
</xsf:roles>
```

description Attribute

A brief description of the form.

Type

xsd:string

Remarks

The **description** attribute is an optional attribute of the [xDocumentClass](#) element.

The value of the **description** attribute can contain up to 255 characters.

Example

The following is an example of the **description** attribute as it is used in the **xDocumentClass** element:

```
<xsf:xDocumentClass
  xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solut
  name="urn:microsoft-com:myTravelReport"
  author="AuthorName"
  description="Travel Report form for entering travel reports, issues, &
  dataFormSolution="yes"
  solutionVersion = "1.0.0.1"
  productVersion="11.0.5106"
  solutionFormatVersion="1.0.0.0">
  ...
</xsf:xDocumentClass>
```


designMode Attribute (view Element)

Determines whether the [view](#) (*.xsl) file specified in a **view** element can be opened in design mode in Microsoft Office InfoPath 2003.

Type

xsf:xdDesignMode

Parent Elements

| Element | Description |
|----------------------|---|
| view | Contains information about a Microsoft Office InfoPath 2003 view. |

Definition

```
<xsd:attribute name="designMode" type="xsf:xdDesignMode" ></xsd
```

Remarks

You can create a custom view for a [form template](#) by creating an [XSL Transformation \(XSLT\)](#) (*.xsl) file that contains constructs that are not supported in design mode in InfoPath. To ensure that a view can't be opened in design mode (and to prevent data loss), set the **designMode** attribute in the [form definition file \(.xsf\)](#) to "protected". Protected views are shown in all views lists and users can be fill them out like all forms, but they cannot be opened in design mode. If the **designMode** attribute is specified as "normal" or is not specified, the view can be opened in design mode.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **designMode** attribute as it is used in the **view** element:

```
<xsf:view name="View" caption="View" designMode="protected">  
  <xsf:mainpane transform="view1.xsl"/>  
  ...  
</xsf:view>
```


detailKey Attribute

Contains the [XPath](#) of the [field](#) in the detail XML fragment that forms the relationship to the master XML fragment.

Type

xsd:string

Parent Elements

| Element | Description |
|------------------------------|---|
| masterDetail | Defines the XML fragments that form a master and detail relationship in a view's repeating tables or repeating sections . |

Definition

```
<xsd:attribute name="detailKey" type="xsd:string" ></xsd:attribute>
```

Remarks

Use a relative XPath for the **detailKey** attribute.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **detailKey** attribute as it is used in the **masterDetail** element:

```
<xsf:editWith caption="group2"
  xd:autogeneration="template"
  component="xCollection">
  <xsf:masterDetail
    masterViewContext="CTRL1_5"
    master="my:group2"
    masterKey="my:field1"
    detailKey="my:field3">
  </xsf:masterDetail>
</xsf:editWith>
```

disableMenuItem Attribute

Specifies whether the menu item for the submit operation should be disabled.

Type

xsf:xdYesNo

Remarks

The **disableMenuItem** attribute is an optional attribute of the [submit](#) element.

Allowed values are "yes" and "no". The default value is "no".

Example

The following is an example of the **disableMenuItem** attribute as it is used in the **submit** element:

```
<xsf:submit  
  caption="Submit"  
  disableMenuItem="no"  
  onAfterSubmit="KeepOpen"  
  showStatusDialog="yes"  
  showSignatureReminder="yes">  
  <xsf:useScriptHandler/>  
  <xsf:successMessage>Submit was successful.</xsf:successMessage>  
  <xsf:errorMessage>Submit was not successful.</xsf:errorMessage>  
</xsf:submit>
```

enabled Attribute

Specifies whether form merging is enabled for the form.

Type

xsf:xdYesNo

Remarks

The **enabled** attribute is a required attribute of the [importParameters](#) element.

Values include "yes" or "no". The default value is "yes".

Example

The following is an example of the **enabled** attribute as it is used in the **importParameters** element:

```
<xsf:importParameters
  enabled="yes"
  <xsf:importSource
    name=""
    schema="MySchema.xsd"
    transform="schematransform.xslt"/>
</xsf:importParameters>
```


enforceScriptTimeout Attribute

Specifies whether to enable or disable a time-out period for scripts in a form.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|-------------------------|--|
| scripts | Defines the source of all business logic scripts used at the document level in the form. |

Definition

`<xsd:attribute name="enforceScriptTimeout" type="xsd:boolean" use="required"/>`

Remarks

Setting the **enforceScriptTimeout** attribute to "no" in the [form definition file \(.xsf\)](#) disables the time-out period for scripts running in the form. The default value is "yes".

For code running in a task pane, there is a separate time-out period provided by Microsoft Internet Explorer. This time-out period is not affected by the value of the **enforceScriptTimeout** attribute.

A time-out period cannot be specified for managed code in InfoPath forms.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **enforceScriptTimeout** attribute as it is used in the **scripts** element:

```
<xsf:scripts language="jscript" enforceScriptTimeout="no">  
  <xsf:script src="internal.js" />  
  <xsf:script src="script.js" />  
</xsf:scripts>
```


expression Attribute

An [XPath](#) expression (relative to the [expressionContext](#) attribute, if specified) that must be evaluated to validate the XML [Document Object Model \(DOM\)](#) node specified in the [match](#) attribute.

Type

xsd:string

Remarks

The **expression** attribute is a required attribute of the [errorCondition](#) element.

If the specified expression evaluates to True, it is considered to be an error condition and the specified error message is displayed.

Example

The following is an example of the **expression** attribute as it is used in the **errorCondition** element:

```
<xsf:customValidation>
  <xsf:errorCondition
    match="/exp:expenseReport"
    expressionContext="exp:reportDate"
    expression="msxsl:string-compare(., ../exp:startDate) < 0 and ../ex
    showErrorOn=".">
    <xsf:errorMessage
      type="modeless"
      shortMessage="The report date occurs before the end of the expe
      The report date occurs before the end of the expense period. Veri:
    </xsf:errorMessage>
  </xsf:errorCondition>
</xsf:customValidation>
```


expression Attribute (assignmentAction Element)

Contains an [XPath](#) expression to populate the value of the [targetField](#) attribute.

Type

xsd:string

Parent Elements

| Element | Description |
|----------------------------------|---|
| assignmentAction | Defines an action to set the value of a field . |

Definition

```
<xsd:attribute name="expression" type="xsd:string" use="required" ><
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **expression** attribute as it is used in the **assignmentAction** element:

```
<xf:rule caption="Receipts" condition="my:expense > 75">
  <xf:dialogBoxMessageAction>Don't forget your receipts!</xf:dialogBoxMessageAction>
  <xf:switchViewAction view="Approval View"/>
  <xf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression="sum(my:expenses/my:expense) * my:taxRate"/>
  <xf:queryAction adapter="Exchange Rates"/>
  <xf:submitAction adapter="Exchange Rates"/>
  <xf:openNewDocumentAction solutionURI="uri:microsoft-Expense1">
  <xf:closeDocumentAction promptToSaveChanges="yes"/>
  <xf:dialogBoxExpressionAction>my:group/my:field1</xf:dialogBoxExpressionAction>
  <xf:exitRuleSet/>
</xf:rule>
```


expression Attribute (calculatedField Element)

Contains the [formula](#), in the form of an [XPath](#) expression, to be evaluated. The result is stored in the [target](#) location.

Type

xsd:string

Parent Elements

| Element | Description |
|---------------------------------|--|
| calculatedField | Defines an individual calculation, including the formula , when the calculation is to be performed, and where the result will be stored. |

Definition

```
<xsd:attribute name="expression" type="xsd:string" use="required" ><
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **expression** attribute as it is used in the **calculatedField** element:

```
<xsf:calculations>  
  <xsf:calculatedField  
    target="/my:myFields/my:average"  
    expression="xdMath:Avg(..my:expenses/my:expense/my:amount)"  
    refresh="onChange" />  
</xsf:calculations>
```


expressionContext Attribute

Specifies the XML [Document Object Model \(DOM\)](#) node on which the expression specified in the [expression](#) attribute is rooted.

Type

xsd:string

Remarks

The **expressionContext** attribute is an optional attribute of the [errorCondition](#) element.

It contains a relative [XPath](#) expression that identifies the XML DOM node (within the context of the matched XML DOM node) on which the expression is rooted and therefore should be evaluated. The default value is "." This is the same as the matched XML DOM node.

Example

The following is an example of the **expressionContext** attribute as it is used in the **errorCondition** element:

```
<xsf:customValidation>
  <xsf:errorCondition
    match="/exp:expenseReport"
    expressionContext="exp:reportDate"
    expression="msxsl:string-compare(., ../exp:startDate) < 0 and ../exp:
    showErrorOn=".">
  <xsf:errorMessage
    type="modeless"
    shortMessage="The report date occurs before the end of the expe
    The report date occurs before the end of the expense period. Veri:
  </xsf:errorMessage>
</xsf:errorCondition>
</xsf:customValidation>
```

feature Attribute

Sets whether the AutoRecover feature is enabled.

Type

xsf:xdEnabledDisabled

Parent Elements

| Element | Description |
|------------------------------|--|
| autoRecovery | Specifies whether the form will save AutoRecover information and whether the AutoRecover setting can be changed by the user. |

Definition

```
<xsd:attribute name="feature" type="xsd:boolean" use="optional"/>
```

Remarks

Setting this attribute to a value other than "enabled" or "disabled" will result in an error message when you attempt to open the form.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **feature** attribute as it is used in the **autoRecovery** element:

```
<xsf:autoRecovery feature="disabled"/>
```


field Attribute

Specifies a relative [XPath](#) expression from the XML [Document Object Model \(DOM\)](#) node specified by the [item](#) attribute of the [xmlToEdit](#) element.

Type

xsd:string

Remarks

The **field** attribute is an optional attribute of the [editWith](#) element.

The **field** attribute refers to the XML DOM node in the form's underlying XML document ,which is to be displayed as an editable field for an xText list editing component. The default value is '.', which corresponds to editing the text content of the XML DOM node specified by the **item** attribute of the **xmlToEdit** element.

Example

The following is an example of the **field** attribute as it is used in the **editWith** element:

```
<xsf:editing>
  <xsf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xsf:editWith
      autoComplete="no"
      field="@artist"
      component="xTextList">
    </xsf:editWith>
  </xsf:xmlToEdit>
</xsf:editing>
```


fileUrl Attribute

Contains the [Uniform Resource Locator \(URL\)](#) of the .xml file.

Type

xsd:anyURI

Remarks

The **fileUrl** attribute is a required attribute of the [xmlFileAdapter](#) element.

Example

The following is an example of the **fileUrl** attribute as it is used in the **xmlFileAdapter** element:

```
<xsf:query>  
  <xsf:xmlFileAdapter fileUrl="currencies.xml" />  
</xsf:query>
```


filter Attribute

Specifies the [XPath](#) expression of the structured XML subtree when submitting a subset of the [XML data](#).

Type

xsd:string

Parent Elements

| Element | Description |
|------------------------------|---|
| partFragment | Defines one substitution group for a specific part of the input Simple Object Access Protocol (SOAP) message. |

Definition

```
<xsd:attribute name="filter" type="xsd:string" use="optional" ></xsd:
```

Remarks

Use the **filter** attribute when submitting the selected field or group as a structured XML subtree, including the tags. Can be omitted when submitting the content of the field or group.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **filter** attribute as it is used in the **partFragment** element:

```
<xsf:partFragment match="/dfs:myFields/dfs:dataFields/s0:IsPrime/s0  
replaceWith=/dfs:myFields/dfs:dataFields/s0:IsPrime"  
filter="."  
sendAsString="yes"/>
```


filterDependency Attribute

Specifies automatic reapplication of a filter when filter fields change.

Type

xsd:string

Parent Elements

| Element | Description |
|--------------------------|---|
| editWith | Specifies an instance of an editing component, and provides the corresponding parameters to determine its exact behavior. |

Definition

```
<xsd:attribute name="filterDependency" type="xsd:string" use="optio
```

Remarks

The **filterDependency** attribute in the [form definition file \(.xsf\)](#) is always present for filters that are specified on repeating tables and repeating sections created in the designer. Even though there is no way to toggle **filterDependency** on and off in [design mode](#) in InfoPath, you can turn off automatic refreshing of filters by removing the **filterDependency** attribute from the **editWith** element in the form definition file.

When the **filterDependency** attribute is present, the filter is reapplied when the **onAfterChange** event of a node is raised.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **filterDependency** attribute as it is used in the **editWith** element:

```
<editWith component="xCollection" widgetIcon="filter|standard" useI  
  filterDependency="xpath1 | xpath2 | xpath3">
```


followingSiblings Attribute

Specifies a relative [XPath](#) expression from the parent node that specifies the XML [Document Object Model \(DOM\)](#) nodes prior to which the insertion of the XML fragment should occur.

Type

xsd:string

Remarks

The **followingSiblings** attribute is an optional attribute of the [chooseFragment](#) element.

The insertion will be as a child of the parent node, but it will be prior to any nodes found that satisfy the XPath expression specified by the **followingSiblings** attribute. If no nodes are found, the insertion acts as an append.

The **followingSiblings** attribute is only used during an insert when the current context is not in an item. The behavior is to append to the content of the parent node, unless the **followingSiblings** attribute is specified, in which case the insertion is still within the content of the parent, but prior to any **followingSiblings** nodes.

Example

The following is an example of the **followingSiblings** attribute as it is used in the **chooseFragment** element:

```
<xf:editing>
  <xf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xf:editWith caption="CD"
      xd:autogeneration="template"
      component="xCollection">
      <xf:fragmentToInsert>
        <xf:chooseFragment parent="CDCollection"
          followingSiblings=".">
          <CD>
            <Title></Title>
            <Artist></Artist>
            <Tracks>
              <Track></Track>
              <Track></Track>
            </Tracks>
          </CD>
        </xf:chooseFragment>
      </xf:fragmentToInsert>
    </xf:editWith>
  </xf:xmlToEdit>
</xf:editing>
```


footer Attribute

Specifies the footer text when printing a view.

Type

xsd:string

Remarks

The **footer** attribute is an optional attribute of the [printSettings](#) element. The **printSettings** element also provides a [header](#) attribute.

The **footer** attribute cannot be greater than 255 characters in length.

When using a header or footer in a form, there are a number of variables that can be used to display information such as page numbers, dates, and times; or to align the text that the header or footer contains.

▶ [Header and footer variables](#)

Example

The following is an example of the **footer** attribute as it is used in the **printSettings** element:

```
<xsf:view name="View" caption="View">
  <xsf:printSettings
    header="Header text goes here."
    footer="Footer text goes here."
    orientation="portrait"
    marginUnitsType="in"
    topMargin="1"
    leftMargin="2"
    rightMargin="2"
    bottomMargin="1"
  </xsf:printSettings>
  ...
</xsf:view>
```


fullyEditableNamespace Attribute

Identifies the [namespace](#) of an [XML Schema](#) in the [form template](#) that can be entirely modified in Microsoft Office InfoPath 2003 [design mode](#).

Type

xsd:anyURI

Remarks

The **fullyEditableNamespace** attribute is an optional attribute of the [solutionProperties](#) element.

Example

The following is an example of the **fullyEditableNamespace** attribute as it is used in the **solutionProperties** element:

```
<xsf: applicationParameters application="InfoPath Design Mode">
  <xsf: solutionProperties
    allowCustomization="no"
    lastOpenView="view1"
    scriptLanguage="JScript"
    automaticallyCreateNodes="no"
    lastVersionNeedingTransform="1.1.0.10"
    fullyEditableNamespace="urn:namespace1:mynames"/>
</xsf:applicationParameters>
```


handlerObject Attribute

Identifies the unique name of the [event handler](#) in the scripting code.

Type

xsd:string

Remarks

The **handlerObject** attribute is a required attribute of the [domEventHandler](#) element.

The referenced script must use this name to define event handlers for the specified XML [Document Object Model \(DOM\)](#) node. For example, script may contain functions such as `TravelExpenses::OnValidate` and `TravelExpenses::OnAfterChange` that are called whenever the specified events occur at the matching XML DOM node.

Example

The following is an example of the **handlerObject** as it is used in the **domEventHandler** element:

```
<xsf:domEventHandlers>
  <xsf:domEventHandler
    match="TravelReport/Expenses"
    handlerObject="TravelExpenses"/>
</xsf:domEventHandlers>
```


header Attribute

Specifies the header text when printing a view.

Type

xsd:string

Remarks

The **header** attribute is an optional attribute of the [printSettings](#) element. The **printSettings** element also provides a [footer](#) attribute.

The **header** attribute cannot be greater than 255 characters in length.

When using a header or footer in a form, there are a number of variables that can be used to display information such as page numbers, dates, and times; or to align the text that the header or footer contains.

▶ [Header and footer variables](#)

Example

The following is an example of the **header** attribute as it is used in the **printSettings** element:

```
<xsf:view name="View" caption="View">
  <xsf:printSettings
    header="Header text goes here."
    footer="Footer text goes here."
    orientation="portrait"
    marginUnitsType="in"
    topMargin="1"
    leftMargin="2"
    rightMargin="2"
    bottomMargin="1"
  </xsf:printSettings>
  ...
</xsf:view>
```


hideStatusBarDisplay attribute

Specifies whether the current role is displayed in the status bar.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|-----------------------|----------------|
| roles | Defines roles. |

Definition

`<xsd:attribute name="hideStatusBarDisplay" type="xsd:boolean" use="optional" />`

Remarks

The name of the role for the current user is displayed in the status bar. If the role of a user is changed programmatically, the status bar is updated.

The value of the **hideStatusBarDisplay** attribute is "no" by default. If the attribute is not included in the roles element, therefore, the role of the current user is displayed in the status bar. The **hideStatusBarDisplay** attribute must be included in the **roles** element and its value must be set to "yes" to prevent the role for the current user from being displayed in the status bar.

There is no mechanism in the user interface of InfoPath for changing the **hideStatusBarDisplay** attribute. To add this attribute to the **roles** element or to change its value, the [form definition file \(.xsf\)](#) of a form must be edited manually.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **hideStatusBarDisplay** attribute as it is used in the **roles** element:

```
<xsf:roles initiator="A" default="C" hideStatusBarDisplay="yes">
  <xsf:role name="A" />
  <xsf:role name="B" />
  <xsf:role name="C" />

  <xsf:membership>
    <xsf:getUserNameFromData dataObject="catalog" select="/dfs:my
    <xsf:userName name="domain\username1" memberOf="A" />
    <xsf:userName name="domain\username2" memberOf="B" />
    <xsf:group name="domain\groupname1" memberOf="C" />
  </xsf:membership>
</xsf:roles>
```


href Attribute (initialXmlDocument Element)

Specifies the name of the [XML template file](#) to be used when a user clicks **Fill Out a Form** on the **File** menu.

Type

xsf:xdFileName

Remarks

The **href** attribute is a required attribute of the [initialXmlDocument](#) element.

Example

The following is an example of the **href** attribute as it is used in the **initialXmlDocument** element:

```
<xsf:fileNew>  
  <xsf:initialXmlDocument  
    caption="Travel Report"  
    href="TravelReportTemplate.xml"/>  
</xsf:fileNew>
```


href Attribute (taskpane Element)

Specifies the relative or absolute [Uniform Resource Locator \(URL\)](#) to an .html file.

Type

xsd:string

Remarks

The **href** attribute is a required attribute of the [taskpane](#) element.

The .html file is loaded into an instance of the task pane when a form is opened. The .html file can include business logic scripting code and can access appropriate Microsoft Office InfoPath 2003 object model members.

Example

The following is an example of the **href** attribute as it is used in the **taskpane** element:

```
<xsf:xDocumentClass>
  ...
  <xsf:taskpane
    caption="My Custom Task Pane"
    href="MyCustomTaskPane.htm"/>
  ...
</xsf:xDocumentClass>
```


href Attribute (useHttpHandler Element)

Specifies the [Uniform Resource Locator \(URL\)](#) to which the form should be submitted.

Type

xsd:anyURI

Remarks

The **href** attribute is a required attribute of the [useHttpHandler](#) element.

Example

The following is an example of the **href** attribute as it is used by the **useHttpHandler** element:

```
<xsf:submit
  caption="Submit"
  disableMenuItem="no"
  onAfterSubmit="KeepOpen"
  showStatusDialog="yes"
  showSignatureReminder="yes">
  <xsf:useHttpHandler>
    href="http://MyServer/InfoPathScripts/MyScript.asp"
    method="POST"
  </xsf:useHttpHandler>
  <xsf:successMessage>Submit was successful.</xsf:successMessage>
  <xsf:errorMessage>Submit was not successful.</xsf:errorMessage>
</xsf:submit>
```


icon Attribute

Provides a [Uniform Resource Locator \(URL\)](#) to a bitmap (.bmp) or graphics interchange format (.gif) file, which is used for the button or menu item.

Type

xsd:string

Remarks

The **icon** attribute is an optional attribute of the [button](#) element.

If an icon is not specified, the caption alone will be used. If both caption and icon are specified, both will be displayed. Alternatively, the value can also be an ID, allowing access to internal system icons. If the value is an integer, it will be interpreted as an ID.

Example

The following is an example of the **icon** attribute as it is used in the **button** element:

```
<xsf:menuArea name="msoInsertMenu">
  <xsf:menu caption="&Section">
    <xsf:button action="xCollection::insert" xmlToEdit="CD_10"
      caption="CD" icon="cd.bmp" showIf="always"></xsf:button>
    <xsf:button action="xCollection::insert" xmlToEdit="Track_14"
      caption="Track" icon="track.bmp" showIf="always"></xsf:button>
    <xsf:button action="xOptional::insert" xmlToEdit="Label_16"
      caption="Label" icon="label.bmp"></xsf:button>
  </xsf:menu>
</xsf:menuArea>
```


infopathGroup Attribute

Contains the name of the [group](#) under which all fields in a SharePoint list or library will be stored.

Type

xsd:string

Parent Elements

| Element | Description |
|---------------------------------------|--|
| sharepointListAdapter | Contains the data adapter information to query a SharePoint list or library. |

Definition

```
<xsd:attribute name="infopathGroup" type="xsd:string" use="required"/>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **infopathGroup** attribute as it is used in the **sharepointListAdapter** element:

```
<xsf:sharepointListAdapter
  name="Status Report library"
  siteUrl="http://xyzco/reports/"
  sharepointGuid="{ABD2E239-0EE7-48F4-B506-C38A1728E195}"
  infopathGroup="XYZReportsLibrary"
  queryAllowed="yes">
  <xsf:field
    sharepointName="File_x0020_Type"
    infopathName="Type"></xsf:field>
  <xsf:field
    sharepointName="xd__x007b_D00F1DBD_..."
    infopathName="Title_1"></xsf:field>
</xsf:sharepointListAdapter>
```

infopathName Attribute

An attribute of the [field](#) element. Contains the corresponding InfoPath field name for the [sharepointName](#) attribute.

Type

xsd:string

Parent Elements

| Element | Description |
|-----------------------|---|
| field | Contains field mapping information for each field in a SharePoint list or library and the corresponding name used in Microsoft Office InfoPath. |

Definition

```
<xsd:attribute name="infopathName" type="xsd:string" use="required
```

Remarks

The **infopathName** appears as the field names in Microsoft Office InfoPath 2003 SP1 for the fields returned from the SharePoint list or library.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **infopathName** attribute as it is used in the **field** element:

```
<xsf:field  
  sharepointName="xd__x007b_D00F1DBD_..."  
  infopathName="Title_1" isLookup="no">  
</xsf:field>
```

initiator Attribute

Designates a particular role as the initiator role.

Type

xsd:string

Parent Elements

| Element | Description |
|-----------------------|----------------|
| roles | Defines roles. |

Definition

```
<xsd:attribute name="initiator" type="xsd:string" use="optional" ></x:
```

Remarks

If a user creates a new InfoPath document from a form template, that user is assigned to the role that has been designated as the initiator role. If no role is designated as the initiator role in the form definition file of a form (that is, if the **initiator** attribute is not included for the **roles** element), users who create forms from the accompanying form template retain the roles to which they are assigned in the **membership** element of the **roles** element. After a form has been saved, closed, and reopened, InfoPath no longer checks for the **initiator** attribute.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **initiator** attribute as it is used in the **roles** element:

```
<xsf:roles initiator="A" default="C" hideStatusBarDisplay="yes">
  <xsf:role name="A" />
  <xsf:role name="B" />
  <xsf:role name="C" />

  <xsf:membership>
    <xsf:getUserNameFromData dataObject="catalog" select="/dfs:my
    <xsf:userName name="domain\username1" memberOf="A" />
    <xsf:userName name="domain\username2" memberOf="B" />
    <xsf:group name="domain\groupname1" memberOf="C" />
  </xsf:membership>
</xsf:roles>
```

initOnLoad Attribute

Specifies whether the data object should be initialized on document load.

Type

xsf:xdYesNo

Remarks

The **initOnLoad** attribute is an optional attribute of the [dataObject](#) element.

Allowed values are "yes" and "no". The default value is "no".

Example

The following is an example of the **initOnLoad** attribute as it is used in the **dataObject** element:

```
<xsf:dataObjects>
  <xsf:dataObject
    name="EmployeeNames"
    schema="EmployeeNames.xsd"
    initOnLoad="yes">
    <xsf:query>
      <xsf:adoAdapter
        connectionString="Provider=Microsoft.Jet.OLEDB.4.0;
          Password=&quot;&quot;;User ID=Admin;
          Data Source=infwind.mdb;Mode=Share Deny None;
          Extended Properties=&quot;&quot;;..."
        commandText="select [EmployeeID],[LastName],[FirstName]
          from [Employees] as [Employees]"
        queryAllowed="yes"
        submitAllowed="yes">
      </xsf:adoAdapter>
    </xsf:query>
  </xsf:dataObject>
</xsf:dataObjects>
```


innerFragment Attribute

Specifies a relative [XPath](#) expression from the parent node to the smallest fragment to be inserted.

Type

xsd:string

Parent Elements

| Element | Description |
|--------------------------------|----------------------------|
| chooseFragment | Specifies an XML fragment. |

Definition

```
<xsd:attribute name="innerFragment" type="xsd:string" use="optional
```

Remarks

The **innerFragment** attribute is used to identify the position of the current data context within the fragment, and to choose the right subtree to insert. This reduces the need to define multiple fragments in the [form definition file \(.xsf\)](#) for different data contexts.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **innerFragment** attribute as it is used in the **chooseFragment** element:

```
<xsf:xmlToEdit name="expense_1" item="/my:myFields/my:expenses
<xsf:editWith caption="expense" component="xCollection">
  <xsf:fragmentToInsert>
    <xsf:chooseFragment innerFragment="my:expenses/my:expense" >
      <my:expenses>
        <my:expense/>
      </my:expenses>
    </xsf:chooseFragment>
  </xsf:fragmentToInsert>
</xsf:editWith>
</xsf:xmlToEdit>
```


isEnabled Attribute

Specifies whether the [rule](#) is enabled for the form.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|----------------------|--|
| rule | Defines an action invoked after an event has occurred in the form. |

Definition

```
<xsd:attribute name="isEnabled" type="xsf:xdYesNo" use="optional"
```

Remarks

The default value for the **isEnabled** attribute is "yes".

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **isEnabled** attribute as it is used in the **rule** element:

```
<xsf:ruleSets>
  <xsf:ruleSet name="RuleSet1">
    <xsf:rule caption="Receipts" condition="my:expense > 75">
      <xsf:dialogBoxMessageAction>Don't forget receipts!</xsf:dialogBo
      <xsf:openNewDocumentAction solution="urn:approvalForm"/>
      <xsf:exitRuleSet/>
    </xsf:rule>
    <xsf:rule caption="Always Submit" isEnabled="no">
      <xsf:submitAction adapter="Expense Database"/>
    </xsf:rule>
  </xsf:ruleSet>

  <xsf:ruleSet name="RuleSet2">
    <xsf:rule caption="Look up contact">
      <xsf:queryAction adapter="Contacts"/>
    </xsf:rule>
  </xsf:ruleSet>
</xsf:ruleSets>
```

isLookup Attribute

Specifies whether a field in a SharePoint list is a lookup field.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|-----------------------|---|
| field | Contains field mapping information for each SharePoint field and the corresponding name used in Microsoft Office InfoPath 2003 SP1. |

Definition

`<xsd:attribute name="isLookup" type="xsd:boolean" use="optional" :`

Remarks

The default value for the **isLookup** attribute is "no".

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **isLookup** attribute as it is used in the **field** element:

```
<xsf:field  
  sharepointName="xd__x007b_D00F1DBD_..."  
  infopathName="Title_1" isLookup="no">  
</xsf:field>
```


item Attribute

Specifies an [XPath](#) expression that determines the XML [Document Object Model \(DOM\)](#) nodes to be edited using the editing component defined in the [editWith](#) element.

Type

xsd:string

Remarks

The **item** attribute is a required attribute of the [xmlToEdit](#) element.

Example

The following is an example of the **item** attribute as it is used in the **xmlToEdit** element:

```
<xf:editing>
  <xf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xf:editWith caption="CD"
      xd:autogeneration="template"
      component="xCollection">
      <xf:fragmentToInsert>
        <xf:chooseFragment parent="CDCollection">
          <CD>
            <Title></Title>
            <Artist></Artist>
            <Tracks>
              <Track></Track>
              <Track></Track>
            </Tracks>
          </CD>
        </xf:chooseFragment>
      </xf:fragmentToInsert>
    </xf:editWith>
  </xf:xmlToEdit>
</xf:editing>
```

language Attribute

Defines the script language used in the business logic source files.

Type

xsf:xdScriptLanguage

Remarks

The **language** attribute is a required attribute of the [scripts](#) element.

Allowed values include "vbscript", "jscript", and "javascript". The default value is "jscript".

Example

The following is an example of the **language** attribute as it is used in the **scripts** element:

```
<xf:scripts language="jscript" enforceScriptTimeout="no">  
  <xf:scriptsrc="internal.js" />  
  <xf:scriptsrc="script.js" />  
</xf:scripts>
```


lastOpenView Attribute

Identifies the name of the [view](#) that was last open in Microsoft Office InfoPath 2003 [design mode](#).

Type

xsd:string

Remarks

The **lastOpenView** attribute is an optional attribute of the [solutionProperties](#) element.

The next time the form is opened in InfoPath design mode, this view is automatically displayed. The value for this attribute should be an existing view name in the [form template](#).

Example

The following is an example of the **lastOpenView** attribute as it is used in the **solutionProperties** element:

```
<xsf: applicationParameters application="InfoPath Design Mode">  
  <xsf: solutionProperties  
    allowCustomization="no"  
    lastOpenView="view1"  
    scriptLanguage="JScript"  
    automaticallyCreateNodes="no"  
    lastVersionNeedingTransform="1.0.0.10"  
    fullyEditableNamespace="urn:namespace1:mynames"/>  
</xsf:applicationParameters>
```


lastVersionNeedingTransform Attribute

Identifies, temporarily, the value of the [maxToVersionUpgrade](#) attribute in the [documentVersionUpgrade](#) element for upgrade with an [.xslt](#) file if scripting code is being used for the upgrade.

Type

xsf:xdSolutionVersion

Remarks

The **lastVersionNeedingTransform** attribute is an optional attribute of the [solutionProperties](#) element.

Example

The following is an example of the **lastVersionNeedingTransform** attribute as it is used in the **solutionProperties** element:

```
<xsf: applicationParameters application="InfoPath Design Mode">  
  <xsf: solutionProperties  
    allowCustomization="no"  
    lastOpenView="view1"  
    scriptLanguage="JScript"  
    automaticallyCreateNodes="no"  
    lastVersionNeedingTransform="1.0.0.10"  
    fullyEditableNamespace="urn:namespace1:mynames"/>  
</xsf:applicationParameters>
```

leftMargin Attribute

Specifies the left margin when printing a view.

Type

xsd:string

Remarks

The **leftMargin** attribute is an optional attribute of the [printSettings](#) element.

The **leftMargin** attribute must be greater than or equal to zero.

Example

The following is an example of the **leftMargin** attribute as it is used in the **printSettings** element:

```
<xsf:view name="View" caption="View">
  <xsf:printSettings
    header="Header text goes here."
    footer="Footer text goes here."
    orientation="portrait"
    marginUnitsType="in"
    topMargin="1"
    leftMargin="2"
    rightMargin="2"
    bottomMargin="1"
  </xsf:printSettings>
  ...
</xsf:view>
```


location Attribute

Contains the namespace [Uniform Resource Identifier \(URI\)](#) and location (a [Uniform Resource Locator \(URL\)](#), relative to the [form definition \(.xsf file\)](#)), and delimited by a space, of the [.xsd file](#) defining the [XML Schema](#).

Type

xsd:string

Remarks

The **location** attribute is a required attribute of the [documentSchema](#) element.

Non-namespace-based XML Schemas are listed with just the .xsd file, omitting the namespace declaration and the white space delimiter.

Example

The following is an example of the **location** attribute as it is used in the **documentSchema** element:

```
<xsf:documentSchemas>  
  <xsf:documentSchema  
    location="urn:schema:custom:Namespace customFilename.xsd"  
    rootSchema="yes"/>  
</xsf:documentSchemas>
```

marginUnitsType Attribute

Specifies the margin unit size when printing a view.

Type

xsd:NMTOKEN

Remarks

The **marginUnitsType** attribute is an optional attribute of the [printSettings](#) element.

Values include "in" (inch) and "cm" (centimeter).

Example

The following is an example of the **marginUnitsType** attribute as it is used in the **printSettings** element:

```
<xsf:view name="View" caption="View">
  <xsf:printSettings
    header="Header text goes here."
    footer="Footer text goes here."
    orientation="portrait"
    marginUnitsType="in"
    topMargin="1"
    leftMargin="2"
    rightMargin="2"
    bottomMargin="1"
  </xsf:printSettings>
  ...
</xsf:view>
```


master Attribute

Contains the [XPath](#) of the XML fragment that is bound to a master table or section.

Type

xsd:string

Parent Elements

| Element | Description |
|------------------------------|---|
| masterDetail | Defines the XML fragments that form a master and detail relationship in a view's repeating tables or repeating sections . |

Definition

```
<xsd:attribute name="master" type="xsd:string" ></xsd:attribute>
```

Remarks

Use a relative XPath for the **master** attribute.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **master** attribute as it is used in the **masterDetail** element:

```
<xsf:editWith caption="group2"
  xd:autogeneration="template"
  component="xCollection">
  <xsf:masterDetail
    masterViewContext="CTRL1_5"
    master="my:group2"
    masterKey="my:field1"
    detailKey="my:field3">
  </xsf:masterDetail>
</xsf:editWith>
```


masterKey Attribute

Contains the [XPath](#) of the [field](#) in the master XML fragment that forms the relationship to the detail XML fragment.

Type

xsd:string

Parent Elements

| Element | Description |
|------------------------------|---|
| masterDetail | Defines the XML fragments that form a master and detail relationship in a view's repeating tables or repeating sections . |

Definition

```
<xsd:attribute name="masterKey" type="xsd:string" ></xsd:attribute>
```

Remarks

Use a relative XPath for the **masterKey** attribute.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **masterKey** attribute as it is used in the **masterDetail** element:

```
<xf:editWith caption="group2"
  xd:autogeneration="template"
  component="xCollection">
  <xf:masterDetail
    masterViewContext="CTRL1_5"
    master="my:group2"
    masterKey="my:field1"
    detailKey="my:field3">
  </xf:masterDetail>
</xf:editWith>
```


masterViewContext Attribute

Contains a string that identifies an HTML element in the view.

Type

xsd:string

Parent Elements

| Element | Description |
|------------------------------|---|
| masterDetail | Defines the XML fragments that form a master and detail relationship in a view's repeating tables or repeating sections . |

Definition

```
<xsd:attribute name="masterViewContext" type="xsd:string" ></xsd:a
```

Remarks

The value of the **masterViewContext** attribute specifies an element that has an **xd:CtrlId** attribute with a matching value in the HTML that represents a view of the form. For example, a **masterViewContext** attribute with a value of "myID" corresponds to the element that has a value of "myID" for its **xd:CtrlId** attribute.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **masterViewContext** attribute as it is used in the **masterDetail** element:

```
<xsf:editWith caption="group2"
  xd:autogeneration="template"
  component="xCollection">
  <xsf:masterDetail
    masterViewContext="CTRL1_5"
    master="my:group2"
    masterKey="my:field1"
    detailKey="my:field3">
  </xsf:masterDetail>
</xsf:editWith>
```


match Attribute (domEventHandler Element)

Identifies the XML [Document Object Model \(DOM\)](#) node for which the [event handler](#) is declared.

Type

xsd:string

Remarks

The **match** attribute is a required attribute of the [domEventHandler](#) element.

The value must be a valid [XPath](#) expression that identifies the XML DOM node.

Note The XPath expression cannot contain predicates.

Example

The following is an example of the **match** attribute as it is used in the **domEventHandler** element:

```
<xsf:domEventHandlers>  
  <xsf:domEventHandler  
    match="TravelReport/Expenses"  
    handlerObject="TravelExpenses"/>  
</xsf:domEventHandlers>
```


match Attribute (errorCondition Element)

Identifies the XML [Document Object Model \(DOM\)](#) nodes on which the custom validation is declared.

Type

xsd:string

Remarks

The **match** attribute is a required attribute of the [errorCondition](#) element.

Example

The following is an example of the **match** attribute as it is used in the **errorCondition** element:

```
<xsf:customValidation>
  <xsf:errorCondition
    match="/exp:expenseReport"
    expressionContext="exp:reportDate"
    expression="msxsl:string-compare(., ../exp:startDate) < 0 and ../exp:
    showErrorOn=".">
    <xsf:errorMessage
      type="modeless"
      shortMessage="The report date occurs before the end of the expe
      The report date occurs before the end of the expense period. Veri:
    </xsf:errorMessage>
  </xsf:errorCondition>
</xsf:customValidation>
```


match Attribute (override Element)

Identifies the XML [Document Object Model \(DOM\)](#) node for which the error message override is defined.

Type

xsd:string

Remarks

The **match** attribute is a required attribute of the [override](#) element.

The value of the **match** attribute must be a valid [XPath](#) expression that identifies the XML DOM node.

Example

The following is an example of the **match** attribute as it is used in the **override** element:

```
<xsf:schemaErrorMessages>
  <xsf:override
    match="/sampleData/number">
    <xsf:errorMessage
      shortMessage="Invalid Number.">
      The value entered must be a valid number.
    </xsf:errorMessage>
  </xsf:override>
</xsf:schemaErrorMessages>
```


match Attribute (partFragment Element)

Contains an [XPath](#) expression that identifies the elements and attributes inside the input Simple Object Access Protocol (SOAP) message that are to be replaced at run time.

Type

xsd:string

Remarks

The **match** attribute is a required attribute of the [partFragment](#) element.

Example

The following is an example of the **match** attribute as it is used in the **partFragment** element:

```
<xsf:query>
  <xsf:webServiceAdapter
    wsdlUrl="http://localhost/infopathwebservicessample/infopathwebs
    queryAllowed="yes"
    submitAllowed="no">
    <xsf:operation
      name="getOrders"
      soapAction="http://tempuri.org/getOrders"
      serviceUrl="http://localhost/infopathwebservicessample/infopathv
    <xsf:input
      source="Submit.xml">
      <xsf:partFragment
        match="/dfs:myFields/dfs:dataFields/s0:IsPrime/s0:inValue"
        replaceWith="/dfs:myFields/dfs:dataFields/s0:IsPrime" />
    </xsf:input>
  </xsf:operation>
</xsf:webServiceAdapter>
</xsf:query>
```

maxLength Attribute

Defines the length of the field in number of bytes.

Type

xsd:byte

Remarks

The **maxLength** attribute is an optional attribute of the [field](#) element.

Example

The following is an example of the **maxLength** attribute as it is used in the **field** element:

```
<xsf:listProperties>
  <xsf:fields>
    <xsf:field
      type="xsd:date"
      maxLength="10"
      name="TravelDate"
      columnName="TravelDate"
      required="yes"
      viewable="yes"
      node="TravelReport/Header/travelDate"
      aggregation="first"/>
  </xsf:fields>
</xsf:listProperties>
```

maxLength Attribute (editWith Element)

Specifies the maximum number of characters allowed for plain single line text boxes.

Type

xsd:simpleType

Parent Elements

| Element | Description |
|--------------------------|---|
| editWith | Specifies an instance of an editing component, and provides the corresponding parameters to determine its exact behavior. |

Definition

```
<xsd:attribute name="maxLength" use="optional" >  
  <xsd:simpleType>  
    <xsd:restriction base="xsd:integer">  
      <xsd:minInclusive value="-1" />  
      <xsd:maxInclusive value="9999" />  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:attribute>
```

Remarks

The **maxLength** attribute is only valid for plain-text edit controls that do not allow paragraph breaks. Such controls correspond to **editWith** elements with a **component** attribute set to "xField" and a **type** attribute set to "plain" (or without a type attribute specified).

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **maxLength** attribute as it is used in the **editWith** element:

```
<editWith component="xField" maxLength="100">
```

maxVersionToUpgrade Attribute

Inclusive value for the latest form that needs to be upgraded.

Type

xsf:xdSolutionVersion

Remarks

The **maxVersionToUpgrade** attribute is an optional attribute of the [useTransform](#) element.

Example

The following example shows the **maxVersionToUpgrade** attribute as it is used in the **useTransform** element:

```
<xsf:documentVersionUpgrade>  
  <xsf:useTransform  
    transform="upgrade.xsl"  
    minVersionToUpgrade="0.0.0.0"  
    maxVersionToUpgrade="1.0.0.5"/>  
</xsf:documentVersionUpgrade>
```


memberOf Attribute (getUserNameFromData Element)

Specifies the role to be associated with a user whose user name is returned by an [XPath](#) query of a data source.

Type

xsd:string

Parent Elements

| Element | Description |
|-------------------------------------|--|
| getUserNameFromData | Retrieves a user name by using an XPath query of the data in the main data source or in a secondary data source. |

Definition

```
<xsd:attribute name="memberOf" type="xsd:string" use="required" >
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **memberOf** attribute as it is used in the **getUserUserNameFromData** element:

```
<xsf:roles initiator="A" default="C" hideStatusBarDisplay="yes">
  <xsf:role name="A" />
  <xsf:role name="B" />
  <xsf:role name="C" />

  <xsf:membership>
    <xsf:getUserNameFromData dataObject="catalog" select="/dfs:my
    <xsf:userName name="domain\username1" memberOf="A" />
    <xsf:userName name="domain\username2" memberOf="B" />
    <xsf:group name="domain\groupname1" memberOf="C" />
  </xsf:membership>
</xsf:roles>
```

memberOf Attribute (group Element)

Specifies the role to which a group is assigned.

Type

xsd:string

Parent Elements

| Element | Description |
|-----------------------|--|
| group | Associates a group with a particular role. |

Definition

```
<xsd:attribute name="memberOf" type="xsd:string" use="required" >
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **memberOf** attribute as it is used in the **group** element:

```
<xsf:roles initiator="A" default="C" hideStatusBarDisplay="yes">
  <xsf:role name="A" />
  <xsf:role name="B" />
  <xsf:role name="C" />

  <xsf:membership>
    <xsf:getUserNameFromData dataObject="catalog" select="/dfs:my
    <xsf:userName name="domain\username1" memberOf="A" />
    <xsf:userName name="domain\username2" memberOf="B" />
    <xsf:group name="domain\groupname1" memberOf="C" />
  </xsf:membership>
</xsf:roles>
```

memberOf Attribute (userName Element)

Specifies the role to which a user is assigned.

Type

xsd:string

Parent Elements

| Element | Description |
|--------------------------|---|
| userName | Associates a user with a particular role. |

Definition

```
<xsd:attribute name="memberOf" type="xsd:string" use="required" >
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **memberOf** attribute as it is used in the **userName** element:

```
<xsf:roles initiator="A" default="C" hideStatusBarDisplay="yes">
  <xsf:role name="A" />
  <xsf:role name="B" />
  <xsf:role name="C" />

  <xsf:membership>
    <xsf:getUserNameFromData dataObject="catalog" select="/dfs:my
    <xsf:userName name="domain\username1" memberOf="A" />
    <xsf:userName name="domain\username2" memberOf="B" />
    <xsf:group name="domain\groupname1" memberOf="C" />
  </xsf:membership>
</xsf:roles>
```

method Attribute

Specifies the HTTP method to use for the submit operation.

Type

xsd:NMTOKEN

Remarks

The **method** attribute is a required attribute of the [useHttpHandler](#) element.

The only supported value is "POST".

Example

The following is an example of the **method** attribute as it is used by the **useHttpHandler** element:

```
<xf:submit
  caption="Submit"
  disableMenuItem="no"
  onAfterSubmit="KeepOpen"
  showStatusDialog="yes"
  showSignatureReminder="yes">
  <xf:useHttpHandler>
    href="http://MyServer/InfoPathScripts/MyScript.asp"
    method="POST"
  </xf:useHttpHandler>
  <xf:successMessage>Submit was successful.</xf:successMessage>
  <xf:errorMessage>Submit was not successful.</xf:errorMessage>
</xf:submit>
```


minVersionToUpgrade Attribute

Inclusive value for the oldest form that can be upgraded.

Type

xsf:xdSolutionVersion

Remarks

The **minVersionToUpgrade** attribute is a required attribute of the [useTransform](#) element.

The **minVersionToUpgrade** attribute is used to prevent running the [XSL Transformation \(XSLT\)](#) on forms so different from the current one that the XSLT causes risk of data loss.

Example

The following example shows the **minVersionToUpgrade** attribute as it is used in the **useTransform** element:

```
<xsf:documentVersionUpgrade>  
  <xsf:useTransform  
    transform="upgrade.xsl"  
    minVersionToUpgrade="0.0.0.0"  
    maxVersionToUpgrade="1.0.0.5"/>  
</xsf:documentVersionUpgrade>
```

mode Attribute

Specifies the signature relationship for the [signedDataBlock](#) element.

Type

`xsf:xdSignatureRelationEnum`

Parent Elements

| Element | Description |
|---------------------------------|---|
| signedDataBlock | Defines a nodeset in the form's underlying XML document to which a digital signature can be applied |

Definition

`<xsd:attribute name="mode" type="xsf:xdSignatureRelationEnum" us`

Remarks

The default **mode** of a **signedDataBlock** element is **single**.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **mode** attribute as it is used in the **signedDataBlock** element:

```
<xf:documentSignatures>
  <xf:signedDataBlock name="main"
    data="my:myfields/my:subtree1 | my:myfields/my:subtree2"
    signatureLocation="my:mifields/sig:signatures/main"
    mode="countersign">
    <xf:message>By pressing the &quot;Sign&quot; button below, I agr
  </xf:signedDataBlock>
</xf:documentSignatures>
```

name Attribute (action Element)

The unique name of the [action](#) as specified by the Microsoft BizTalk 2003 Human Workflow Services (HWS) workflow designer.

Type

xsf:xdHWSname

Parent Elements

| Element | Description |
|------------------------|--|
| action | Contains the information for an individual action. |

Definition

```
<xsd:attribute name="name" type="xsf:xdHWSname" use="required"
```

Remarks

The **name** attribute is a required attribute of the **action** element.

The **name** is used for the **onClick** event of the corresponding button in the **Workflow** task pane.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **name** attribute as it is used in the **action** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval" />
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate" />
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response" />
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response" />
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond" />
  </xsf:allowedTasks>
</xsf:hwsWorkflow>

<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes" queryAllowed="no">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID=
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1" />
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
```

```
        replaceWith="/a:some/b:thing" dataObject="Aux1" />  
    </xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```


name attribute (adoAdapter element)

Contains the name of the [adoAdapter](#) element.

Type

xsf:xdTitle

Parent Elements

| Element | Description |
|----------------------------|---|
| adoAdapter | Defines an ActiveX Data Objects (ADO) data adapter that retrieves data from an ADO data source for the specified data object. |

Definition

```
<xsd:attribute name="name" type="xsf:xdTitle" use="optional" ></xsd
```

Remarks

The **name** attribute is an optional attribute of the **adoAdapter** element.

Example

The following is an example of the **name** attribute as it is used in the **adoAdapter** element:

```
<xsf:query>
  <xsf:adoAdapter name="EmpInformation"
    connectionString="Provider=Microsoft.Jet.OLEDB.4.0;
      Password=&quot;&quot;;User ID=Admin;
      Data Source=infwind.mdb;Mode=Share Deny None;
      Extended Properties=&quot;&quot;;..."
    commandText="select [EmployeeID],[LastName],[FirstName]
      from [Employees] as [Employees]"
    queryAllowed="yes"
    submitAllowed="yes">
  </xsf:adoAdapter>
</xsf:query>
```


name Attribute (button Element)

Used to associate the OnClick [event handler](#) of the button with a scripting function.

Type

xsd:NMTOKEN

Remarks

The **name** attribute is an optional attribute of the [button](#) element, but is required for buttons that use scripting code for their actions.

Example

The following is an example of the **name** attribute as it is used in the **button** element:

```
<xsf:menuArea name="msoViewMenu">
  <xsf:button caption="CD Collection"
    name="SwitchToView0"></xsf:button>
  <xsf:button caption="All Tracks"
    name="SwitchToView1"></xsf:button>
</xsf:menuArea>
```

In the form's internal scripting file, the following event handlers are used for the button actions:

```
function SwitchToView0::OnClick()
{
  XDocument.View.SwitchView("CD Collection");
}
```

```
function SwitchToView1::OnClick()
{
  XDocument.View.SwitchView("All Tracks");
}
```


name Attribute (customCategory Element)

Specifies the name of the custom category.

Type

xsf:xdTitle

Parent Elements

| Element | Description |
|--------------------------------|--|
| customCategory | Specifies the category that the form template appears under in the Fill Out a Form task pane. |

Definition

```
<xsd:attribute name="name" type="xsf:xdTitle" use="required"></xsd
```

Remarks

The **name** attribute is a required attribute of the **customCategory** element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **name** attribute as it is used in the **customCategory** element:

```
<xsf:customCategory name="Reports"/>
```

name Attribute (dataObject Element)

The unique name for the data object.

Type

xsf:xdTitle

Remarks

The **name** attribute is a required attribute of the [dataObject](#) element.

Example

The following is an example of the **name** attribute as it is used in the **dataObject** element:

```
<xsf:dataObjects>
  <xsf:dataObject
    name="EmployeeNames"
    schema="EmployeeNames.xsd"
    initOnLoad="yes">
    <xsf:query>
      <xsf:adoAdapter
        connectionString="Provider=Microsoft.Jet.OLEDB.4.0;
          Password=&quot;&quot;;User ID=Admin;
          Data Source=infwind.mdb;Mode=Share Deny None;
          Extended Properties=&quot;&quot;;..."
        commandText="select [EmployeeID],[LastName],[FirstName]
          from [Employees] as [Employees]"
        queryAllowed="yes"
        submitAllowed="yes">
      </xsf:adoAdapter>
    </xsf:query>
  </xsf:dataObject>
</xsf:dataObjects>
```


name Attribute (davAdapter Element)

The name of a [davAdapter](#), used to invoke the adapter from form code.

Type

xsf:xdTitle

Parent Elements

| Element | Description |
|----------------------------|--|
| davAdapter | Contains information to enable InfoPath files to be submitted to a server that is running Microsoft Windows SharePoint Services or a Web-based Distributed Authoring and Versioning (WebDAV) server. |

Definition

```
<xsd:attribute name="name" type="xsf:xdTitle" use="required" ></xsd
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **name** attribute as it is used in the **davAdapter** element:

```
<xsf:davAdapter name="SubmitToSharePoint" overwriteAllowed="ye  
<xsf:fileName value="my:myFields/my:fileName" valueType="expre  
<xsf:folderURL value="http://some_server/some_doc_lib"/>  
</xsf:davAdapter>
```

name Attribute (emailAdapter Element)

Contains the name of the [emailAdapter](#) element.

Type

xsf:xdTitle

Parent Elements

| Element | Description |
|------------------------------|--|
| emailAdapter | Contains the information to submit an InfoPath file as an attachment to an e-mail message, with a specified set of recipients, a subject, and an introduction. |

Definition

```
<xsd:attribute name="name" type="xsf:xdTitle" use="required" ></xsd
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **name** attribute as it is used in the **emailAdapter** element:

```
<xsf:emailAdapter name="Submit" submitAllowed="yes">  
  <xsf:to value="someone@example.com" valueType="literal"/>  
  <xsf:cc value="my:ccNames" valueType="expression"/>  
  <xsf:bcc value="my:bccNames" valueType="expression"/>  
  <xsf:subject value="My report" valueType="literal"/>  
  <xsf:intro value="See below"/>  
  <xsf:attachmentFileName value="Status Report" valueType="literal"/>  
</xsf:emailAdapter>
```

name Attribute (extension Element)

A unique name identifying the extension being specified.

Type

xsd:NMTOKEN

Remarks

The **name** attribute is a required attribute of the [extension](#) element.

Note Use of the **extension** element is reserved. Microsoft Office InfoPath 2003 ignores any content within the **extension** element.

Example

The following is an example of the **name** attribute as it is used in the **extension** element:

```
<xsf:extensions>
  <xsf:extension
    name="someValue"
    anyAttributesHere="someValue">
    ...open content model here...
  </xsf:extension>
</xsf:extensions>
```


name Attribute (externalView Element)

Contains the name of an [externalView](#) element.

Type

xsf:xdViewName

Parent Elements

| Element | Description |
|------------------------------|---|
| externalView | Defines a view that cannot be edited in InfoPath. |

Definition

```
<xsd:attribute name="name" type="xsf:xdViewName" use="required"
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **name** attribute as it is used in the **externalView** element:

```
<xsf:externalViews>  
  <xsf:externalView name="Sales Doc">  
    <xsf:mainpane transform="myWordView.xsl" />  
  </xsf:externalView>  
</xsf:externalViews>
```

name Attribute (field Element)

Identifies the friendly name of the field to be used on the form list view.

Type

xsf:xdTitle

Remarks

The **name** attribute is a required attribute of the [field](#) element.

Example

The following is an example of the **name** attribute as it is used in the **field** element:

```
<xsf:listProperties>
  <xsf:fields>
    <xsf:field
      type="xsd:date"
      name="TravelDate"
      columnName="TravelDate"
      required="yes"
      viewable="yes"
      node="TravelReport/Header/travelDate"
      aggregation="first"/>
  </xsf:fields>
</xsf:listProperties>
```


name Attribute (file Element)

Specifies the name of the file.

Type

xsf:xdFileName

Remarks

The **name** attribute is a required attribute of the [file](#) element.

This must be a [relative URL](#) from the URL of the [form definition \(.xsf\) file](#). All files specified here are inside the form template, so [absolute URLs](#) are not supported.

Example

The following is an example of the **name** attribute as it is used in the **file** element:

```
<xsf:package>
  <xsf:files>
    <xsf:file name="view_1.xsl">
      <xsf:fileProperties>
        <xsf:property
          name="lang"
          type="string"
          value="1033"/>
      </xsf:fileProperties>
    </xsf:file>
  </xsf:files>
</xsf:package>
```

name Attribute (group Element)

Specifies the name of a group of users to be assigned to a particular role.

Type

xsd:string

Parent Elements

| Element | Description |
|-----------------------|--|
| group | Associates a group with a particular role. |

Definition

```
<xsd:attribute name="name" type="xsd:string" use="required" ></xsd:
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **name** attribute as it is used in the **group** element:

```
<xsf:roles initiator="A" default="C" hideStatusBarDisplay="yes">
  <xsf:role name="A" />
  <xsf:role name="B" />
  <xsf:role name="C" />

  <xsf:membership>
    <xsf:getUserNameFromData dataObject="catalog" select="/dfs:my
    <xsf:userName name="domain\username1" memberOf="A" />
    <xsf:userName name="domain\username2" memberOf="B" />
    <xsf:group name="domain\groupname1" memberOf="C" />
  </xsf:membership>
</xsf:roles>
```

name Attribute (hwsAdapter Element)

Contains the name of the data adapter.

Type

xsf:xdTitle

Parent Elements

| Element | Description |
|----------------------------|---|
| hwsAdapter | Defines the Microsoft Biztalk 2004 HWS (Human Workflow Services) data adapter to start or extend an activity flow, and respond to a task. |

Definition

```
<xsd:attribute name="name" type="xsf:xdTitle" use="required" ></xsd
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **name** attribute as it is used in the **hwsAdapter** element:

```
<hwsAdapter
  name="xsd:title"
  wsdlUrl="xsd:string"
  queryAllowed="xsd:yesNo"
  submitAllowed="xsd:yesNo">
  <hwsOperation type="xsd:string" typeID="xsd:string" serviceUrl="xs
</hwsAdapter>
```


name Attribute (importSource Element)

Identifies the name of the source form as defined in the [processing instruction](#) of that form's underlying XML document.

Type

xsd:string

Remarks

The **name** attribute is a required attribute of the [importSource](#) element.

Example

The following is an example of the **name** attribute as it used in the **importSource** element:

```
<xsf:importParameters
  enabled="yes"
  <xsf:importSource
    name="My Form"
    schema="MySchema.xsd"
    transform="schematransform.xslt"/>
</xsf:importParameters>
```

name Attribute (menuArea Element)

Corresponds to one of the built-in Microsoft Office InfoPath 2003 top-level menus.

Type

xsd:NMTOKEN

Remarks

The **name** attribute is a required attribute of the [menuArea](#) element.

There cannot be more than one menu area with the same name within a given view.

InfoPath has the following named menu areas that correspond to the built-in menu elements that can be customized using the **menuArea** element:

| Name | Description |
|---------------------------------|---|
| msoFileMenu | Menu items that are added to the File menu. |
| msoEditMenu | Menu items that are added to the Edit menu. |
| msoInsertMenu | Menu items that are added to the Insert menu. |
| msoViewMenu | Menu items that are added to the View menu. |
| msoFormatMenu | Menu items that are added to the Format menu. |
| msoToolsMenu | Menu items that are added to the Tools menu. |
| msoTableMenu | Menu items that are added to the Table menu. |
| msoHelpMenu | Menu items that are added to the Help menu. |
| msoStructuralEditingContextMenu | Menu items that are added to the right-click shortcut menu. |

Example

The following is an example of the **name** attribute as it is used in the **menuArea** element:

```
<xsf:menuArea name="msoInsertMenu">
  <xsf:menu caption="&Section">
    <xsf:button action="xCollection::insert" xmlToEdit="CD_10"
      caption="CD" showIf="always"></xsf:button>
    <xsf:button action="xCollection::insert" xmlToEdit="Track_14"
      caption="Track" showIf="always"></xsf:button>
    <xsf:button action="xOptional::insert" xmlToEdit="Label_16"
      caption="Label"></xsf:button>
  </xsf:menu>
</xsf:menuArea>
```

name Attribute (operation Element)

Contains the unique name of the Web service method.

Type

xsd:string

Remarks

The **name** attribute is a required attribute of the [operation](#) element.

Example

The following is an example of the **name** attribute as it is used in the **operation** element:

```
<xsf:query>
  <xsf:webServiceAdapter
    wsdlUrl="http://localhost/infopathwebservicessample/infopathwebs
    queryAllowed="yes"
    submitAllowed="no">
    <xsf:operation
      name="getOrders"
      soapAction="http://tempuri.org/getOrders"
      serviceUrl="http://localhost/infopathwebservicessample/infopathv
      <xsf:input
        source="Submit.xml">
      </xsf:input>
    </xsf:operation>
  </xsf:webServiceAdapter>
</xsf:query>
```

name Attribute (property Element)

Defines the name of the property.

Type

xsd:string

Remarks

The **name** attribute is a required attribute of the [property](#) element. A **fileType** value of "ActiveX-CAB" identifies that the file is a .cab file added by the designer and indicates that the file should be managed by the ActiveX support features of the designer. The **timestamp** property identifies the latest version of the .cab file. The version information is used for automatically updating the file when the developer has a more recent version in the ActiveX ICT/CAB directory.

Example

The following are examples of the **name** attribute as it is used in the **property** element:

```
<xsf:files>
  <xsf:file name="example.cab">
    <xsf:fileProperties>
      <xsf:property name="fileType" type="string" value="ActiveX.<br>
      <xsf:property name="timestamp" type="string" value="xyz" />
    </xsf:fileProperties>
  </xsf:file>
</xsf:files>
</xsf:package>
```

```
<xsf:package>
  <xsf:files>
    <xsf:file name="view_1.xsl">
      <xsf:fileProperties>
        <xsf:property
          name="lang"
          type="string"
          value="1033"/>
      </xsf:fileProperties>
    </xsf:file>
  </xsf:files>
</xsf:package>
```

name Attribute (role Element)

Specifies a string that can be used to identify a particular role.

Type

xsf:xdRoleName

Parent Elements

| Element | Description |
|----------------------|---------------|
| role | Defines role. |

Definition

`<xsd:attribute name="name" type="xsf:xdRoleName" use="required" :`

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **name** attribute as it is used in the **role** element:

```
<xsf:roles initiator="A" default="C" hideStatusBarDisplay="yes">
  <xsf:role name="A" />
  <xsf:role name="B" />
  <xsf:role name="C" />

  <xsf:membership>
    <xsf:getUserNameFromData dataObject="catalog" select="/dfs:my
    <xsf:userName name="domain\username1" memberOf="A" />
    <xsf:userName name="domain\username2" memberOf="B" />
    <xsf:group name="domain\groupname1" memberOf="C" />
  </xsf:membership>
</xsf:roles>
```

name Attribute (ruleSet Element)

Contains the name of the [ruleSet](#) element.

Type

xsd:string

Parent Elements

| Element | Description |
|-------------------------|---|
| ruleSet | Contains one or more rule elements. |

Definition

```
<xsd:attribute name="name" type="xsd:string" use="required" ></xsd:
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **name** attribute as it is used in the **ruleSet** element:

```
<xf:ruleSets>
  <xf:ruleSet name="RuleSet1">
    <xf:rule caption="Receipts" condition="my:expense > 75">
      <xf:dialogBoxMessageAction>Don't forget receipts!</xf:dialogBo
      <xf:openNewDocumentAction solution="urn:approvalForm"/>
      <xf:exitRuleSet/>
    </xf:rule>
    <xf:rule caption="Always Submit" isEnabled="no">
      <xf:submitAction adapter="Expense Database"/>
    </xf:rule>
  </xf:ruleSet>

  <xf:ruleSet name="RuleSet2">
    <xf:rule caption="Look up contact">
      <xf:queryAction adapter="Contacts"/>
    </xf:rule>
  </xf:ruleSet>
</xf:ruleSets>
```


name Attribute (sharepointListAdapter Element)

Contains the name of the [sharepointListAdapter](#) element; also used as the [secondary data source](#) name.

Type

xsf:xdTitle

Parent Elements

| Element | Description |
|---------------------------------------|--|
| sharepointListAdapter | Contains the data adapter information to query a SharePoint list or library. |

Definition

```
<xsd:attribute name="name" type="xsf:xdTitle" use="required"></xsd
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **name** attribute as it is used in the **sharepointListAdapter** element:

```
<xsf:sharepointListAdapter
  name="Status Report library"
  siteUrl="http://www.contoso.com/reports/"
  sharepointGuid="{ABD2E239-0EE7-48F4-B506-C38A1728E195}"
  infopathGroup="ContosoReportsLibrary"
  queryAllowed="yes">
  <xsf:field
    sharepointName="File_x0020_Type"
    infopathName="Type"></xsf:field>
  <xsf:field
    sharepointName="xd__x007b_D00F1DBD_..."
    infopathName="Title_1"></xsf:field>
</xsf:sharepointListAdapter>
```

name Attribute (signedDataBlock Element)

Contains the name of the [signedDataBlock](#) element.

Type

xsf:xdSignedDataBlockName

Parent Elements

| Element | Description |
|---------------------------------|--|
| signedDataBlock | Defines a nodeset in the form's underlying XML document to which a digital signature can be applied. |

Definition

`<xsd:attribute name="name" type="xsf:xdSignedDataBlockName" use`

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **name** attribute as it is used in the **signedDataBlock** element:

```
<xf:documentSignatures>
  <xf:signedDataBlock name="main"
    data="my:myfields/my:subtree1 | my:myfields/my:subtree2"
    signatureLocation="my:mifields/sig:signatures/main"
    mode="countersign">
    <xf:message>By clicking the "Sign" button below, I agree
  </xf:signedDataBlock>
</xf:documentSignatures>
```


name Attribute (task Element)

The unique name of the task as specified by the Microsoft Biztalk Server 2004 HWS (Human Workflow Services) workflow designer.

Type

xsf:xdHWSname

Parent Elements

| Element | Description |
|----------------------|---|
| task | The HWS task information enabled for the form |

Definition

```
<xsd:attribute name="name" type="xsf:xdHWSname" use="required"
```

Remarks

The **name** attribute is used for the **onClick** event of the button in the **Workflow** task pane.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **name** attribute as it is used in the **task** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval" />
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate" />
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response" />
    <xsf:task name="getVPAApproval" taskTypeID="436"
      caption="Send Response" />
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond" />
  </xsf:allowedTasks>
</xsf:hwsWorkflow>

<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes" queryAllowed="no">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID=
    serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1" />
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
```

```
        replaceWith="/a:some/b:thing" dataObject="Aux1" />  
    </xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```

name Attribute (toolbar Element)

Identifies the corresponding toolbar within different views.

Type

xsf:xdTitle

Remarks

The **name** attribute is a required attribute of the [toolbar](#) element.
Must be unique within a given solution.

Example

The following is an example of the **name** attribute as it is used in the **toolbar** element:

```
<xsf:toolbar caption="CD Collection Toolbar"
  name="CD Collection Toolbar">
  <xsf:button action="xCollection::insert" xmlToEdit="CD_10"
    caption="New CD" showIf="always"></xsf:button>
  <xsf:button action="xCollection::insert" xmlToEdit="Track_14"
    caption="New Track" showIf="always"></xsf:button>
  <xsf:button action="xOptional::insert" xmlToEdit="Label_16"
    caption="New Label" showIf="always"></xsf:button>
  <xsf:menu caption="Remove">
    <xsf:button action="xCollection::remove" xmlToEdit="CD_10"
      caption="CD" showIf="always"></xsf:button>
    <xsf:button action="xCollection::remove" xmlToEdit="Track_14"
      caption="Track" showIf="always"></xsf:button>
    <xsf:button action="xOptional::remove" xmlToEdit="Label_16"
      caption="Label" showIf="always"></xsf:button>
  </xsf:menu>
</xsf:toolbar>
```

name Attribute (userName Element)

Specifies the name of a user to be assigned to a particular role.

Type

xsd:string

Parent Elements

| Element | Description |
|--------------------------|---|
| userName | Associates a user with a particular role. |

Definition

```
<xsd:attribute name="name" type="xsd:string" use="required"></xsd:
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **name** attribute as it is used in the **userName** element:

```
<xsf:roles initiator="A" default="C" hideStatusBarDisplay="yes">
  <xsf:role name="A" />
  <xsf:role name="B" />
  <xsf:role name="C" />

  <xsf:membership>
    <xsf:getUserNameFromData dataObject="catalog" select="/dfs:my
    <xsf:userName name="domain\username1" memberOf="A" />
    <xsf:userName name="domain\username2" memberOf="B" />
    <xsf:group name="domain\username3" memberOf="C" />
  </xsf:membership>
</xsf:roles>
```

name Attribute (view Element)

Identifies the view for object model calls when switching views, and for specifying the default view.

Type

xsf:xdViewName

Remarks

The **name** attribute is a required attribute of the [view](#) element.

Example

The following is an example of the **name** attribute as it is used in the **view** element:

```
<xsf:views default="View">  
  <xsf:view name="View" caption="View">  
    <xsf:mainpane transform="view1.xsl"></xsf:mainpane>  
    ...  
  </xsf:view>  
</xsf:views>
```


name Attribute (webServiceAdapter Element)

Contains the name of the [webServiceAdapter](#) element.

Type

xsf:xdTitle

Parent Elements

| Element | Description |
|-----------------------------------|--|
| WebServiceAdapter | Defines a Web service data adapter that retrieves data from a Web service for the specified data object. |

Definition

```
<xsd:attribute name="name" type="xsf:xdTitle" use="optional"></xsd
```

Remarks

The **name** attribute is an optional attribute of the **webServiceAdapter** element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **name** attribute as it is used in the **webServiceAdapter** element:

```
<xsf:query>
  <xsf:webServiceAdapter
    wsdlUrl="http://localhost/infopathwebservicessample/infopathwebs
    queryAllowed="yes"
    submitAllowed="no"
    useDataSet="no">
    <xsf:operation
      name="getOrders"
      soapAction="http://www.contoso.com/getOrders"
      serviceUrl="http://localhost/infopathwebservicessample/infopathv
    <xsf:input
      source="Submit.xml">
    </xsf:input>
    </xsf:operation>
  </xsf:webServiceAdapter>
</xsf:query>
```


name Attribute (xDocumentClass Element)

Provides a unique, [Uniform Resource Name \(URN\)](#)-based name for the form that the [form definition \(.xsf\) file](#) defines.

Type

xsd:string

Remarks

The **name** attribute is an optional attribute of the [xDocumentClass](#) element.

If this attribute is missing, the form is named from the [Uniform Resource Locator \(URL\)](#) or form definition file name, which can be obtained from the [processing instructions](#) of the form's underlying XML document.

Example

The following is an example of the **name** attribute as it is used in the **xDocumentClass** element:

```
<xsf:xDocumentClass
  xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solut
  name="urn:microsoft-com:myTravelReport"
  author="AuthorName"
  description="Travel Report form for entering travel reports, issues, e
  dataFormSolution="yes"
  solutionVersion = "1.0.0.1"
  productVersion="11.0.5106"
  solutionFormatVersion="1.0.0.0">
  ...
</xsf:xDocumentClass>
```

name Attribute (xmlFileAdapter Element)

Contains the name of the [xmlFileAdapter](#) element.

Type

xsf:xdTitle

Parent Elements

| Element | Description |
|--------------------------------|--|
| xmlFileAdapter | Defines an .xml file data adapter that retrieves data from an .xml file for the specified data object. |

Definition

```
<xsd:attribute name="name" type="xsf:xdTitle" use="optional" ></xsd
```

Remarks

The **name** attribute is an optional attribute of the **xmlFileAdapter** element.

Example

The following is an example of the **name** attribute as it is used in the **xmlFileAdapter** element:

```
<xsf:query>  
  <xsf:xmlFileAdapter name="CurrencyInfo" fileUrl="currencies.xml"  
</xsf:query>
```

name Attribute (xmlToEdit Element)

Used in the [xmlToEdit](#) attribute of the [button](#) element to associate actions of the associated editing component to buttons defined in menus and toolbars.

Type

xsd:NMTOKEN

Remarks

The **name** attribute is a required attribute of the [xmlToEdit](#) element.

There should be no more than one **xmlToEdit** element with the same name in a given view.

Example

The following is an example of the **name** attribute as it is used in the **xmlToEdit** element:

```
<xf:editing>
  <xf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xf:editWith caption="CD"
      xd:autogeneration="template"
      component="xCollection">
      <xf:fragmentToInsert>
        <xf:chooseFragment parent="CDCollection">
          <CD>
            <Title></Title>
            <Artist></Artist>
            <Tracks>
              <Track></Track>
              <Track></Track>
            </Tracks>
          </CD>
        </xf:chooseFragment>
      </xf:fragmentToInsert>
    </xf:editWith>
  </xf:xmlToEdit>
</xf:editing>
```


node Attribute

Defines the [XPath](#) expression needed to extract the value of the specified property from the form's underlying XML document.

Type

xsd:string

Remarks

The **node** attribute is a required attribute of the [field](#) element.

Example

The following is an example of the **node** attribute as it is used in the **field** element:

```
<xsf:listProperties>
  <xsf:fields>
    <xsf:field
      type="xsd:date"
      name="TravelDate"
      columnName="TravelDate"
      required="yes"
      viewable="yes"
      node="TravelReport/Header/travelDate"
      aggregation="first"/>
  </xsf:fields>
</xsf:listProperties>
```

onAfterSubmit Attribute

Specifies whether the form should be closed, kept open, or if a new form should be created after a successful submission.

Type

xsd:NMTOKEN

Remarks

The **onAfterSubmit** attribute is an optional attribute of the [submit](#) element.

Values include "Close", "KeepOpen", and "OpenNew". The default value is "KeepOpen".

If the submit operation is not successful, then the **onAfterSubmit** attribute is ignored and the form is kept open.

Example

The following is an example of the **onAfterSubmit** attribute as it is used in the **submit** element:

```
<xf:submit
  caption="Submit"
  disableMenuItem="no"
  onAfterSubmit="KeepOpen"
  showStatusDialog="yes"
  showSignatureReminder="yes">
  <xf:useScriptHandler/>
  <xf:successMessage>Submit was successful.</xf:successMessage>
  <xf:errorMessage>Submit was not successful.</xf:errorMessage>
</xf:submit>
```

orientation Attribute

Specifies the orientation when printing a view.

Type

xsd:NMTOKEN

Remarks

The **orientation** attribute is an optional attribute of the [printSettings](#) element.

Values include "portrait" and "landscape".

Example

The following is an example of the **orientation** attribute as it is used in the **printSettings** element:

```
<xsf:view name="View" caption="View">
  <xsf:printSettings
    header="Header text goes here."
    footer="Footer text goes here."
    orientation="portrait"
    marginUnitsType="in"
    topMargin="1"
    leftMargin="2"
    rightMargin="2"
    bottomMargin="1"
  </xsf:printSettings>
  ...
</xsf:view>
```


overwriteAllowed Attribute

Specifies whether the adapter can overwrite an existing file.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|----------------------------|--|
| davAdapter | Contains information to enable InfoPath files to be submitted to a Microsoft Windows SharePoint Services server or a Web-based Distributed Authoring and Versioning (WebDAV) server. |

Definition

```
<xsd:attribute name="overwriteAllowed" type="xsf:xdYesNo" use="o
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **overwriteAllowed** attribute as it is used in the **davAdapter** element:

```
<xsf:davAdapter name="SubmitToSharePoint" overwriteAllowed="y  
<xsf:fileName value="my:myFields/my:fileName" valueType="expre  
<xsf:folderURL value="http://some_server/some_doc_lib"/>  
</xsf:davAdapter>
```

pageRangeEnd Attribute

Specifies the last page to be printed.

Type

xsd:Integer

Parent Elements

| Element | Description |
|-------------------------------|---|
| printSettings | Specifies the printer settings used when printing the view. |

Definition

```
<xsd:attribute name="pageRangeEnd" >  
  <xsd:simpleType>  
    <xsd:restriction base="xsd:integer">  
      <xsd:minInclusive value="1" />  
      <xsd:maxInclusive value="32000" />  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:attribute>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **pageRangeEnd** attribute as it is used in the **printSettings** element:

```
<xf:printSettings
  orientation="landscape"
  header="&Pqsdf"
  footer="&D"
  printerName="\\printserver\printer"
  paperSource="Auto Select"
  paperSize="Envelope DL"
  topMargin="0.8"
  leftMargin="0.8"
  bottomMargin="0.8"
  rightMargin="0.8"
  marginUnitsType="in"
  copies="2"
  collate="no"
  pageRangeStart="1"
  pageRangeEnd="1" >
  <xf:header>
    <font>
      <div>&Pqsdf</div>
    </font>
  </xf:header>
  <xf:footer>
    <font>
      <div>&D</div>
    </font>
  </xf:footer>
</xf:printSettings>
```

pageRangeStart Attribute

Specifies the first page to be printed.

Type

xsd:Integer

Parent Elements

| Element | Description |
|-------------------------------|---|
| printSettings | Specifies the printer settings used when printing the view. |

Definition

```
<xsd:attribute name="pageRangeStart" >  
  <xsd:simpleType>  
    <xsd:restriction base="xsd:integer">  
      <xsd:minInclusive value="1" />  
      <xsd:maxInclusive value="32000" />  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:attribute>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **pageRangeStart** attribute as it is used in the **printSettings** element:

```
<xf:printSettings
  orientation="landscape"
  header="&Pqsdf"
  footer="&D"
  printerName="\\printserver\printer"
  paperSource="Auto Select"
  paperSize="Envelope DL"
  topMargin="0.8"
  leftMargin="0.8"
  bottomMargin="0.8"
  rightMargin="0.8"
  marginUnitsType="in"
  copies="2"
  collate="no"
  pageRangeStart="1"
  pageRangeEnd="1" >
  <xf:header>
    <font>
      <div>&Pqsdf</div>
    </font>
  </xf:header>
  <xf:footer>
    <font>
      <div>&D</div>
    </font>
  </xf:footer>
</xf:printSettings>
```

paperSize attribute

Specifies the size of the paper.

Type

xsd:String

Parent Elements

| Element | Description |
|-------------------------------|---|
| printSettings | Specifies the printer settings used when printing the view. |

Definition

```
<xsd:attribute name="paperSize" >  
  <xsd:simpleType>  
    <xsd:restriction base="xsd:string">  
      <xsd:maxLength value="255" />  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:attribute>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **paperSize** attribute as it is used in the **printSettings** element:

```
<xf:printSettings
  orientation="landscape"
  header="&Pqsdf"
  footer="&D"
  printerName="\\printserver\printer"
  paperSource="Auto Select"
  paperSize="Envelope DL"
  topMargin="0.8"
  leftMargin="0.8"
  bottomMargin="0.8"
  rightMargin="0.8"
  marginUnitsType="in"
  copies="2"
  collate="no"
  pageRangeStart="1"
  pageRangeEnd="1" >
  <xf:header>
    <font>
      <div>&Pqsdf</div>
    </font>
  </xf:header>
  <xf:footer>
    <font>
      <div>&D</div>
    </font>
  </xf:footer>
</xf:printSettings>
```

paperSource Attribute

Specifies the source of the paper.

Type

xsd:String

Parent Elements

| Element | Description |
|-------------------------------|---|
| printSettings | Specifies the printer settings used when printing the view. |

Definition

```
<xsd:attribute name="paperSource" >  
  <xsd:simpleType>  
    <xsd:restriction base="xsd:string">  
      <xsd:maxLength value="255" />  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:attribute>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **paperSource** attribute as it is used in the **printSettings** element:

```
<xf:printSettings
  orientation="landscape"
  header="&Pqsdf"
  footer="&D"
  printerName="\\printserver\printer"
  paperSource="Auto Select"
  paperSize="Envelope DL"
  topMargin="0.8"
  leftMargin="0.8"
  bottomMargin="0.8"
  rightMargin="0.8"
  marginUnitsType="in"
  copies="2"
  collate="no"
  pageRangeStart="1"
  pageRangeEnd="1" >
  <xf:header>
    <font>
      <div>&Pqsdf</div>
    </font>
  </xf:header>
  <xf:footer>
    <font>
      <div>&D</div>
    </font>
  </xf:footer>
</xf:printSettings>
```


parent Attribute

Specifies a relative [XPath](#) expression from the container node.

Type

xsd:string

Remarks

The **parent** attribute is an optional attribute of the [chooseFragment](#) element.

Refers to the XML [Document Object Model \(DOM\)](#) node under which this fragment should be inserted. The default value is ".", which corresponds to inserting directly under the container parent node.

Example

The following is an example of the **parent** attribute as it is used in the **chooseFragment** element:

```
<xf:editing>
  <xf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xf:editWith caption="CD"
      xd:autogeneration="template"
      component="xCollection">
      <xf:fragmentToInsert>
        <xf:chooseFragment parent="CDCollection"
          followingSiblings=".">
          <CD>
            <Title></Title>
            <Artist></Artist>
            <Tracks>
              <Track></Track>
              <Track></Track>
            </Tracks>
          </CD>
        </xf:chooseFragment>
      </xf:fragmentToInsert>
    </xf:editWith>
  </xf:xmlToEdit>
</xf:editing>
```

printerName Attribute

Specifies the printer name.

Type

xsd:String

Parent Elements

| Element | Description |
|-------------------------------|---|
| printSettings | Specifies the printer settings used when printing the view. |

Definition

```
<xsd:attribute name="printerName">  
  <xsd:simpleType>  
    <xsd:restriction base="xsd:string">  
      <xsd:maxLength value="255" />  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:attribute>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **printerName** attribute as it is used in the **printSettings** element:

```
<xf:printSettings
  orientation="landscape"
  header="&Pqsdf"
  footer="&D"
  printerName="\\printserver\printer"
  paperSource="Auto Select"
  paperSize="Envelope DL"
  topMargin="0.8"
  leftMargin="0.8"
  bottomMargin="0.8"
  rightMargin="0.8"
  marginUnitsType="in"
  copies="2"
  collate="no"
  pageRangeStart="1"
  pageRangeEnd="1" >
  <xf:header>
    <font>
      <div>&Pqsdf</div>
    </font>
  </xf:header>
  <xf:footer>
    <font>
      <div>&D</div>
    </font>
  </xf:footer>
</xf:printSettings>
```

printerSpecificSettings Attribute

Specifies settings for a particular printer.

Type

xsd:String

Parent Elements

| Element | Description |
|-------------------------------|---|
| printSettings | Specifies the printer settings used when printing the view. |

Definition

```
<xsd:attribute name="printerSpecificSettings">  
  <xsd:simpleType>  
    <xsd:restriction base="xsd:string">  
      <xsd:maxLength value="255" />  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:attribute>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **printerSpecificSettings** attribute as it is used in the **printSettings** element:

```
<xsf:printSettings
  orientation="landscape"
  header="&Pqsdf"
  footer="&D"
  printerName="\\printserver\printer"
  paperSource="Auto Select"
  paperSize="Envelope DL"
  topMargin="0.8"
  leftMargin="0.8"
  bottomMargin="0.8"
  rightMargin="0.8"
  marginUnitsType="in"
  copies="2"
  collate="no"
  pageRangeStart="1"
  pageRangeEnd="1"
  printerSpecificSettings="name of file that contains printer-specific
  <xsf:header>
    <font>
      <div>&Pqsdf</div>
    </font>
  </xsf:header>
  <xsf:footer>
    <font>
      <div>&D</div>
    </font>
  </xsf:footer>
```

</xsf:printSettings>

printView Attribute

Specifies the name of another view to use for printing this view.

Type

xsd:string

Remarks

The **printView** attribute is an optional attribute of the [view](#) element.

Example

The following is an example of the **printView** attribute as it is used in the **view** element:

```
<xsf:views default="View">
  <xsf:view name="View" caption="View" printView="PrintView">
    <xsf:mainpane transform="view1.xsl"></xsf:mainpane>
    ...
  </xsf:view>
</xsf:views>
```


productVersion Attribute

Identifies the version number of Microsoft Office InfoPath 2003 [form template](#) with which the form was created or for which a particular form is intended.

Type

xsd:string

Remarks

The **productVersion** attribute is an optional attribute of the [xDocumentClass](#) element.

Its format is #####.#####.##### (major.minor.build).

Example

The following is an example of the **productVersion** attribute as it is used in the **xDocumentClass** element:

```
<xsf:xDocumentClass
  xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solut
  name="urn:microsoft-com:myTravelReport"
  author="AuthorName"
  description="Travel Report form for entering travel reports, issues, e
  dataFormSolution="yes"
  solutionVersion = "1.0.0.1"
  productVersion="11.0.5106"
  solutionFormatVersion="1.0.0.0">
  ...
</xsf:xDocumentClass>
```


promptToSaveChanges Attribute

Specifies whether the user is prompted to save changes to the [form](#) before the action completes.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|-------------------------------------|-------------------------------------|
| closeDocumentAction | Defines a form close action. |

Definition

`<xsd:attribute name="promptToSaveChanges" type="xsd:boolean" use="optional" default="false" />`

Remarks

The default value is "yes".

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **promptToSaveChanges** attribute as it is used in the **closeDocumentAction** element:

```
<xf:rule caption="Receipts" condition="my:expense > 75">
  <xf:dialogBoxMessageAction>Don't forget your receipts!</xf:dialogBoxMessageAction>
  <xf:switchViewAction view="Approval View"/>
  <xf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression="sum(my:expenses/my:expense) * my:taxRate"/>
  <xf:queryAction adapter="Exchange Rates"/>
  <xf:submitAction adapter="Exchange Rates"/>
  <xf:openNewDocumentAction solutionURI="uri:microsoft-Expense1">
    <xf:closeDocumentAction promptToSaveChanges="yes"/>
  <xf:dialogBoxExpressionAction>my:group/my:field1</xf:dialogBoxExpressionAction>
  <xf:exitRuleSet/>
</xf:rule>
```

proofing Attribute

Switches the proofing features, such as the spelling checker, on or off.

Type

xsf:xdYesNo

Remarks

The **proofing** attribute is an optional attribute of the [editWith](#) element. Values include "yes" and "no". The default value is "yes".

Example

The following is an example of the **proofing** attribute as it is used in the **editWith** element:

```
<xsf:editing>
  <xsf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xsf:editWith caption="CD"
      proofing="yes"
      component="xCollection">
      <xsf:fragmentToInsert>
        <xsf:chooseFragment parent="CDCollection">
          <CD>
            <Title></Title>
            <Artist></Artist>
            <Tracks>
              <Track></Track>
              <Track></Track>
            </Tracks>
          </CD>
        </xsf:chooseFragment>
      </xsf:fragmentToInsert>
    </xsf:editWith>
  </xsf:xmlToEdit>
</xsf:editing>
```


publishSaveUrl Attribute

Contains the location of the saved [form template](#) if different from the value of the [publishUrl](#) attribute.

Type

xsd:string

Parent Elements

| Element | Description |
|------------------------------------|--|
| solutionProperties | Contains design-time information about an InfoPath form. |

Definition

```
<xsd:attribute name="publishSaveUrl" type="xsd:string" use="optional"/>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **publishSaveUrl** attribute as it is used in the **solutionProperties** element:

```
<xsf: applicationParameters application="InfoPath Design Mode">
  <xsf: solutionProperties
    allowCustomization="no"
    lastOpenView="view1"
    scriptLanguage="JScript"
    automaticallyCreateNodes="no"
    lastVersionNeedingTransform="1.1.0.10"
    fullyEditableNamespace="urn:names?pace1:mynames"/
    publishSaveUrl="C:\Documents and Settings\username\Desktop\7"
  </xsf:solutionProperties>
</xsf:applicationParameters>
```


publishUrl Attribute

Identifies where the form was published and where the form template should download updates from.

Type

xsd:string

Remarks

The **publishUrl** attribute is an optional attribute of the [xDocumentClass](#) element.

The **publishUrl** attribute is set automatically when a form is published or deployed through the InfoPath [design mode](#). When a form is opened, it will attempt to retrieve the latest updates from the published location.

Example

The following is an example of the **publishUrl** attribute as it is used in the **xDocumentClass** element:

```
<xsf:xDocumentClass
  xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solut
  name="urn:microsoft-com:myTravelReport"
  author="AuthorName"
  description="Travel Report form for entering travel reports, issues, e
  dataFormSolution="yes"
  solutionVersion = "1.0.0.1"
  productVersion="11.0.5106"
  solutionFormatVersion="0.9.0.0"
  publishUrl="http://MyServer/InfoPathTemplates/MyTemplate.xsn">
  ...
</xsf:xDocumentClass>
```

queryAllowed Attribute (adoAdapter Element)

Specifies whether data can be retrieved from the data source through the query method of the data adapter object.

Type

xsf:xdYesNo

Remarks

The **queryAllowed** attribute is an optional attribute of the [adoAdapter](#) element.

Allowed values are "yes" and "no". The default value is "yes".

Example

The following is an example of the **queryAllowed** attribute as it is used in the **adoAdapter** element:

```
<xf:query>
  <xf:adoAdapter
    connectionString="Provider=Microsoft.Jet.OLEDB.4.0;
      Password=&quot;&quot;;User ID=Admin;
      Data Source=infwind.mdb;Mode=Share Deny None;
      Extended Properties=&quot;&quot;;..."
    commandText="select [EmployeeID],[LastName],[FirstName]
      from [Employees] as [Employees]"
    queryAllowed="yes"
    submitAllowed="yes">
  </xf:adoAdapter>
</xf:query>
```


queryAllowed Attribute (davAdapter Element)

Specifies whether the adapter can be used for querying the data source.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|----------------------------|--|
| davAdapter | Contains information to enable InfoPath files to be submitted to a server that is running Microsoft Windows Sharepoint Services or a Web-based Distributed Authoring and Versioning (WebDAV) server. |

Definition

```
<xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optio
```

Remarks

The **queryAllowed** attribute is generally omitted for the **davAdapter** element, corresponding to a default value of "no". A value of "yes" for this attribute causes an error.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **queryAllowed** attribute as it is used in the **davAdapter** element:

```
<xsf:davAdapter name="SubmitToSharePoint" overwriteAllowed="ye
  submitAllowed="yes" queryAllowed="no">
  <xsf:fileName value="my:myFields/my:fileName" valueType="expre
  <xsf:folderURL value="http://contoso/some_doc_lib"/>
</xsf:davAdapter>
```

queryAllowed Attribute (emailAdapter Element)

Specifies whether the adapter can be used for querying the data source.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|------------------------------|--|
| emailAdapter | Contains the information to submit an InfoPath file as an attachment to an e-mail message, with a specified set of recipients, a subject, and an introduction. |

Definition

```
<xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optio
```

Remarks

The **queryAllowed** attribute is generally omitted for the **emailAdapter** element, corresponding to a default value of "no".

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **queryAllowed** attribute as it is used in the **emailAdapter** element:

```
<xsf:emailAdapter name="Submit" submitAllowed="yes" queryAllowed="yes"
  <xsf:to value="someone@example.com" valueType="literal"/>
  <xsf:cc value="my:ccNames" valueType="expression"/>
  <xsf:bcc value="someoneelse@example.com" valueType="literal"/>
  <xsf:subject value="My report" valueType="literal"/>
  <xsf:intro value="See below"/>
  <xsf:attachmentFileName value="Status Report" valueType="literal"/>
</xsf:emailAdapter>
```

queryAllowed Attribute (hwsAdapter Element)

Specifies whether the adapter can be used for querying the data source.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|----------------------------|---|
| hwsAdapter | Defines the Microsoft Biztalk 2004 HWS (Human Workflow Services) data adapter to start or extend an activity flow, and respond to a task. |

Definition

```
<xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optio
```

Remarks

The **queryAllowed** attribute can be omitted from the **hwsAdapter** element, corresponding to a default value of "no".

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **queryAllowed** attribute as it is used in the **hwsAdapter** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval" />
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate" />
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response" />
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response" />
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond" />
  </xsf:allowedTasks>
</xsf:hwsWorkflow>

<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID=
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1" />
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
```

```
        replaceWith="/a:some/b:thing" dataObject="Aux1" />  
    </xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```

queryAllowed Attribute (sharepointListAdapter Element)

Specifies whether the adapter can be used for querying the data source.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|---------------------------------------|--|
| sharepointListAdapter | Contains the data adapter information to query a SharePoint list or library. |

Definition

```
<xsd:attribute name="queryAllowed" type="xsf:xdYesNo" use="optio
```

Remarks

The **queryAllowed** value is always set to "yes" for the **sharepointListdapter**.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **queryAllowed** attribute as it is used in the **sharepointListAdapter** element:

```
<xsf:sharepointListAdapter
  name="Status Report library"
  siteUrl="http://www.contoso.com/reports/"
  sharepointGuid="{ABD2E239-0EE7-48F4-B506-C38A1728E195}"
  infopathGroup="ContosoReportsLibrary"
  queryAllowed="yes">
  <xsf:field
    sharepointName="File_x0020_Type"
    infopathName="Type"></xsf:field>
  <xsf:field
    sharepointName="xd__x007b_D00F1DBD_..."
    infopathName="Title_1"></xsf:field>
</xsf:sharepointListAdapter>
```

queryAllowed Attribute (webServiceAdapter Element)

Specifies whether data can be retrieved from the data source through the query method of the data adapter object.

Type

xsf:xdYesNo

Remarks

The **queryAllowed** attribute is an optional attribute of the [webServiceAdapter](#) element.

Allowed values are "yes" and "no". The default value is "yes".

Example

The following is an example of the **queryAllowed** attribute as it is used in the **webServiceAdapter** element:

```
<xsf:query>
  <xsf:webServiceAdapter
    wsdlUrl="http://localhost/infopathwebservicessample/infopathwebs
queryAllowed="yes"
    submitAllowed="no">
    <xsf:operation
      name="getOrders"
      soapAction="http://tempuri.org/getOrders"
      serviceUrl="http://localhost/infopathwebservicessample/infopathv
      <xsf:input
        source="Submit.xml">
      </xsf:input>
    </xsf:operation>
  </xsf:webServiceAdapter>
</xsf:query>
```


refresh Attribute

Specifies when the [expression](#) will be evaluated.

Type

xsd:string

Parent Elements

| Element | Description |
|---------------------------------|--|
| calculatedField | Defines an individual calculation, including the formula , when the calculation is to be performed, and where the result will be stored. |

Definition

```
<xsd:attribute name="refresh" type="xsd:string" use="required" ></xs
```

Remarks

The expression specified by the **expression** attribute is evaluated according to the **refresh** attribute of the **calculatedField** element. A value of "onInit" causes the expression to be evaluated when the node is initialized. A value of "onChange" causes the expression to be evaluated when one or more parameters of the expression change.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **refresh** attribute as it is used in the **calculatedField** element:

```
<xsf:calculations>
  <xsf:calculatedField
    target="/my:myFields/my:average"
    expression="xdMath:Avg(..my:expenses/my:expense/my:amount)"
    refresh="onChange" />
</xsf:calculations>
```

removeAncestors Attribute

Specifies the number of ancestor (parent) elements to be removed when the last item is removed.

Type

xsd:nonNegativeInteger

Remarks

The **removeAncestors** attribute is an optional attribute of the [editWith](#) element.

Default = 0. Must be a non-negative integer.

Example

The following is an example of the **removeAncestors** attribute as it is used in the **editWith** element:

```
<xf:editing>
  <xf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    <xf:editWith caption="CD"
      removeAncestors="0"
      component="xCollection">
      <xf:fragmentToInsert>
        <xf:chooseFragment parent="CDCollection">
          <CD>
            <Title></Title>
            <Artist></Artist>
            <Tracks>
              <Track></Track>
              <Track></Track>
            </Tracks>
          </CD>
        </xf:chooseFragment>
      </xf:fragmentToInsert>
    </xf:editWith>
  </xf:xmlToEdit>
</xf:editing>
```


replaceWith Attribute

Contains an [XPath](#) expression that identifies the values in the source document that should be used to replace parts of the input Simple Object Access Protocol (SOAP) message.

Type

xsd:string

Remarks

The **replaceWith** attribute is a required attribute of the [partFragment](#) element.

Example

The following is an example of the **replaceWith** attribute as it is used in the **partFragment** element:

```
<xf:query>
  <xf:webServiceAdapter
    wsdlUrl="http://localhost/infopathwebservicessample/infopathwebs
    queryAllowed="yes"
    submitAllowed="no">
    <xf:operation
      name="getOrders"
      soapAction="http://tempuri.org/getOrders"
      serviceUrl="http://localhost/infopathwebservicessample/infopathv
    <xf:input
      source="Submit.xml">
      <xf:partFragment
        match="/dfs:myFields/dfs:dataFields/s0:IsPrime/s0:inValue"
        replaceWith="/dfs:myFields/dfs:dataFields/s0:IsPrime" />
    </xf:input>
  </xf:operation>
</xf:webServiceAdapter>
</xf:query>
```

required Attribute

Identifies whether this field accepts Null values.

Type

xsf:xdYesNo

Remarks

The **required** attribute is an optional attribute of the [field](#) element. Values include "yes" and "no". The default value is "no".

Example

The following is an example of the **required** attribute as it is used in the **field** element:

```
<xsf:listProperties>
  <xsf:fields>
    <xsf:field
      type="xsd:date"
      name="TravelDate"
      columnName="TravelDate"
      required="yes"
      viewable="yes"
      node="TravelReport/Header/travelDate"
      aggregation="first"/>
  </xsf:fields>
</xsf:listProperties>
```

requireFullTrust Attribute

Allows the form to run as a fully trusted form when a form template is registered or signed with a certificate.

Type

xsf:xdYesNo

Remarks

The **requireFullTrust** attribute is an optional attribute of the [xDocumentClass](#) element.

Forms that have the **requireFullTrust** attribute set to "yes" get full trust security privileges in Microsoft Office InfoPath 2003. Allowed attribute values are "yes" and "no". The default value is "no".

Note Fully trusted forms need to be registered, otherwise they cannot be opened in InfoPath.

Example

The following is an example of the **requireFullTrust** attribute as it is used in the **xDocumentClass** element:

```
<xsf:xDocumentClass
  xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solut
  name="urn:microsoft-com:myTravelReport"
  author="AuthorName"
  description="Travel Report form for entering travel reports, issues, e
  dataFormSolution="yes"
  solutionVersion = "1.0.0.1"
  productVersion="11.0.5106"
  solutionFormatVersion="1.0.0.0"
  requireFullTrust="yes">
  ...
</xsf:xDocumentClass>
```

rightMargin Attribute

Specifies the right margin when printing a view.

Type

xsd:string

Remarks

The **rightMargin** attribute is an optional attribute of the [printSettings](#) element.

The **rightMargin** attribute must be greater than or equal to zero.

Example

The following is an example of the **rightMargin** attribute as it is used in the **printSettings** element:

```
<xsf:view name="View" caption="View">
  <xsf:printSettings
    header="Header text goes here."
    footer="Footer text goes here."
    orientation="portrait"
    marginUnitsType="in"
    topMargin="1"
    leftMargin="2"
    rightMargin="2"
    bottomMargin="1"
  </xsf:printSettings>
  ...
</xsf:view>
```


rootSchema Attribute

Identifies an [XML Schema](#) as the top-level schema of the form being filled out.

Type

xsf:xdYesNo

Remarks

The **rootSchema** attribute is an optional attribute of the [documentSchema](#) element.

Only one of the XML Schemas defined for a form can be marked as the root schema. Allowed values are "yes" and "no". The default value is "no".

Example

The following is an example of the **rootSchema** attribute as it is used in the **documentSchema** element:

```
<xsf:documentSchemas>  
  <xsf:documentSchema  
    location="urn:schema:custom:Namespace customFilename.xsd"  
    rootSchema="yes"/>  
</xsf:documentSchemas>
```

ruleSet Attribute

Contains the name of the [ruleSet](#) action element to be invoked.

Type

xsd:string

Parent Elements

| Element | Description |
|-------------------------------|--|
| ruleSetAction | Defines the ruleSet action element to be invoked. |

Definition

```
<xsd:attribute name="ruleSet" type="xsd:string" use="required"></xsd:attribute>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **ruleSet** attribute as it is used in the **ruleSetAction** element:

```
<xsf:onLoad>  
  <xsf:ruleSetAction ruleSet="RuleSet4"/>  
</xsf:onLoad>
```


schema Attribute (dataObject Element)

The name of an [XML Schema](#) file.

Type

xsd:string

Remarks

The schema **attribute** is an optional attribute of the [dataObject](#) element.

Microsoft Office InfoPath 2003 automatically packages the XML Schema for each secondary data object as part of the [form template](#). An entry is made in the [form definition \(.xsf\) file](#) for the XML Schema file using the [files](#) element, and it is the file name that is referenced from the **schema** attribute of the **dataObject** element.

Example

The following is an example of the **schema** attribute as it is used in the **dataObject** element:

```
<xsf:dataObjects>
  <xsf:dataObject
    name="EmployeeNames"
    schema="EmployeeNames.xsd"
    initOnLoad="yes">
    <xsf:query>
      <xsf:adoAdapter
        connectionString="Provider=Microsoft.Jet.OLEDB.4.0;
          Password=&quot;&quot;;User ID=Admin;
          Data Source=infwind.mdb;Mode=Share Deny None;
          Extended Properties=&quot;&quot;;..."
        commandText="select [EmployeeID],[LastName],[FirstName]
          from [Employees] as [Employees]"
        queryAllowed="yes"
        submitAllowed="yes">
      </xsf:adoAdapter>
    </xsf:query>
  </xsf:dataObject>
</xsf:dataObjects>
```


schema Attribute (importSource Element)

Identifies the [XML Schema](#) file that should be used during the merge operation to validate the form being merged.

Type

xsf:xdFileName

Remarks

The **schema** attribute is a required attribute of the [importSource](#) element.

If the source document to be merged belongs to the specified schema, the specified parameters are used for merging into the current form. If the **schema** attribute is not specified, a "Schema XSD not found" error is returned.

Example

The following is an example of the **schema** attribute as it used in the **importSource** element:

```
<xsf:importParameters
  enabled="yes"
  <xsf:importSource
    name="My Form"
    schema="MySchema.xsd"
    transform="schematransform.xslt"/>
</xsf:importParameters>
```

scriptLanguage Attribute

Identifies the name of the scripting language used to implement the business logic of the Microsoft Office InfoPath 2003 form.

Type

xsf:xdScriptLanguage

Remarks

The **scriptLanguage** attribute is an optional attribute of the [solutionProperties](#) element.

InfoPath supports the following values for this attribute: "JavaScript", "JScript", and "VBScript".

Example

The following is an example of the **scriptLanguage** attribute as it is used in the **solutionProperties** element:

```
<xsf: applicationParameters application="InfoPath Design Mode">  
  <xsf: solutionProperties  
    allowCustomization="no"  
    lastOpenView="view1"  
    scriptLanguage="JScript"  
    automaticallyCreateNodes="no"  
    lastVersionNeedingTransform="1.0.0.10"  
    fullyEditableNamespace="urn:namespace1:mynames"/>  
</xsf:applicationParameters>
```


select Attribute

An [XPath](#) query expression that returns one or more data nodes that contain user names.

Type

xsd:string

Parent Elements

| Element | Description |
|-------------------------------------|--|
| getUserNameFromData | Retrieves a user name by using an XPath query of the data in the main data source or in a secondary data source. |

Definition

```
<xsd:attribute name="select" type="xsd:string" use="required"></xsd:
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **select** attribute as it is used in the **getUserNameFromData** element:

```
<xf:roles initiator="A" default="C" hideStatusBarDisplay="yes">
  <xf:role name="A"/>
  <xf:role name="B"/>
  <xf:role name="C"/>

  <xf:membership>
    <xf:getUserNameFromData dataObject="catalog"
      select="/dfs:myFields/dfs:dataFields/d:UserA" memberOf="B"/>
    <xf:userName name="Domain\username1" memberOf="A"/>
    <xf:userName name="Domain\username2" memberOf="B"/>
    <xf:group name="Domain\username3" memberOf="C"/>
  </xf:membership>
</xf:roles>
```

sendAsString Attribute

Specifies that the data is submitted as a string through the [webServiceAdapter](#) data adapter.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|------------------------------|---|
| partFragment | Defines one substitution group for a specific part of the input Simple Object Access Protocol (SOAP) message. |

Definition

```
<xsd:attribute name="sendAsString" type="xsf:xdYesNo" use="option
```

Remarks

It may be necessary to set the **sendAsString** attribute to "yes" for digitally signed data, because this setting preserves non-printing characters in the data.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **sendAsString** attribute as it is used in the **partFragment** element:

```
<xsf:partFragment match="/dfs:myFields/dfs:dataFields/s0:IsPrime/s0  
replaceWith=/dfs:myFields/dfs:dataFields/s0:IsPrime"  
filter="." sendAsString="yes"/>
```


serviceUrl Attribute

Contains the Web service [Uniform Resource Locator \(URL\)](#) to which the request should be sent.

Type

xsd:string

Remarks

The **serviceUrl** attribute is a required attribute of the [operation](#) element.

Example

The following is an example of the **serviceUrl** attribute as it is used in the **operation** element:

```
<xsf:query>
  <xsf:webServiceAdapter
    wsdlUrl="http://localhost/infopathwebservicessample/infopathwebs
    queryAllowed="yes"
    submitAllowed="no">
    <xsf:operation
      name="getOrders"
      soapAction="http://tempuri.org/getOrders"
      serviceUrl="http://localhost/infopathwebservicessample/infopath
    <xsf:input
      source="Submit.xml">
    </xsf:input>
    </xsf:operation>
  </xsf:webServiceAdapter>
</xsf:query>
```


serviceUrl Attribute (hwsOperation Element)

Specifies the [Uniform Resource Locator \(URL\)](#) of the Microsoft BizTalk 2004 Human Workflow Services (HWS) Web service.

Type

xsd:string

Parent Elements

| Element | Description |
|------------------------------|--|
| hwsOperation | Defines the HWS operation type, such as adding an action to a new activity flow, adding an action to an existing activity flow, or responding to a task. |

Definition

```
<xsd:attribute name="serviceUrl" type="xsd:string" use="required"><
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **serviceUrl** attribute as it is used in the **hwsOperation** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval"/>
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate"/>
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response"/>
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response"/>
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond"/>
  </xsf:allowedTasks>
</xsf:hwsWorkflow>

<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID="
    serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1"/>
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
```

```
        replaceWith="/a:some/b:thing" dataObject="Aux1"/>  
    </xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```

sharepointGuid Attribute

Contains the GUID of the SharePoint list.

Type

xsd:string

Parent Elements

| Element | Description |
|---------------------------------------|---|
| sharepointListAdapter | Contains the data adapter information needed to query a SharePoint list or library. |

Definition

```
<xsd:attribute name="sharepointGuid" type="xsd:string" use="require
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **sharepointGuid** attribute as it is used in the **sharepointListAdapter** element:

```
<xsf:sharepointListAdapter
  name="Status Report library"
  siteUrl="http://xyzco/reports/"
  sharepointGuid="{ABD2E239-0EE7-48F4-B506-C38A1728E195}"
  infopathGroup="XyzReportsLibrary"
  queryAllowed="yes">
  <xsf:field
    sharepointName="File_x0020_Type"
    infopathName="Type"></xsf:field>
  <xsf:field
    sharepointName="xd__x007b_D00F1DBD_..."
    infopathName="Title_1"></xsf:field>
</xsf:sharepointListAdapter>
```


sharepointName Attribute

Contains the name of a [field](#) in a SharePoint list.

Type

xsd:string

Parent Elements

| Element | Description |
|-----------------------|--|
| field | Contains field mapping information for each field in a SharePoint list and the corresponding name used in Microsoft Office InfoPath 2003 Service Pack 1. |

Definition

```
<xsd:attribute name="sharepointName" type="xsd:string" use="required"/>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **sharepointName** attribute as it is used in the **field** element:

```
<xsf:field  
  sharepointName="xd__x007b_D00F1DBD_..."  
  infopathName="Title_1"  
  isLookup="no">  
</xsf:field>
```

shortMessage Attribute

Identifies the short error message to return in case of invalid data.

Type

xsd:string

Remarks

The **shortMessage** attribute is a required attribute of the [errorMessage](#) element.

This is displayed in a ScreenTip in the default error user interface. For modal errors, this attribute is ignored and the detailed message is used instead. Maximum length of the error message is 127 characters.

Example

The following is an example of the **shortMessage** attribute as it is used in the **errorMessage** element:

```
<xsf:customValidation>
  <xsf:errorCondition
    match="/exp:expenseReport"
    expressionContext="exp:reportDate"
    expression="msxsl:string-compare(., ../exp:startDate) < 0 and ../exp:
    showErrorOn=".">
    <xsf:errorMessage
      type="modeless"
      shortMessage="The report date occurs before the end of the exp
      The report date occurs before the end of the expense period. Veri:
    </xsf:errorMessage>
  </xsf:errorCondition>
</xsf:customValidation>
```


showErrorOn Attribute

Identifies XML [Document Object Model \(DOM\)](#) nodes (within the context of the expression context XML DOM node) on which the error should be displayed when the form is filled out.

Type

xsd:string

Remarks

The **showErrorOn** attribute is an optional attribute of the [errorCondition](#) element.

Contains (relative to expression context XML DOM node) an [XPath](#) expression. Default is "." This is the same as the expression context.

Example

The following is an example of the **showErrorOn** attribute as it is used in the **errorCondition** element:

```
<xsf:customValidation>
  <xsf:errorCondition
    match="/exp:expenseReport"
    expressionContext="exp:reportDate"
    expression="msxsl:string-compare(., ../exp:startDate) < 0 and ../exp:
showErrorOn=".">
  <xsf:errorMessage
    type="modeless"
    shortMessage="The report date occurs before the end of the expe
    The report date occurs before the end of the expense period. Veri:
  </xsf:errorMessage>
</xsf:errorCondition>
</xsf:customValidation>
```

showIf Attribute

Specifies the editing context of the button.

Type

xsd:NMTOKEN

Remarks

The **showIf** attribute is an optional attribute of the [button](#) element.

Allowed values include "always", "enabled", and "immediate". Default value is "always". Only applies to buttons used with editing components. If the **showIf** attribute is set to "enabled", then the button is visible only if the action is contextually enabled. If the **showIf** attribute is set to "immediate", then the button is visible only if the action is contextually immediate.

Example

The following is an example of the **showIf** attribute as it is used in the **button** element:

```
<xsf:menuArea name="msoInsertMenu">
  <xsf:menu caption="&Section">
    <xsf:button action="xCollection::insert" xmlToEdit="CD_10"
      caption="CD" showIf="immediate"></xsf:button>
    <xsf:button action="xCollection::insert" xmlToEdit="Track_14"
      caption="Track" showIf="immediate"></xsf:button>
    <xsf:button action="xOptional::insert" xmlToEdit="Label_16"
      caption="Label"></xsf:button>
  </xsf:menu>
</xsf:menuArea>
```


showMenuItem Attribute

Displays a menu item in the InfoPath **View** menu and adds a check mark next to the item when it is in use.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|----------------------|--|
| view | Contains information about an InfoPath view. |

Definition

```
<xsd:attribute name="showMenuItem" type="xsf:xdYesNo" use="opti
```

Remarks

The **showMenuItem** attribute is an optional attribute of the **view** element. The default value is "yes".

If the **showMenuItem** attribute for a menu item in the **View** menu is set to "yes", the item appears in the **View** menu and has a check mark next to it when it is in use. The menu item appears with the caption that has been defined for it and in the order specified in the [form definition file \(.xsf\)](#). If the **showMenuItem** attribute is not present or is set to "no", the menu item does not appear in the **View** menu. Users can still add items to the **View** menu by customizing it, but such items will not display a check mark when they are in use.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **showMenuItem** attribute as it is used in the **view** element:

```
<xsf:view name="View 1" caption="First view" showMenuItem="yes"  
  <xsf:mainpane transform="view1.xsl"></xsf:mainpane>  
</xsf:view>
```

showSignatureReminder Attribute

Specifies whether a dialog box should be displayed to prompt the user to digitally sign the document before submitting it.

Type

xsf:xdYesNo

Remarks

The **showSignatureReminder** attribute is an optional attribute of the [submit](#) element.

Values include "yes" and "no". The default value is "no". If set to "yes" and the form is not digitally signed when a user tries to submit it, the dialog box is displayed.

Example

The following is an example of the **showSignatureReminder** attribute as it is used in the **submit** element:

```
<xf:submit
  caption="Submit"
  disableMenuItem="no"
  onAfterSubmit="KeepOpen"
  showStatusDialog="yes"
  showSignatureReminder="yes">
  <xf:useScriptHandler/>
  <xf:successMessage>Submit was successful.</xf:successMessage>
  <xf:errorMessage>Submit was not successful.</xf:errorMessage>
</xf:submit>
```

showStatusDialog Attribute

Specifies whether the status dialog box should be shown after the submit operation.

Type

xsf:xdYesNo

Remarks

The **showStatusDialog** attribute is an optional attribute of the [submit](#) element.

Values include "yes" and "no". The default value is "yes".

If the **showStatusDialog** attribute is set to "yes" and no custom messages are defined (using the [errorMessage](#) or [successMessage](#) elements), then the Microsoft Office InfoPath 2003 default submit messages are displayed.

Example

The following is an example of the **showStatusDialog** attribute as it is used in the **submit** element:

```
<xf:submit
  caption="Submit"
  disableMenuItem="no"
  onAfterSubmit="KeepOpen"
  showStatusDialog="yes"
  showSignatureReminder="yes">
  <xf:useScriptHandler/>
  <xf:successMessage>Submit was successful.</xf:successMessage>
  <xf:errorMessage>Submit was not successful.</xf:errorMessage>
</xf:submit>
```


signatureLocation Attribute

Contains an [XPath](#) expression that points to the XML [Document Object Model \(DOM\)](#) node within the form's underlying XML document that is used for storing the digital signature.

Type

xsd:string

Remarks

The **signatureLocation** attribute is a required attribute of the [documentSignatures](#) element.

Example

The following is an example of the **signatureLocation** attribute as it is used in the **documentSignatures** element:

```
<xsf:documentSignatures  
  signatureLocation="/employees/my:signatures1">  
</xsf:documentSignatures>
```


signatureLocation Attribute (signedDataBlock Element)

Contains an [XPath](#) expression that points to the XML Document Object Model (DOM) node in the form's underlying XML document that is used for storing the digital signature.

Type

xsd:string

Parent Elements

| Element | Description |
|---------------------------------|---|
| signedDataBlock | Defines a node set in the form's underlying XML document to which a digital signature can be applied. |

Definition

```
<xsd:attribute name="signatureLocation" type="xsd:string" use="requ
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **signatureLocation** attribute as it is used in the **signedDataBlock** element:

```
<xf:documentSignatures>
  <xf:signedDataBlock name="main"
    data="my:myfields/my:subtree1 | my:myfields/my:subtree2"
    signatureLocation="my:mifields/sig:signatures/main"
    mode="countersign">
    <xf:message>By pressing the "Sign" button below, I agree
      to the terms of this document.</xf:message>
  </xf:signedDataBlock>
</xf:documentSignatures>
```


siteUrl Attribute

Contains the [Uniform Resource Locator \(URL\)](#) of a SharePoint site.

Type

xsd:string

Parent Elements

| Element | Description |
|---------------------------------------|---|
| sharepointListAdapter | Contains the data adapter information needed to query a SharePoint list or library. |

Definition

```
<xsd:attribute name="siteUrl" type="xsd:string" use="required"></xsd:attribute>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **siteUrl** attribute as it is used in the **sharepointListAdapter** element:

```
<xf:sharepointListAdapter
  name="Status Report library"
  siteUrl="http://xyzco/reports/"
  sharepointGuid="{ABD2E239-0EE7-48F4-B506-C38A1728E195}"
  infopathGroup="XyzReportsLibrary"
  queryAllowed="yes">
  <xf:field
    sharepointName="File_x0020_Type"
    infopathName="Type"></xf:field>
  <xf:field
    sharepointName="xd__x007b_D00F1DBD_..."
    infopathName="Title_1"></xf:field>
</xf:sharepointListAdapter>
```

soapAction Attribute

Contains the value of the **SOAPAction** attribute in the Simple Object Access Protocol (SOAP) request message.

Type

xsd:string

Remarks

The **soapAction** attribute is a required attribute of the [operation](#) element.

Example

The following is an example of the **soapAction** attribute as it is used in the **operation** element:

```
<xsf:query>
  <xsf:webServiceAdapter
    wsdlUrl="http://localhost/infopathwebservicessample/infopathwebs
    queryAllowed="yes"
    submitAllowed="no">
    <xsf:operation
      name="getOrders"
      soapAction="http://tempuri.org/getOrders"
      serviceUrl="http://localhost/infopathwebservicessample/infopathv
    <xsf:input
      source="Submit.xml">
    </xsf:input>
    </xsf:operation>
  </xsf:webServiceAdapter>
</xsf:query>
```


solutionFormatVersion Attribute

Identifies the version number that represents the format of the [form definition file \(.xsf\)](#).

Type

xsf:xdSolutionVersion

Remarks

The **solutionFormatVersion** attribute is a required attribute of the [xDocumentClass](#) element.

Its format is #####.#####.#####.##### (major.minor.revision.build). The version number of the .xsf file allows Microsoft Office InfoPath 2003 to determine whether the current form is compatible with the product version in which it is being opened.

Example

The following is an example of the **solutionFormatVersion** attribute as it is used in the **xDocumentClass** element:

```
<xsf:xDocumentClass
  xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solut
  name="urn:microsoft-com:myTravelReport"
  author="AuthorName"
  description="Travel Report form for entering travel reports, issues, e
  dataFormSolution="yes"
  solutionVersion = "1.0.0.1"
  productVersion="11.0.5106"
  solutionFormatVersion="1.0.0.0">
  ...
</xsf:xDocumentClass>
```


solutionURI Attribute

The [Uniform Resource Identifier \(URI\)](#) of the solution on which the new form will be based.

Type

xsd:anyURI

Parent Elements

| Element | Description |
|---------------------------------------|-------------------------------|
| openNewDocumentAction | Defines a form create action. |

Definition

```
<xsd:attribute name="solutionURI" type="xsd:anyURI" use="required
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **solutionURI** attribute as it is used in the **openNewDocumentAction** element:

```
<xf:rule caption="Receipts" condition="my:expense > 75">
  <xf:dialogBoxMessageAction>Don't forget your receipts!</xf:dialogBoxMessageAction>
  <xf:switchViewAction view="Approval View"/>
  <xf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression="sum(my:expenses/my:expense) * my:taxRate"/>
  <xf:queryAction adapter="Exchange Rates"/>
  <xf:submitAction adapter="Exchange Rates"/>
  <xf:openNewDocumentAction solutionURI="uri:microsoft-Expense">
  <xf:closeDocumentAction promptToSaveChanges="yes"/>
  <xf:dialogBoxExpressionAction>my:group/my:field1</xf:dialogBoxExpressionAction>
  <xf:exitRuleSet/>
</xf:rule>
```

solutionVersion Attribute

Identifies the version number of the form.

Type

xsf:xdSolutionVersion

Remarks

The **solutionVersion** attribute is an optional attribute of the [xDocumentClass](#) element.

Its format is #####.#####.#####.##### (major.minor.revision.build).

Example

The following is an example of the **solutionVersion** attribute as it is used in the **xDocumentClass** element:

```
<xsf:xDocumentClass
  xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solut
  name="urn:microsoft-com:myTravelReport"
  author="AuthorName"
  description="Travel Report form for entering travel reports, issues, e
  dataFormSolution="yes"
  solutionVersion = "1.0.0.1"
  productVersion="11.0.5106"
  solutionFormatVersion="1.0.0.0">
  ...
</xsf:xDocumentClass>
```


source Attribute

Contains the name of the resource file in the [form template](#) that contains the [XML Schema](#) for the input Simple Object Access Protocol (SOAP) message of the selected operation of the Web service.

Type

xsd:string

Remarks

The **source** attribute is a required attribute of the [input](#) element.

Example

The following is an example of the **source** attribute as it is used in the **input** element:

```
<xf:query>
  <xf:webServiceAdapter
    wsdlUrl="http://localhost/infopathwebservicessample/infopathwebs
    queryAllowed="yes"
    submitAllowed="no">
    <xf:operation
      name="getOrders"
      soapAction="http://tempuri.org/getOrders"
      serviceUrl="http://localhost/infopathwebservicessample/infopathv
    <xf:input
      source="Submit.xml">
      <xf:partFragment
        match="/dfs:myFields/dfs:dataFields/s0:IsPrime/s0:inValue"
        replaceWith="/dfs:myFields/dfs:dataFields/s0:IsPrime" />
    </xf:input>
  </xf:operation>
</xf:webServiceAdapter>
</xf:query>
```


src Attribute

Provides a [relative URL](#) within the [form template](#) to the specified script source file.

Type

xsf:xdFileName

Remarks

The **src** attribute is a required attribute of the [script](#) element.

Example

The following is an example of the **src** attribute as it is used in the **script** element:

```
<xsf:scripts language="jscript">  
  <xsf:script src="myscripts.js"/>  
</xsf:scripts>
```

submitAllowed Attribute (adoAdapter Element)

Specifies whether data can be submitted to the data source through the submit method of the data adapter object.

Type

xsf:xdYesNo

Remarks

The **submitAllowed** attribute is an optional attribute of the [adoAdapter](#) element.

Allowed values are "yes" and "no". The default value is "no".

Example

The following is an example of the **submitAllowed** attribute as it is used in the **adoAdapter** element:

```
<xf:query>
  <xf:adoAdapter
    connectionString="Provider=Microsoft.Jet.OLEDB.4.0;
      Password=&quot;&quot;;User ID=Admin;
      Data Source=infwind.mdb;Mode=Share Deny None;
      Extended Properties=&quot;&quot;;..."
    commandText="select [EmployeeID],[LastName],[FirstName]
      from [Employees] as [Employees]"
    queryAllowed="yes"
    submitAllowed="yes">
  </xf:adoAdapter>
</xf:query>
```


submitAllowed Attribute (davAdapter Element)

Specifies whether the adapter can be used for submitting to the data source.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|----------------------------|--|
| davAdapter | Contains information needed to submit InfoPath forms to a server running Microsoft Windows SharePoint Services or to a Web-based Distributed Authoring and Versioning (WebDAV) server. |

Definition

```
<xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="opti
```

Remarks

The value of the **submitAllowed** attribute is always set to "yes" for the **davAdapter** element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **submitAllowed** attribute as it is used in the **davAdapter** element:

```
<xsf:davAdapter name="SubmitToSharePoint" overwriteAllowed="ye
  <xsf:fileName value="my:myFields/my:fileName" valueType="expre
  <xsf:folderURL value="http://some_server/some_doc_lib"/>
</xsf:davAdapter>
```

submitAllowed Attribute (emailAdapter Element)

Specifies whether the adapter can be used for submitting to the data source.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|------------------------------|---|
| emailAdapter | Contains the information needed to submit an InfoPath form as an attachment to an e-mail message, with a specified set of recipients, a subject, and an introduction. |

Definition

```
<xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="opti
```

Remarks

The value of the **submitAllowed** attribute is always set to "yes" for the **emailAdapter** element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **submitAllowed** attribute as it is used in the **emailAdapter** element:

```
<xsf:emailAdapter name="Submit" submitAllowed="yes">  
  <xsf:to value="someone@example.com" valueType="literal"/>  
  <xsf:cc value="my:ccNames" valueType="expression"/>  
  <xsf:bcc value="someoneelse@example.com" valueType="literal"/>  
  <xsf:subject value="My report" valueType="literal"/>  
  <xsf:intro value="See below"/>  
  <xsf:attachmentFileName value="Status Report" valueType="literal"/>  
</xsf:emailAdapter>
```

submitAllowed Attribute (hwsAdapter Element)

Specifies whether the adapter can be used for submitting to the data source.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|----------------------------|--|
| hwsAdapter | Defines the Microsoft BizTalk 2004 Human Workflow Services (HWS) data adapter, which can be used to start or extend an activity flow or respond to a task. |

Definition

```
<xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="opti
```

Remarks

The value of the **submitAllowed** attribute is always set to "yes" for the **hwsAdapter** element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **submitAllowed** attribute as it is used in the **hwsAdapter** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval"/>
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate"/>
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response"/>
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response"/>
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond"/>
  </xsf:allowedTasks>
</xsf:hwsWorkflow>

<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID="
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1"/>
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
```

```
        replaceWith="/a:some/b:thing" dataObject="Aux1"/>  
    </xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```

submitAllowed Attribute (sharepointListAdapter Element)

Specifies whether the adapter can be used for submitting to the data source.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|---------------------------------------|---|
| sharepointListAdapter | Contains the data adapter information needed to query a SharePoint list or library. |

Definition

```
<xsd:attribute name="submitAllowed" type="xsf:xdYesNo" use="opti
```

Remarks

The **submitAllowed** attribute is omitted for the **sharepointListAdapter** element, corresponding to a default value of "no".

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **submitAllowed** attribute as it is used in the **sharepointListAdapter** element:

```
<xsf:sharepointListAdapter
  name="Status Report library"
  siteUrl="http://xyzco/reports/"
  sharepointGuid="{ABD2E239-0EE7-48F4-B506-C38A1728E195}"
  infopathGroup="XyzReportsLibrary"
  queryAllowed="yes">
  <xsf:field
    sharepointName="File_x0020_Type"
    infopathName="Type"></xsf:field>
  <xsf:field
    sharepointName="xd__x007b_D00F1DBD_..."
    infopathName="Title_1">
  </xsf:field>
</xsf:sharepointListAdapter>
```

submitAllowed Attribute (webServiceAdapter Element)

Specifies whether data can be submitted to the data source through the submit method of the data adapter object.

Type

xsf:xdYesNo

Remarks

The **submitAllowed** attribute is an optional attribute of the [webServiceAdapter](#) element.

Allowed values are "yes" and "no". The default value is "no".

Example

The following is an example of the **submitAllowed** attribute as it is used in the **webServiceAdapter** element:

```
<xsf:query>
  <xsf:webServiceAdapter
    wsdlUrl="http://localhost/infopathwebservicessample/infopathwebs
    queryAllowed="yes"
    submitAllowed="no">
    <xsf:operation
      name="getOrders"
      soapAction="http://tempuri.org/getOrders"
      serviceUrl="http://localhost/infopathwebservicessample/infopathv
    <xsf:input
      source="Submit.xml">
    </xsf:input>
    </xsf:operation>
  </xsf:webServiceAdapter>
</xsf:query>
```


target Attribute (calculatedField Element)

Contains the [XPath](#) location where the result of the [expression](#) attribute will be stored.

Type

xsd:string

Parent Elements

| Element | Description |
|---------------------------------|--|
| calculatedField | Defines an individual calculation, including the formula , when the calculation is to be performed, and where the result will be stored. |

Definition

```
<xsd:attribute name="target" type="xsd:string" use="required" ></xsd
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **target** attribute as it is used in the **calculatedField** element:

```
<xsf:calculations>  
  <xsf:calculatedField  
    target="/my:myFields/my:average"  
    expression="xdMath:Avg(..my:expenses/my:expense/my:amount)"  
    refresh="onChange" />  
</xsf:calculations>
```


targetField Attribute

Contains an [XPath](#) expression for the target node.

Type

xsd:string

Parent Elements

| Element | Description |
|----------------------------------|---|
| assignmentAction | Defines an action to set the value of a field . |

Definition

```
<xsd:attribute name="targetField" type="xsd:string" use="required"><
```

Remarks

The **targetField** attribute identifies the node that will receive the value from the [expression](#) attribute.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **targetField** attribute as it is used in the **assignmentAction** element:

```
<xf:rule caption="Receipts" condition="my:expense > 75">
  <xf:dialogBoxMessageAction>Don't forget your receipts!</xf:dialogBoxMessageAction>
  <xf:switchViewAction view="Approval View"/>
  <xf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression = "sum(my:expenses/my:expense) * my:taxRate"/>
  <xf:queryAction adapter="Exchange Rates"/>
  <xf:submitAction adapter="Exchange Rates"/>
  <xf:openNewDocumentAction solutionURI="uri:microsoft-Expense1">
  <xf:closeDocumentAction promptToSaveChanges="yes"/>
  <xf:dialogBoxExpressionAction>my:group/my:field1</xf:dialogBoxExpressionAction>
  <xf:exitRuleSet/>
</xf:rule>
```

taskpaneVisible Attribute

Specifies whether the **Workflow** task pane is visible.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|-----------------------------|--|
| hwsWorkflow | Contains the information to enable the Workflow task pane and to enable individual actions and tasks associated with a Microsoft Biztalk 2004 Human Workflow Services (HWS) server. |

Definition

```
<xsd:attribute name="taskpaneVisible" type="xsd:boolean" />
```

Remarks

The default value for the **taskpaneVisible** attribute is "yes". The **taskpaneVisible** attribute is an optional attribute of the **hwsWorkflow** element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **taskpaneVisible** attribute as it is used in the **hwsWorkflow** element:

```
<xf:hwsWorkflow taskpaneVisible="yes">
  <xf:location url="http://www.contoso.com/hwsservice/hwsservice.as
  <xf:allowedActions>
    <xf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval" />
    <xf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate" />
  </xf:allowedActions>
  <xf:allowedTasks>
    <xf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response" />
    <xf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response" />
    <xf:task name="delegateToManager" taskTypeID="420"
      caption="Respond" />
  </xf:allowedTasks>
</xf:hwsWorkflow>
```


taskTypeID Attribute

The unique ID of the Microsoft BizTalk Server 2004 Human Workflow Services (HWS) workflow task.

Type

xsd:string

Parent Elements

| Element | Description |
|----------------------|---|
| task | The HWS task information enabled for the form . |

Definition

```
<xsd:attribute name="taskTypeID" type="xsd:string" use="required">
```

Remarks

The **taskTypeID** attribute uses a [namespace](#) to uniquely identify a task but does not represent the current instance of the task being performed. The current instance of the task is stored in the XML instance file.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **taskTypeID** attribute as it is used in the **task** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval"/>
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate"/>
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response"/>
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response"/>
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond"/>
  </xsf:allowedTasks>
</xsf:hwsWorkflow>

<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes" queryAllowed="no">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID="
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1"/>
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
```

```
        replaceWith="/a:some/b:thing" dataObject="Aux1"/>  
    </xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```

tooltip Attribute

Provides the ScreenTip text to be used for the button.

Type

xsf:xdTitle

Remarks

The **tooltip** attribute is an optional attribute of the [button](#) element.

This attribute applies only to buttons used on a toolbar.

Example

The following is an example of the **tooltip** attribute as it is used in the **button** element:

```
<xsf:toolbar caption="CD Collection Toolbar" name="CD Collection T
  <xsf:button action="xCollection::insert" xmlToEdit="CD_10"
    caption="New CD" showIf="always"
      tooltip="Insert a CD.">
  </xsf:button>
  ...
</xsf:toolbar>
```

topMargin Attribute

Specifies the top margin when printing a view.

Type

xsd:string

Remarks

The **topMargin** attribute is an optional attribute of the [printSettings](#) element.

The **topMargin** attribute must be greater than or equal to zero.

Example

The following is an example of the **topMargin** attribute as it is used in the **printSettings** element:

```
<xsf:view name="View" caption="View">
  <xsf:printSettings
    header="Header text goes here."
    footer="Footer text goes here."
    orientation="portrait"
    marginUnitsType="in"
    topMargin="1"
    leftMargin="2"
    rightMargin="2"
    bottomMargin="1"
  </xsf:printSettings>
  ...
</xsf:view>
```


transform Attribute (importSource Element)

Identifies the [.xslt](#) file that should be used during the merge operation when the source form (the one that is being merged in) matches the [XML Schema](#) specified in the corresponding **schema** attribute.

Type

xsf:xdFileName

Remarks

The **transform** attribute is a required attribute of the [importSource](#) element.

Example

The following is an example of the **transform** attribute as it is used in the **importSource** element:

```
<xsf:importParameters
  enabled="yes"
  <xsf:importSource
    name="My Form"
    schema="MySchema.xsd"
    transform="schematransform.xslt"/>
</xsf:importParameters>
```


transform Attribute (mainpane Element)

Specifies the [relative URL](#) to the [XSL Transformation \(XSLT\)](#) that is used for the view.

Type

xsf:xdFileName

Remarks

The **transform** attribute is a required attribute of the [mainpane](#) element.

Example

The following is an example of the **transform** attribute as it is used in the **mainpane** element:

```
<xsf:views default="View">
  <xsf:view name="View" caption="View">
    <xsf:mainpane transform="view1.xsl"></xsf:mainpane>
    ...
  </xsf:view>
</xsf:views>
```


transform Attribute (useTransform Element)

Specifies the [.xslt](#) file name relative to the [form template](#).

Type

xsf:xdFileName xsf:xdEmptyString

Remarks

The **transform** attribute is a required attribute of the [useTransform](#) element. Its maximum length is 64 characters.

Example

The following example shows the **transform** attribute as it is used in the **useTransform** element:

```
<xsf:documentVersionUpgrade>  
  <xsf:useTransform  
    transform="upgrade.xsl"  
    minVersionToUpgrade="0.0.0.0"  
    maxVersionToUpgrade="1.0.0.5"/>  
</xsf:documentVersionUpgrade>
```


treatBlankValueAsZero Attribute

Specifies whether a blank [field](#) should be calculated with a value of zero.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|------------------------------|---|
| calculations | Contains definitions for calculations performed in the form , and specifies how blank values are handled. |

Definition

`<xsd:attribute name="treatBlankValueAsZero" type="xsd:boolean" use="optional" />`

Remarks

Default is "yes".

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **treatBlankValueAsZero** attribute as it is used in the **calculations** element:

```
<xsf:calculations>
  <xsf:calculatedField
    target="/my:myFields/my:average"
    expression="xdMath:Avg(..my:expenses/my:expense/my:amount)"
    refresh="onChange"/>
    treatBlankValueAsZero="yes"/>
</xsf:calculations>
```


trustLevel Attribute

Specifies the trust level of a form template.

Type

xsf:xdTrustLevel

Parent Elements

| Element | Description |
|--------------------------------|---|
| xDocumentClass | The root element of the form definition file (.xsf) . Contains all other elements and attributes of the form definition file. |

Definition

```
<xsd:attribute name="trustLevel" type="xsf:xdTrustLevel" use="optio
```

Remarks

If the **trustLevel** attribute is present along with the **requireFullTrust** attribute in the form definition file, and the **requireFullTrust** attribute is set to "yes", then the **requireFullTrust** attribute takes precedence. A form template that is not permitted full trust (because it lacks proper certificate, installation, or registration) will fail to load in edit mode, and a warning will be displayed. If the form template is permitted full trust, it will load and run properly in the full trust security zone.

The default value is "domain". If the **trustLevel** attribute is not present, the default value is also "domain".

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **trustLevel** attribute as it is used in the **xDocumentClass** element:

```
<xsf:xDocumentClass
  xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solut
  name="urn:schemas-microsoft-com:myTravelReport"
  author="AuthorName"
  description="Travel Report form for entering travel reports, issues, e
  dataFormSolution="yes"
  solutionVersion = "1.0.0.1"
  productVersion="11.0.5106"
  solutionFormatVersion="1.0.0.0">
  dataFormSolution="yes"
  requireFullTrust="yes"
  trustLevel="restricted"
  trustSetting="manual"
  publishUrl="http://MyServer/InfoPathTemplates/MyTemplate.xsn">
  ...
</xsf:xDocumentClass>
```


trustSetting Attribute

Specifies the trust setting of a form template.

Type

xsf:xdManualAuto

Parent Elements

| Element | Description |
|--------------------------------|---|
| xDocumentClass | The root element of the form definition file (.xsf) . Contains all other elements and attributes of the form definition file. |

Definition

`<xsd:attribute name="trustSetting" type="xsd:string" use="optional"/>`

Remarks

The default value is "manual". If the **trustSetting** attribute is not present, the default value is also "manual".

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **trustSetting** attribute as it is used in the **xDocumentClass** element:

```
<xsf:xDocumentClass
  xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solut
  name="urn:schemas-microsoft-com:myTravelReport"
  author="AuthorName"
  description="Travel Report form for entering travel reports, issues, e
  dataFormSolution="yes"
  solutionVersion = "1.0.0.1"
  productVersion="11.0.5106"
  solutionFormatVersion="1.0.0.0">
  dataFormSolution="yes"
  requireFullTrust="yes"
  trustLevel="restricted"
  trustSetting="manual"
  publishUrl="http://MyServer/InfoPathTemplates/MyTemplate.xsn">
  ...
</xsf:xDocumentClass>
```


type Attribute (editWith Element)

Specifies the type of editing for the fields that match the [XPath](#) expression specified by the [item](#) attribute of the [xmlToEdit](#) element.

Type

xsd:enumeration

Remarks

The **type** attribute is an optional attribute of the [editWith](#) element.

The **type** attribute is used with the xField editing component, and it supports the following values: "plain", "plainMultiline", "formatted", "formattedMultiline", and "rich". The default value is "plain".

Note that if the matched XML [Document Object Model \(DOM\)](#) node does not support the full editing services specified, the effective type is downgraded appropriately (this means downgrading to "plain"). For example, an attribute node will behave as plain even if set to rich. Only data that is in a CDATA section, or that corresponds to XHTML, can support values other than "plain".

Example

The following is an example of the **type** attribute as it is used in the **editWith** element:

```
<xsf:xmlToEdit name="Label_4"  
  item="/CustomUISample/CDCollection/CD/Label">  
  <xsf:editWith type="rich" autoComplete="no"  
    component="xField">  
    </xsf:editWith>  
</xsf:xmlToEdit>
```

type Attribute (errorMessage Element)

Identifies the type of error message to return.

Type

xsd:NMTOKEN

Remarks

The **type** attribute is an optional attribute of the [errorMessage](#) element. Allowed values are "modeless" and "modal". Default value is "modeless".

A dialog box with the long message is returned for "modal" errors. When the dialog box is closed, the field is marked with a dashed red border to indicate that the value is invalid. A user can read the error message by right-clicking the field.

For "modeless" errors, no dialog box is displayed. The field is marked with a dashed red border to indicate that the value is invalid. A user can read the error message by right-clicking the field.

Note If the field is invalid as a result of scripting code or because it was invalid to begin with, but has not been edited, then it will be marked with a red underline.

Example

The following is an example of the **type** attribute as it is used in the **errorMessage** element:

```
<xsf:customValidation>
  <xsf:errorCondition
    match="/exp:expenseReport"
    expressionContext="exp:reportDate"
    expression="msxsl:string-compare(., ../exp:startDate) < 0 and ../exp:
    showErrorOn=".">
    <xsf:errorMessage
      type="modeless"
      shortMessage="The report date occurs before the end of the expe
      The report date occurs before the end of the expense period. Veri:
    </xsf:errorMessage>
  </xsf:errorCondition>
</xsf:customValidation>
```


type Attribute (field Element)

Identifies the standard [XML Schema](#) data type.

Type

xsd:NMTOKEN

Remarks

The **type** attribute is a required attribute of the [field](#) element.

Example

The following is an example of the **type** attribute as it is used in the **field** element:

```
<xsf:listProperties>
  <xsf:fields>
    <xsf:field
      type="xsd:date"
      name="TravelDate"
      columnName="TravelDate"
      required="yes"
      viewable="yes"
      node="TravelReport/Header/travelDate"
      aggregation="first"/>
  </xsf:fields>
</xsf:listProperties>
```

type Attribute (hwsOperation Element)

Specifies the Microsoft BizTalk Server 2004 Human Workflow Services (HWS) operation type.

Type

xsd:string

Parent Elements

| Element | Description |
|------------------------------|--|
| hwsOperation | Defines the HWS operation type, such as adding an action to a new activity flow, adding an action to an existing activity flow, or responding to a task. |

Definition

```
<xsd:attribute name="type" type="xsd:string" use="required"></xsd:a
```

Remarks

The allowed HWS operation types are:

addActionToNewActivityFlow Starts a new workflow.

addActionToActivityFlow Starts a new workflow, or if one exists, extends the workflow.

sendTaskResponse Responds to a task.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **type** attribute as it is used in the **hwsOperation** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval"/>
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate"/>
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response"/>
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response"/>
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond"/>
  </xsf:allowedTasks>
</xsf:hwsWorkflow>

<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID="
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1"/>
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
```

```
        replaceWith="/a:some/b:thing" dataObject="Aux1"/>  
    </xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```


type Attribute (property Element)

Defines the type of the property.

Type

xsd:QName

Remarks

The **type** attribute is a required attribute of the [property](#) element.

All simple [XML Schema](#) data types are allowed.

Example

The following is an example of the **type** attribute as it is used in the **property** element:

```
<xsf:package>
  <xsf:files>
    <xsf:file name="view_1.xsl">
      <xsf:fileProperties>
        <xsf:property
          name="lang"
          type="string"
          value="1033"/>
      </xsf:fileProperties>
    </xsf:file>
  </xsf:files>
</xsf:package>
```


typeID Attribute

Contains the globally unique identifier (GUID) for the Microsoft BizTalk Server 2004 Human Workflow Services (HWS) operation.

Type

xsd:string

Parent Elements

| Element | Description |
|------------------------------|--|
| hwsOperation | Defines the HWS operation type, such as adding an action to a new activity flow, adding an action to an existing activity flow, or responding to a task. |

Definition

```
<xsd:attribute name="typeID" type="xsd:string" use="required"></xsd
```

Remarks

The GUID is automatically generated upon submit of the [hwsAdapter](#) element, and it updates the [processing instructions](#) of the XML instance file.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **typeID** attribute as it is used in the **hwsOperation** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval"/>
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate"/>
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response"/>
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response"/>
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond"/>
  </xsf:allowedTasks>
</xsf:hwsWorkflow>

<xsf:hwsAdapter name="Start Approval"
  wsdlUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?WS
  submitAllowed="yes">
  <xsf:hwsOperation type="addActionToNewActivityFlow" typeID="
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xsf:input source="HWSMessage1.xml">
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1"/>
    <xsf:partFragment match="/ns1:HWSMessage/ActionSection/pa
```

```
        replaceWith="/a:some/b:thing" dataObject="Aux1"/>  
    </xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```


ui Attribute (exportToExcel Element)

Sets whether the user can export the contents of the form to a Microsoft Office Excel 2003 workbook.

Type

xsf:xdEnabledDisabled

Parent Elements

| Element | Description |
|-------------------------------|---|
| exportToExcel | Specifies whether the user can use the form's menus, toolbars, or keyboard shortcuts to export the form to an Microsoft Office Excel 2003 workbook. |

Definition

```
<xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="require
```

Remarks

Setting this attribute to a value other than "enabled" or "disabled" will result in an error message when you attempt to open the form.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **ui** attribute as it is used in the **exportToExcel** element:

```
<xsf:exportToExcel ui="disabled"/>
```


ui Attribute (exportToWeb Element)

Sets whether the user can export the contents of the form to a Web page.

Type

xsf:xdEnabledDisabled

Parent Elements

| Element | Description |
|-----------------------------|--|
| exportToWeb | Specifies whether the user can use the form's menus, toolbars, or keyboard shortcuts to export the form to a Web page. |

Definition

```
<xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="require
```

Remarks

Setting this attribute to a value other than "enabled" or "disabled" will result in an error message when you attempt to open the form.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **ui** attribute as it is used in the **exportToWeb** element:

```
<xsf:exportToWeb ui="disabled"/>
```


ui Attribute (print Element)

Sets whether the user can print the form.

Type

xsf:xdEnabledDisabled

Parent Elements

| Element | Description |
|-----------------------|--|
| print | Specifies whether the user can use the form's menus, toolbars, or keyboard shortcuts to print the form |

Definition

```
<xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="require
```

Remarks

Setting this attribute to a value other than "enabled" or "disabled" will result in an error message when you attempt to open the form.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **ui** attribute as it is used in the **print** element:

```
<xsf:printui="disabled"/>
```


ui Attribute (save Element)

Sets whether the user can save the form.

Type

xsf:xdEnabledDisabled

Parent Elements

| Element | Description |
|----------------------|--|
| save | Specifies whether the user can use the form's menus, toolbars, or keyboard shortcuts to save the form. |

Definition

```
<xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="require
```

Remarks

Setting this attribute to a value other than "enabled" or "disabled" will result in an error message when you attempt to open the form.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **ui** attribute as it is used in the **save** element:

```
<xsf:save ui="disabled"/>
```


ui Attribute (sendMail Element)

Sets whether the user can send the form as an e-mail attachment.

Type

xsf:xdEnabledDisabled

Parent Elements

| Element | Description |
|--------------------------|--|
| sendMail | Specifies whether the user can use the form's menus, toolbars, or keyboard shortcuts to send the form as an e-mail attachment. |

Definition

```
<xsd:attribute name="ui" type="xsf:xdEnabledDisabled" use="require
```

Remarks

Setting this attribute to a value other than "enabled" or "disabled" will result in an error message when you attempt to open the form.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **ui** attribute as it is used in the **sendMail** element:

```
<xsf:sendMail ui="disabled"/>
```


url Attribute

Contains the [Uniform Resource Locator \(URL\)](#) for a Microsoft BizTalk 2004 Human Workflow Services (HWS) Web service.

Type

xsd:string

Parent Elements

| Element | Description |
|--------------------------|--------------------------------------|
| location | The location of the HWS Web service. |

Definition

```
<xsd:attribute name="url" type="xsd:string" use="required"></xsd:att
```

Remarks

The **url** attribute is a required attribute of the **location** element.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **url** attribute as it is used in the **location** element:

```
<xsf:hwsWorkflow taskpaneVisible="yes">
  <xsf:location url="http://www.contoso.com/hwsservice/hwsservice.as
  <xsf:allowedActions>
    <xsf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval"/>
    <xsf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate"/>
  </xsf:allowedActions>
  <xsf:allowedTasks>
    <xsf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response"/>
    <xsf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response"/>
    <xsf:task name="delegateToManager" taskTypeID="420"
      caption="Respond"/>
  </xsf:allowedTasks>
</xsf:hwsWorkflow>
```

useDataSet Attribute

Specifies whether an adapter will support an ADO.NET DataSet.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|-----------------------------------|---|
| webServiceAdapter | Defines a Web service data adapter that retrieves data from a Web service or submits data to a Web service. |

Definition

```
<xsd:attribute name="useDataSet" type="xsd:boolean" use="optional"/>
```

Remarks

Default is "no".

Use of the **useDataSet** attribute requires Microsoft Office InfoPath 2003 Service Pack 1.

An ADO.NET DataSet cannot be used as a query parameter for the **webServiceAdapter** element or any other data adapter.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **useDataSet** attribute as it is used in the **webServiceAdapter** element:

```
<xsf:webServiceAdapter  
  wsdlUrl="http://www.contoso.com/DataSet.asmx?WSDL"  
  queryAllowed="yes"  
  useDataSet="yes">  
  ...  
</xsf:webServiceAdapter>
```


useFilter Attribute

Indicates the user wants a filter widget.

Type

xsd:simpleType

Parent Elements

| Element | Description |
|--------------------------|--|
| editWith | (Optional element) Specifies an instance of an editing component, and provides the corresponding parameters to determine its exact behavior. |

Definition

```
<xsd:attribute name="useFilter" use="optional">  
  <xsd:simpleType>  
    <xsd:restriction base="xsd:string">  
      <xsd:enumeration value="yes"/>  
      <xsd:enumeration value="no"/>  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:attribute>
```

Remarks

The **useFilter** attribute is an optional attribute of the **editWith** element. Its default value is "no". If the user selects the filter widget check box in the property dialog of a [repeating section](#) or [repeating table](#), and the **useFilter** attribute is set to "yes", the [widgetIcon](#) attribute is set to "Filter". The **useFilter** attribute is only applicable for repeating sections and tables.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **useFilter** attribute as it is used in the **editWith** element:

```
<editWith component="xCollection" widgetIcon="filter|standard"  
useFilter="yes|no" filterDependency="xpath1 | xpath2 | xpath3">
```

useScriptHandler Attribute

Specifies whether to use the event handler defined for the [OnMergeRequest](#) event when importing (merging) forms.

Type

xsf:xdYesNo

Parent Elements

| Element | Description |
|----------------------------------|--|
| importParameters | Contains all the parameters that define how the import (merge) forms feature works for the form. |

Definition

```
<xsd:attribute name="useScriptHandler" type="xsf:xdYesNo" use="op
```

Remarks

If you create an event handler for the **OnMergeRequest** event of a form template, you must set the **useScriptHandler** attribute to "yes" before it will run.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **useScriptHandler** attribute as it is used in the **importParameters** element:

```
<xsf:importParameters
  enabled="yes"
  useScriptHandler="yes">
  <xsf:importSource
    name="MySource"
    schema="MySchema.xsd"
    transform="schematransform.xslt"/>
</xsf:importParameters>
```

value Attribute (attachmentFileName Element)

Contains the value of the [attachmentFileName](#) element.

Type

xsd:string

Parent Elements

| Element | Description |
|------------------------------------|--|
| attachmentFileName | Contains the file name of the file attachment to be included with an e-mail message when the form is submitted using the emailAdapter element. |

Definition

```
<xsd:attribute name="value" type="xsd:string" use="required" ></xsd:
```

Remarks

If the **value** attribute is an empty string, the e-mail message will not include a file attachment.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **value** attribute as it is used in the **attachmentFileName** element:

```
<xsf:attachmentFileName value="Status Report" valueType="literal"/>
```

value Attribute (attributeData Element)

Specifies the value of the attribute to be inserted.

Type

xsd:string

Remarks

The **value** attribute is a required attribute of the [attributeData](#) element.

Example

The following is an example of the **value** attribute as it is used in the **attributeData** element:

```
<xsf:editWith component="xOptional">
  <xsf:fragmentToInsert>
    <xsf:chooseFragment parent="report">
      <xsf:attributeData attribute="author" value="author name"/>
    </xsf:chooseFragment>
  </xsf:fragmentToInsert>
</xsf:editWith>
```

value Attribute (bcc Element)

Specifies the value of the **bcc** element, as a literal string or as an expression based on the [valueType](#) attribute.

Type

xsd:string

Parent Elements

| Element | Description |
|---------------------|---|
| bcc | Contains the recipient information for the bcc line of an e-mail message when the form is submitted using the emailAdapter element. |

Definition

```
<xsd:attribute name="value" type="xsd:string" use="required"></xsd:
```

Remarks

Multiple addresses on the bcc line of an e-mail message must be separated by semicolons.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **value** attribute as it is used in the **bcc** element:

```
<xsf:bcc value="someone@example.com" valueType="literal"/>
```

value Attribute (cc Element)

Specifies the value of the **cc** element, as a literal string or as an expression based on the [valueType](#) attribute.

Type

xsd:string

Parent Elements

| Element | Description |
|--------------------|---|
| cc | Contains the recipient information for the cc line of the e-mail message when the form is submitted using the emailAdapter element. |

Definition

```
<xsd:attribute name="value" type="xsd:string" use="required"></xsd:
```

Remarks

Multiple addresses on the cc line of an e-mail message must be separated by semicolons.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **value** attribute as it is used in the **cc** element:

```
<xsf:cc value="my:ccNames" valueType="expression"/>
```


value Attribute (fileName Element)

Specifies the value of the **fileName** element, as a literal string or as an expression based on the [valueType](#) attribute.

Type

xsd:string

Parent Elements

| Element | Description |
|--------------------------|--|
| fileName | Contains the name of the file as a literal string or XPath expression. |

Definition

```
<xsd:attribute name="value" type="xsd:string" use="required"></xsd:
```

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **value** attribute as it is used in the **fileName** element:

```
<xsf:fileName value="my:myFields/my:fileName" valueType="expres
```


value Attribute (folderURL Element)

Specifies the [Uniform Resource Locator \(URL\)](#) of the [Web-based Distributed Authoring and Versioning \(WebDAV\)](#) server or server that is running Microsoft Windows SharePoint Services to which the file is submitted.

Type

xsd:string

Parent Elements

| Element | Description |
|---------------------------|--|
| folderURL | Contains the URL of the server to which the file is submitted. |

Definition

```
<xsd:attribute name="value" type="xsd:string" use="required"></xsd:
```

Remarks

The URL must begin with "http://" or "https://". Other common protocols will cause an error.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **value** attribute as it is used in the **folderURL** element:

```
<folderURL value="http://some_server/some_doc_lib"/>
```

value Attribute (intro Element)

Contains the value of the **intro** element.

Type

xsd:string

Parent Elements

| Element | Description |
|-----------------------|---|
| intro | Contains the introduction for the e-mail message when the form is submitted using the emailAdapter element. |

Definition

```
<xsd:attribute name="value" type="xsd:string" use="required"></xsd:
```

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **value** attribute as it is used in the **intro** element:

```
<xsf:intro value="See below"/>
```


value Attribute (property Element)

For simple properties, specifies a value for the property. For complex and multi-valued properties, the specified value is defined as a container XML tree using an [open content model](#).

Type

xsd:string

Remarks

The **value** attribute is a required attribute of the [property](#) element. The .cab files that are included in an InfoPath form package are listed in the **package** and **files** elements of an .xsf file. A **fileType** value of "ActiveX-CAB" identifies that the file is a .cab file added by the designer and indicates that the file should be managed by the ActiveX support features of the designer.

Example

The following are examples of the **value** attribute as it is used in the **property** element:

```
<xsf:files>
  <xsf:file name="example.cab">
    <xsf:fileProperties>
      <xsf:property name="fileType" type="string" value="ActiveX-
      <xsf:property name="timestamp" type="string" value="xyz" />
    </xsf:fileProperties>
  </xsf:file>
</xsf:files>
</xsf:package>
```

```
<xsf:package>
  <xsf:files>
    <xsf:file name="view_1.xsl">
      <xsf:fileProperties>
        <xsf:property
          name="lang"
          type="string"
          value="1033"/>
      </xsf:fileProperties>
    </xsf:file>
  </xsf:files>
</xsf:package>
```

value Attribute (subject Element)

Contains the value of the **subject** element, either as an expression or as a literal string.

Type

xsd:string

Parent Elements

| Element | Description |
|-------------------------|--|
| subject | Contains the subject line of the e-mail message when the form is submitted using the emailAdapter element. |

Definition

```
<xsd:attribute name="value" type="xsd:string" use="required"></xsd:
```

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **value** attribute as it is used in the **subject** element:

```
<xsf:subject value="My report" valueType="literal"/>
```

value Attribute (to Element)

Specifies the value of the **to** element, as a literal string or as an expression based on the [valueType](#) attribute.

Type

xsd:string

Parent Elements

| Element | Description |
|--------------------|---|
| to | Parent element that contains the "to" line information of the emailAdapter element. |

Definition

```
<xsd:attribute name="value" type="xsd:string" use="required"></xsd:
```

Remarks

Multiple addresses on the "to" line of an e-mail message must be separated by semicolons.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **value** attribute as it is used in the **to** element:

```
<xsf:to value="someone@example.com" valueType="literal"/>
```


valueType Attribute (attachmentFileName Element)

Specifies whether the [value](#) attribute of the [attachmentFileName](#) element should be interpreted as a literal value or an [XPath](#) expression which returns a value.

Type

xsf:xdExpressionLiteral

Parent Elements

| Element | Description |
|------------------------------------|--|
| attachmentFileName | Contains the file name of the file attachment to be included with an e-mail message when the form is submitted using the emailAdapter element. |

Definition

```
<xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="required" />
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **valueType** attribute as it is used in the **attachmentFileName** element:

```
<xsf:attachmentFileName value="Status Report" valueType="literal"/>
```


valueType Attribute (bcc Element)

Specifies whether the [value](#) attribute of the **bcc** element should be interpreted as a literal value or as an [XPath](#) expression that returns a value.

Type

xsf:xdExpressionLiteral

Parent Elements

| Element | Description |
|---------------------|---|
| bcc | Contains the recipient information for the bcc line of an e-mail message when the form is submitted using the emailAdapter element. |

Definition

```
<xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="required" />
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **valueType** attribute as it is used in the **bcc** element:

```
<xsf:bcc value="someone@example.com" valueType="literal"/>
```


valueType Attribute (cc Element)

Specifies whether the [value](#) attribute of the **cc** element should be interpreted as a literal value or as an [XPath](#) expression that returns a value.

Type

xsf:xdExpressionLiteral

Parent Elements

| Element | Description |
|--------------------|--|
| cc | Specifies the value of the cc element, as a literal string or as an expression based on the valueType attribute. |

Definition

```
<xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="required" />
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **valueType** attribute as it is used in the **cc** element:

```
<xsf:cc value="my:ccNames" valueType="expression"/>
```


valueType Attribute (fileName Element)

Specifies whether the value of the **fileName** element should be interpreted as a literal value or as an [XPath](#) expression that returns a value.

Type

xsf:xdExpressionLiteral

Parent Elements

| Element | Description |
|--------------------------|--|
| fileName | Contains the name of the file as a literal string or as an XPath expression. |

Definition

```
<xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="required" />
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **valueType** attribute as it is used in the **fileName** element:

```
<xsf:fileName value="my:myFields/my:fileName" valueType="expre
```


valueType Attribute (subject Element)

Specifies whether the [value](#) attribute of the **subject** element should be interpreted as a literal value or as an [XPath](#) expression that returns a value.

Type

xsf:xdExpressionLiteral

Parent Elements

| Element | Description |
|-------------------------|---|
| subject | Contains the subject line of an e-mail message when the form is submitted using the emailAdapter element. |

Definition

```
<xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="required" />
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **valueType** attribute as it is used in the **subject** element:

```
<xsf:subject value="My report" valueType="literal"/>
```


valueType Attribute (to Element)

Specifies whether the [value](#) attribute of the **to** element should be interpreted as a literal value or as an [XPath](#) expression that returns a value.

Type

xsf:xdExpressionLiteral

Parent Elements

| Element | Description |
|--------------------|---|
| to | Parent element that contains the "to" line information of the emailAdapter element. |

Definition

```
<xsd:attribute name="valueType" type="xsf:xdExpressionLiteral" use="required" />
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **valueType** attribute as it is used in the **to** element:

```
<xsf:to value="someone@example.com" valueType="literal"/>
```


version Attribute

Specifies the ActiveX control version number.

Type

xsd:string

Parent Elements

| Element | Description |
|--------------------------------|---|
| allowedControl | Specifies the ActiveX controls that are allowed to be instantiated. |

Definition

```
<xsd:attribute name="version" type="xsd:string" use="optional"></xsd:attribute>
```

Remarks

If the **clsid** attribute for an ActiveX control is included in the **allowedControl** element in a **form definition file (.xsf)**, but the control is not installed on the user's computer, the control will be installed automatically if the **cabFile** attribute for the control is present and the CAB file is present in the **form template** file (.xsn) or in the same directory as the form definition file. If the control is already installed on the user's computer but is an earlier version than the one listed in the **version** attribute, the install process still will initiate if the CAB file for the control is present. If the CAB file is not present for some reason or the installation fails or is stopped, the form will not open.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **version** attribute as it is used in the **allowedControl** element:

```
<xsf:permissions>
  <xsf:allowedControl
    cabFile="{84F32C01-78D8-4B93-8ED4-106DA70224C2}.cab"
    clsid="{84F32C01-78D8-4B93-8ED4-106DA70224C2}"
    version="1,0,0,1"/>
  <xsf:allowedControl
    clsid="{F08DF954-8592-11D1-B16A-00C0F0283630}"/>
</xsf:permissions>
```


view Attribute

Contains the name of the [view](#) that is switched to as a result of the [rule](#) element action.

Type

xsf:xdViewName

Parent Elements

| Element | Description |
|----------------------------------|-------------------------------|
| switchViewAction | Defines a view switch action. |

Definition

```
<xsd:attribute name="view" type="xsf:xdViewName" use="required">
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **view** attribute as it is used in the **switchViewAction** element:

```
<xsf:rule caption="Receipts" condition="my:expense > 75">
  <xsf:dialogBoxMessageAction>Don't forget your receipts!</xsf:dialogBoxMessageAction>
  <xsf:switchViewAction view="Approval View"/>
  <xsf:assignmentAction targetField="my:group8/my:group9/my:targetField"
    expression="sum(my:expenses/my:expense) * my:taxRate"/>
  <xsf:queryAction adapter="Exchange Rates"/>
  <xsf:submitAction adapter="Exchange Rates"/>
  <xsf:openNewDocumentAction solutionURI="uri:microsoft-Expense1">
  <xsf:closeDocumentAction promptToSaveChanges="yes"/>
  <xsf:dialogBoxExpressionAction>my:group/my:field1</xsf:dialogBoxExpressionAction>
  <xsf:exitRuleSet/>
</xsf:rule>
```

viewable Attribute

Identifies whether the field should be added to the default view.

Type

xsf:xdYesNo

Remarks

The **viewable** attribute is an optional attribute of the [field](#) element. Values include "yes" and "no". The default value is "no".

Example

The following is an example of the **viewable** attribute as it is used in the **field** element:

```
<xsf:listProperties>
  <xsf:fields>
    <xsf:field
      type="xsd:date"
      name="TravelDate"
      columnName="TravelDate"
      required="yes"
      viewable="yes"
      node="TravelReport/Header/travelDate"
      aggregation="first"/>
  </xsf:fields>
</xsf:listProperties>
```


viewController Attribute

Specifies a string that identifies an HTML element in the view.

Type

xsd:string

Remarks

The **viewContext** attribute is an optional attribute of the [xmlToEdit](#) element.

The **viewContext** attribute names an HTML element that has the **xd:CtrlId** attribute. For example, `viewContext="myID"` in the [form definition \(.xsf\) file](#) corresponds to `xd:CtrlId="myID"` in the XSLT/HTML code for the view.

Note If the **xd:CtrlId** attribute is not used in the XSLT/HTML code for the view, then the **viewContext** attribute is ignored.

The **viewContext** attribute can be used to disambiguate controls when two instances of the same control have the same XML context (so that the XML context is identical). An example is a table of contents with chapter titles, and below, the same chapters repeated with full content. Two [editWith](#) elements for `item="chapter"` but different view contexts can specify distinct behavior.

Example

The following is an example of the **viewController** attribute as it is used in the **xmlToEdit** element:

```
<xf:editing>
  <xf:xmlToEdit name="CD_10"
    item="/CustomUISample/CDCollection/CD"
    container="/CustomUISample">
    viewController="cdID"
  <xf:editWith caption="CD"
    xd:autogeneration="template"
    component="xCollection">
    <xf:fragmentToInsert>
      <xf:chooseFragment parent="CDCollection">
        <CD>
          <Title></Title>
          <Artist></Artist>
          <Tracks>
            <Track></Track>
            <Track></Track>
          </Tracks>
        </CD>
      </xf:chooseFragment>
    </xf:fragmentToInsert>
  </xf:editWith>
</xf:xmlToEdit>
</xf:editing>
```


widgetIcon Attribute

Specifies whether a modified icon will be shown for filtered items.

Type

xsd:simpleType

Parent Elements

| Element | Description |
|--------------------------|--|
| editWith | (Optional element) Specifies an instance of an editing component, and provides the corresponding parameters to determine its exact behavior. |

Definition

```
<xsd:attribute name="widgetIcon" use="optional">  
  <xsd:simpleType>  
    <xsd:restriction base="xsd:string">  
      <xsd:enumeration value="standard"/>  
      <xsd:enumeration value="filter"/>  
    </xsd:restriction>  
  </xsd:simpleType>  
</xsd:attribute>
```

Remarks

The **widgetIcon** attribute is an optional attribute of the **editWith** element. Its default value is "standard".

The properties dialog boxes for repeating sections and tables have a **Filter Data** button on the **Display** tab. Below the **Filter Data** button is a check box that allows the user to select whether a modified icon will be shown to indicate filtered items. If the user applies a filter and selects this check box, the **widgetIcon** attribute in the [form definition file \(.xsf\)](#) is set to "filter". The **widgetIcon** attribute is only applicable for repeating sections and tables.

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **widgetIcon** attribute as it is used in the **editWith** element:

```
<editWith component="xCollection"  
widgetIcon="filter|standard" useFilter="yes|no"  
filterDependency="xpath1 | xpath2 | xpath3">
```


wsdlUrl Attribute

Contains the [Uniform Resource Locator \(URL\)](#) of the Web Services Description Language (WSDL) file that describes the Web service specification.

Type

xsd:string

Remarks

The **wsdlUrl** attribute is a required attribute of the [webServiceAdapter](#) element.

Example

The following is an example of the **wsdlUrl** attribute as it is used in the **webServiceAdapter** element:

```
<xf:query>
  <xf:webServiceAdapter
    wsdlUrl="http://localhost/infopathwebservicesample/infopathwebs
    queryAllowed="yes"
    submitAllowed="no">
    <xf:operation
      name="getOrders"
      soapAction="http://tempuri.org/getOrders"
      serviceUrl="http://localhost/infopathwebservicesample/infopathv
      <xf:input
        source="Submit.xml">
      </xf:input>
    </xf:operation>
  </xf:webServiceAdapter>
</xf:query>
```


wSDLUrl Attribute (hwsAdapter Element)

Contains the [Uniform Resource Locator \(URL\)](#) of the Microsoft BizTalk 2004 Human Workflow Services (HWS) Web service.

Type

xsd:string

Parent Elements

| Element | Description |
|----------------------------|--|
| hwsAdapter | Defines the HWS data adapter to start or extend an activity flow or respond to a task. |

Definition

```
<xsd:attribute name="wsdlUrl" type="xsd:string" use="required"></xsd:attribute>
```

Remarks

Note This item is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Editions Service Pack 1 or later is not installed. Any form defined by a form definition file (.xsf) that includes this item cannot be opened in InfoPath when service pack features are disabled or unavailable.

Example

The following is an example of the **wSDLUrl** attribute as it is used in the **hwsAdapter** element:

```
<xf:hwsWorkflow taskpaneVisible="yes">
  <xf:location url="http://www.contoso.com/hwsservice/hwsservice.a
  <xf:allowedActions>
    <xf:action name="approval" actionTypeID="123"
      canInitiateWorkflow="yes" caption="Get Approval"/>
    <xf:action name="delegate" actionTypeID="234"
      canInitiateWorkflow="no" caption="Delegate"/>
  </xf:allowedActions>
  <xf:allowedTasks>
    <xf:task name="getManagerApproval" taskTypeID="435"
      caption="Send Response"/>
    <xf:task name="getVPApproval" taskTypeID="436"
      caption="Send Response"/>
    <xf:task name="delegateToManager" taskTypeID="420"
      caption="Respond"/>
  </xf:allowedTasks>
</xf:hwsWorkflow>

<xf:hwsAdapter name="Start Approval"
  wSDLUrl="http://www.contoso.com/hwsservice/hwsservice.asmx?W!
  submitAllowed="yes">
  <xf:hwsOperation type="addActionToNewActivityFlow" typeID="
  serviceUrl="http://www.contoso.com/hwsservice/hwsservice.asmx
  <xf:input source="HWSMessage1.xml">
    <xf:partFragment match="/ns1:HWSMessage/ActionSection/pa
      replaceWith="/my:myFields/my:param1"/>
    <xf:partFragment match="/ns1:HWSMessage/ActionSection/pa
```

```
        replaceWith="/a:some/b:thing" dataObject="Aux1"/>  
    </xsf:input>  
</xsf:hwsOperation>  
</xsf:hwsAdapter>
```

xmlns Attribute

Defines the [xsf](#) namespace.

Type

xsd:string

Remarks

The **xmlns** attribute is a required attribute of the [xDocumentClass](#) element, and is the standard mechanism for declaring XML namespaces.

Example

The following is an example of the **xmlns** attribute as it is used in the **xDocumentClass** element:

```
<xsf:xDocumentClass
  xmlns:xsf="http://schemas.microsoft.com/office/infopath/2003/solut
  name="urn:microsoft-com:myTravelReport"
  author="AuthorName"
  description="Travel Report form for entering travel reports, issues, e
  dataFormSolution="yes"
  solutionVersion = "1.0.0.1"
  productVersion="11.0.5106"
  solutionFormatVersion="1.0.0.0">
  ...
</xsf:xDocumentClass>
```

xmlToEdit Attribute

Specifies the name of an [xmlToEdit](#) element, for which the button is used.

Type

xsd:NMTOKEN

Remarks

The **xmlToEdit** attribute is an optional attribute of the [button](#) element, but is required for buttons used with editing components.

Example

The following is an example of the **xmlToEdit** attribute as it is used in the **button** element:

```
<xsf:menuArea name="msoInsertMenu">
  <xsf:menu caption="&Section">
    <xsf:button action="xCollection::insert" xmlToEdit="CD_10"
      caption="CD" showIf="always"></xsf:button>
    <xsf:button action="xCollection::insert" xmlToEdit="Track_14"
      caption="Track" showIf="always"></xsf:button>
    <xsf:button action="xOptional::insert" xmlToEdit="Label_16"
      caption="Label"></xsf:button>
  </xsf:menu>
</xsf:menuArea>
```

XDocument Property (SaveEvent Object)

A read-only property that returns a reference to the [XDocument](#) object that is associated with the [SaveEvent](#) object in an [OnSaveRequest](#) event.

expression.**XDocument**

expression Required. Returns a reference to a **SaveEvent** object.

Security Level

2: Can be accessed only by forms running in the same domain as the currently open form, or by forms that have been granted cross-domain permissions.

Remarks

Note This object model member is not supported when the **Disable Service Pack features** option on the **Advanced** tab of the **Options** dialog box in InfoPath is selected or when Microsoft Office 2003 Service Pack 1 or later is not installed. Any form that implements this object model member in its code will generate an error message if it is opened in InfoPath when service pack features are disabled or unavailable.