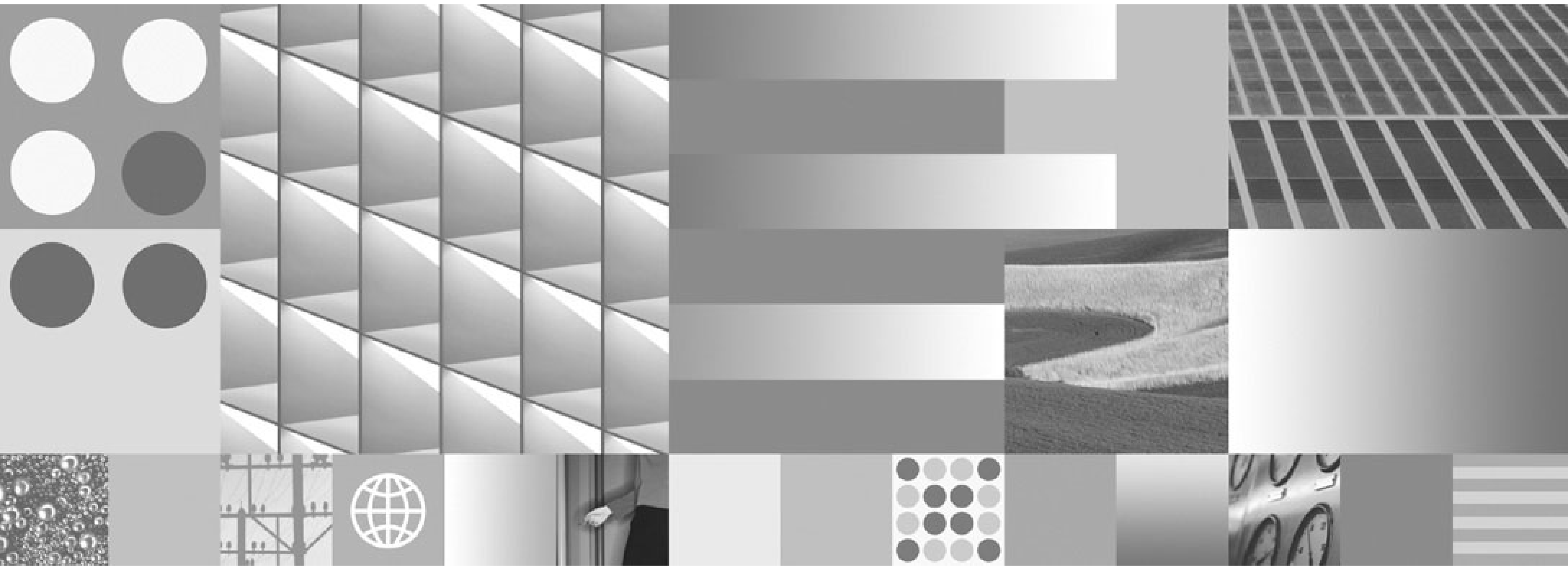
IBM FileNet Forms Manager

Version 5.0.0

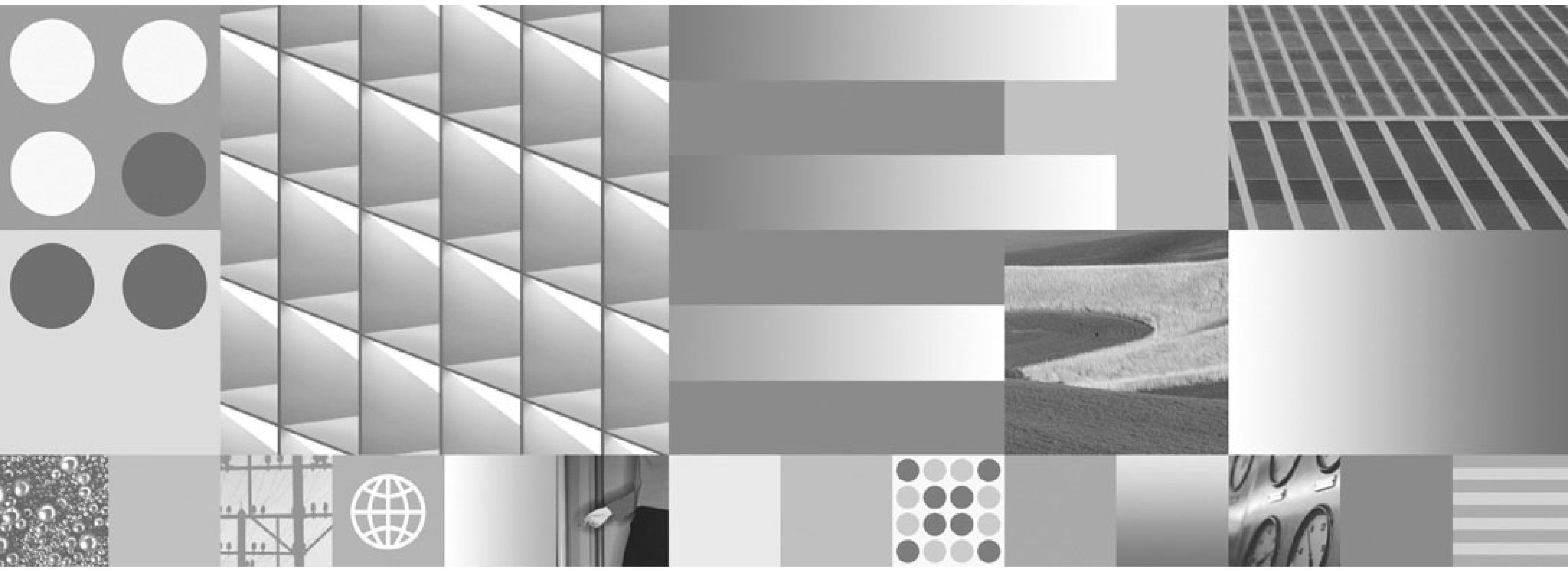


API Syntax

GC31-5499-00

IBM FileNet Forms Manager

Version 5.0.0



API Syntax

GC31-5499-00

Note

Before using this information and the product it supports, read the information in "Notices" on page 38.

This edition applies to version 5.0.0 of IBM FileNet Forms Manager (product number 5724-R89) and to all subsequent releases and modifications until otherwise indicated in new editions.

Table of Contents

Revision Log

Date	Revisions
09/2007	Updated guide to IBM documentation specifications.
11/2006	Added content on using scripts and buttons.
11/2006	Updated copyright information and version number; added revision log.
08/2005	Initial posting.

FileNet Forms Manager API Syntax

FileNet® Forms Manager has three components:

- Forms Manager Server
- eForms Central library
- FileNet eForms

Forms Manager Server is comprised of the files that are stored on the web server and the services behind the eForms Central library. Services can include auto-incrementing of numbers on forms created from templates, form tracking, and database lookups (e.g., country codes, industry or company-specific data such as part numbers).

The eForms Central library is a repository of electronic files that are available from your organization’s web server.

FileNet eForms is comprised of HTML eForms (used for filling out a form in a web browser) and Desktop eForms (used for filling out forms on the desktop).

The API syntax for the Forms Manager Server is described in this document. Detailed examples have been provided for your convenience at the end of this chapter.

About this Document

This document is intended for application integrators who can understand and write XML (for creating custom toolbars), and have experience using middleware scripting environments such as ASP or Cold Fusion (for using custom data sources). The following describes some cases for when you want to use this document:

Forms Manager Server Internal Use

Users are presented with forms containing customized toolbars. These toolbars are embedded within the definition of the ITX Form Template and can include custom names and actions. For example, a Purchase Order Form may require a toolbar with a button named “Email” instead of the standard “Send” and a custom button that goes to a specific URL link. The Toolbar Sources API syntax shown on pages 5-15 would apply to this task.

Web Applications

A web developer is creating a web application that includes a form. Form presentation and data is externally stored and controlled with external HTML pages and databases. The form is called on demand and filled with prefill or existing data. The Template Sources, Toolbar Sources, and Data Sources API syntax shown on pages 5-15 would apply to this task.

API Overview

An HTML eForm is rendered by a request to Form.aspx. The target of this request can be a document window, a frame, or an iframe. This request requires some or all of the following parameters:

- template: Describes the optional template source used to open the form.

- data: Describes the optional data sources used to open the form.
- toolbar: Describes the optional toolbar sources that are used to alter the appearance and describes the functionality of the toolbars presented when the form opens.
- title: The optional title used when the form is opened. The title appears as the title of the form window and as the default title in the “Save As” dialog box when the user saves the form.
- state: The initial state of the form which can be set to new (“0”) or existing (“1”).

Each of these can be specified as query string or post parameters. If one of the parameters references another parameter that contains a large data structure, the referenced parameter must be passed as a post parameter.

Form.aspx is located in **Program Files > FileNet > Forms Manager > [Name of eForms Central library]**. This page can be referenced in a url, e.g., http://libraryname/form.aspx?template=local,Expense2003.

Template Sources

A template source provides the template document that contains the graphic and intelligence components of a form. Only a single template source can be specified. The template source must be indicated explicitly. The following data source type is supported:

Data Source Type and Syntax	Description
`local,`***`templateReference`***	The form template is obtained from a template document stored in the eForms Central library with the template ID ***templateReference*** and the document type “ITX”. Use the template’s template ID as the template reference. **NOTE** For performance reasons, parsed templates are cached at the server and are referenced internally by both the template ID and a “last modified” timestamp.

Examples

```
Form.aspx?template=local,Expenses2003
```

This example specifies that a template with the template ID “Expenses2003” is to be opened from the eForms Central library.

Toolbar Sources

A toolbar source provides a toolbar configuration to be applied to the existing collection of toolbars and buttons that are to be displayed on a form. A toolbar configuration describes the toolbar configuration commands to be applied to the existing collection of toolbars and buttons. A form can have multiple or no toolbar sources.

Toolbar configurations are provided by the following:

- Built-in toolbar configurations.

- template and data sources.
- Toolbar source parameter.

Toolbar configurations are applied in the following order:

- HTML eForms built-in toolbar configuration.
- Forms Manager Server built-in toolbar configuration.
- Template source toolbar configuration.
- Toolbar configuration from each data source in reverse order (the primary data source is merged last).
- Toolbar configurations from the toolbar parameter in standard order.

HTML eForms provides the following empty built-in toolbars:

- **app** Forms Manager Server toolbar configuration extends the app toolbar by inserting the buttons needed to save, send, and track the form.
- **form** HTML eForms toolbar configuration extends the form toolbar by inserting the buttons needed to generate a PDF version of the form, and display Help.
- **script** Reserved for later use.
- **custom** By default, custom toolbar buttons with embedded scripts are inserted into the custom toolbar.

Toolbar Source Types

The `url`, `form`, `file`, and `inline` toolbar source types are supported. All toolbar source types except `inline` reference a toolbar configuration XML document that describes a series of toolbar configuration commands to be applied to the existing toolbars (see “Toolbar Configuration” on page 13). The inline toolbar source type defines the toolbar configuration commands within the parameter. See “Toolbar Configuration Structure” on page 13 for a complete description of each command.

Arguments that can repeat zero or more times are displayed using brace ({}) symbols. Optional arguments are displayed using bracket ([ ]) symbols. See also “Syntax Considerations” on page 26.

Toolbar Source Type and Syntax	**Description**
`url,`***`url`***	The toolbar configuration is obtained from the web site at the specified url.
`form,`***`param`***	The toolbar configuration is obtained from the HTTP post parameter specified by ***param***. The request must be an HTTP post request.
`file,`***`param`***	The toolbar configuration is obtained from the HTTP post parameter specified by ***param***. The request must be an HTTP post request with enctype=”multipart/form-data”.
`inline,`***`command`***`{,`***`command`***`}`	The toolbar configuration is specified inline as a list of toolbar configuration ***commands***. See “Inline Toolbar Source Commands” on page 11 for a description of the commands and their syntax.

Examples

Url

Form.aspx?template=local,WCExpenses&data=local,0&toolbar=url,http\://hostname/ClearToolbar.xml

This example specifies that the WCExpenses form is to be opened with a url based toolbar source.

Form

Form.aspx?data=form,dat,0,New&toolbar=form,tb

This example specifies that a new form is to be opened with the title `New` from the post parameter `dat`. The toolbar configuration obtained from the post parameter `tb` is applied.

Inline Toolbar Source Commands

The following toolbar commands are supported by the inline toolbar source type. See “Toolbar Configuration Structure” on page 13 for a complete description of each command.

Command and Syntax	**Description**
`add,`***`button`***`,[`***`text`***`],[`***`icon`***`],[`***`action`***`]`	Adds a button to the specified toolbar. A ***`button`*** argument in the form***`toolbarID:buttonID`*** specifies both the toolbar and the button. A button argument in the form ***`buttonID`*** specifies the default toolbar. See "Toolbar Configuration Structure" on page 13 for more information about toolbar and button IDs. The ***`text`*** argument must take the form ***`language:label:toolTip{:language:label:toolTip}`***. See "Toolbar Configuration" on page 13 for a description of toolbar button text. The tool tip value can be omitted as follows: `en:Send:null` or `en:Send:`. If the ***`icon`*** argument is specified in the form ***`id:iconID`***, the built-in icon with icon ID ***`iconID`*** is displayed. If the ***`icon`*** argument is specified in the form ***`url:address`***, the icon is obtained from the specified web address. If the ***`icon`*** argument is specified as ***none***, no icon is used. If the ***`icon`*** argument is not specified, the default icon for the action is used. See "" on page 18 for more information. The ***`action`*** argument specifies the built-in action performed when the user clicks the button. It takes the form ***`actionID{:param}`***. The text, icon, and action arguments may be optional. See "Toolbar Button Actions" on page 17 for more information.
`mod,`***`button`***`,[`***`text`***`],[`***`icon`***`],[`***`action`***`]`	Modify the specified button in the specified toolbar as specified by the ***text***, ***icon***, and ***action*** parameters. See the add command above for a description of each argument. See "Toolbar Configuration Structure" on page 13 for more information
`rem,`***`buttonID`***	Remove the specified button, which may be of any type, from the toolbar in which it is found. See "Toolbar Configuration Structure" on page 13 for more information.
`clr,[`***`toolbarID`***`]`	Remove all toolbar buttons from the specified toolbar(s). If the ***toolbar*** argument is omitted or specified as ***all***, then remove all toolbar buttons from all toolbars. See "Toolbar Configuration Structure" on page 13 for more information.

Examples

Inline

Form.aspx?template=local,Test&data=local,47&toolbar=inline,clr,all,add,submit

This example specifies that a local form is to be opened with an inline toolbar source that clears all the toolbars and inserts a single button using the well-known button ID `submit`.

Form

Form.aspx?template=local,Test&data=form,dat&toolbar=form,tb&title=New&state=0

This example opens a new data document from the template with the template ID “Test” and title “New” from the form parameter “dat”. It merges the toolbar configuration obtained from the form parameter “tb”.

Toolbar Configuration

A toolbar configuration describes a series of commands to be applied to the existing toolbars when the form opens. A toolbar configuration can be stored with a template or a form. It can also be included as a Form.aspx parameter when the request is made to render a form. See “Toolbar Configuration Commands” on page 10 for a description of the commands that can be applied to the existing toolbars. Also see Appendix A, “Toolbar Configuration Schema” for information on how to validate the XML code for your toolbar.

Toolbar Configuration Structure

The toolbar configuration structures for the URL, Form, and File types are based on XML. Built-in toolbar configurations are provided with Forms Manager Server. Toolbar configurations must appear in the top-level element of a template, between the <revisionInfo> and the <dataModel> elements. Toolbar configurations can also appear Inline as HTTP post parameters referenced by the `toolbar` parameter of the Form.aspx request.

The following example illustrates how the default toolbars are generated.

```
<toolbarConfig>
    <add buttonID =”submit”/>
    <add buttonID =”pdf”/>
    <add buttonID =”help”/>
    <add buttonID =”save”/>
    <add buttonID =”saveas”/>
    <add buttonID =”send”/>
    <add buttonID =”track”/>
</toolbarConfig>
```

In addition to specifying whole toolbars, a toolbar configuration can override or remove parts of a previously configured toolbar. Thus, the template configuration can override or remove parts of the built-in configurations, and so on.

For example, here is a configuration that could be used in a template to remove the **PDF** button:

```
<toolbarConfig>
    <remove buttonID=”pdf” />
</toolbarConfig>
```

The following configuration clears all buttons from the form toolbar, removes the PDF button, replaces the label and tool tip of the **Save Document** button, and adds a new toolbar with a **Privacy Policy** button that uses the Get action to open the appropriate web page. It also uses the url icon source for the **WorldCorp Home** button.

```
<toolbarConfig>
    <clear toolbarID=”form”/>
    <remove buttonID=”pdf”/>
    <modify buttonID=”save”>
        <text language=”en” label=”Save Document” toolTip=”Save the document” />
    </modify>
    <add toolbarID=”policies” buttonID=”privacy”>
        <text language=”en” label=”Privacy Policy” toolTip=”View Privacy Policy” />
        <icon type=”id” src=”get” />
        <action actionID=”get”>
            <param>http://www.worldcorp.com/Policies.htm</param>
        </action>
    </add>
    <add toolbarID=”policies” buttonID=”worldcorp”>
        <text language=”en” label=”WorldCorp Home” toolTip=”WorldCorp home page” />
        <icon type=”url” src=”http://www.worldcorp.com/worldcorp.gif” />
        <action actionID=”custom”>
            <param name=”url” type=”string”>
            <![CDATA[
                window.open(”http://www.worldcorp.com”);
            ]]>
            </param>
        </action>
    </add>
</toolbarConfig>
```

Toolbar Configuration Commands

You can use the following commands to configure your toolbars.

toolbarConfig

This is the root element of a toolbar configuration.

Attributes: NONE

Elements: add, modify, remove, clear

add

The button is added to the specified toolbar. All button settings that are not explicitly set assume default values where possible. An exception is thrown if default values cannot be assumed (e.g., the url for a post action is not specified). If no toolbar is specified, the button is added to its default toolbar. Because the button ID must be unique, any other button with a matching ID in any toolbar is removed. If the specified toolbar does not exist, it is appended to the list of existing toolbars.

Attributes:

Attribute	Value	Description
toolbarID	String	The ID of the toolbar to which the button will be added.
buttonID	String	The ID of the button to be added.

Elements: text, icon, action

The text, icon, and action elements may be optional. See “Toolbar Button Actions” on page 17 for more information. If one or more text elements are provided, the language that best matches the browser settings is selected when the button is displayed.

`e.g., <add toolbarID ="custom" buttonID="send" />`

modify

The button is modified according to the explicit settings specified. All button settings not explicitly specified are left as before. If the button is found in a different toolbar, it is moved to the specified toolbar. If no toolbar is specified, the button is modified in the toolbar in which it is found. If the button is not found, it is created in the specified or default toolbar with default settings as required. An exception is thrown if default settings cannot be assumed. If the specified toolbar does not exist, it is appended to the list of existing toolbars.

Attributes:

Attribute	Value	Description
toolbarID	String	The ID of the toolbar in which the button will be modified.
buttonID	String	The ID of the button to be modified.

Elements: text, icon, action

The text, icon, and action elements may be optional. See "Toolbar Button Actions" on page 17 for more information. If one or more text elements are provided, all the original text is discarded and replaced with the new text. If no text element is specified, no changes are made to the original text.

```
e.g., <modify toolbarID="app" buttonID="SendToServer"/>
```

remove

The button is removed from the toolbar in which it is found. If the button is not found, no action is performed.

Attributes:

Attribute	Value	Description
buttonID	String	The ID of the button to be removed.

Elements: None

```
e.g., <remove buttonID="save" />
```

clear

All buttons are removed from the specified toolbar. If the specified toolbar does not exist, no action is performed. If no toolbar is specified, all buttons are removed from all toolbars.

Attributes:

Attribute	**Value**	**Description**
toolbarID	String	The ID of the toolbar from which all buttons will be removed. If this attribute is “all” or omitted, all toolbars are cleared.

Elements: None

```
e.g., <clear toolbarID="form" />
```

text

The text tag provides localized text for the toolbar button in a particular language. As many text tags as requested can be added. Based on browser settings, the best language is selected and displayed on the button. See “” on page 18 for more information.

Attributes:

Attribute	**Value**	**Description**
language	String	The language of the localized text, e.g., `en-US`.
label	String	The localized label for the button.
toolTip	String	The localized tool tip for the button. If the tool tip is not specified, the button label is used.

Elements: None

```
e.g.,<add toolbarID=”resources” buttonID=”worldcorp”>
        <text language=”en” label=”WorldCorp Home” toolTip=”WorldCorp home page” />
    </add>
```

icon

The icon tag provides the icon for the toolbar button. See “” on page 18 for more information and review the example at the end of this section.

Attributes:

Attribute	**Value**	**Description**
type	String	If `type="id"`, the built-in icon with matching ID is obtained internally. If `type="url"`, the icon is obtained from the specified url. If `type="none"`, no icon is used.
src	String	If `type="id"`, specifies the icon ID. If `type="url"`, specifies the url. If `type="none"`, no icon is used.

Elements: None

```
e.g., <add toolbarID="resources" buttonID="worldcorp">
        <text language="en" label="WorldCorp Home" toolTip="WorldCorp home page" />
        <icon type="url" src="http://www.worldcorp.com/worldcorp.gif" />
    </add>
```

action

The action tag describes the action associated with an action type button. The action is invoked when the button is clicked. This tag is ignored for all other types of buttons.

Attributes:

Attribute	Value	Description
actionID	String	The ID of the action to be performed. See "Toolbar Button Actions" on page 17 for more information.

Elements: param

```
e.g., <add toolbarID="resources" buttonID="worldcorp">
        <text language="en" label="WorldCorp Home" toolTip="WorldCorp home page" />
        <icon type="url" src="http://www.worldcorp.com/worldcorp.gif" />
        <action actionID="custom">
        </action>
    </add>
```

param

The param tag provides a parameter for the action.

Attributes: None

Elements: The parameter value. Proper encoding is required. For example, the JavaScript™ source code specified by the source parameter of the custom action can be contained in a CDATA section. See the following section on "Toolbar Button Actions" for more information.

```
e.g., <add toolbarID="resources" buttonID="worldcorp">
        <text language="en" label="WorldCorp Home" toolTip="WorldCorp home page" />
        <icon type="url" src="http://www.worldcorp.com/worldcorp.gif" />
        <action actionID="custom">
            <param>
                <![CDATA[
                window.open("http://www.worldcorp.com");
                ]]>
            </param>
        </action>
    </add>
```

Toolbar Button Actions

Each button requires a button ID, toolbar ID, text, icon, and an action. An action is the operation performed when a button is clicked. Some actions require parameters. If the action ID is not explicitly provided and the button ID is equivalent to the action ID of an action that requires no parameters, the action ID is implied by the button ID. If the toolbar ID, text, or icon value is not provided, the default value is implied by the action ID. Default text is not provided for all actions.

The following table lists all actions that are supported and their default values.

Action ID	**Action Parameters**	**Default Toolbar**	**Default English Text**	**Default Icon**	**Description**
save	N/A	app	Save	💾	Resaves the form to the eForms Central library. Behaves like `saveas` if the form was not previously saved.
saveas	N/A	app	Save As	💾	Saves the form with a new title or location into the eForms Central library.
send	N/A	app	Send	✉	Sends the form to another user.
track	N/A	app	Track	📊	Displays the form’s tracking status.
submit	N/A	form	Submit	📤	Submits the form as configured in eForms Designer.
pdf	N/A	form	PDF	📄	Renders a PDF version of the form in another window.
help	N/A	form	Help	❓	Displays the Help document associated with this form as configured in eForms Designer.
get	url	form	N/A	🌐	Displays the page at the specified url in a new window.
post	url	form	N/A	📤	Submits the form data formatted as “text/xml” to the specified url. **Note:** anonymous access is required for the url.
custom	source	custom	N/A	⚙	Executes the JavaScript specified by the source parameter.

Toolbar Rendering

Toolbar buttons are rendered on the server on demand using the specified icon and the label whose language best matches the languages supported by the browser. The button can be rendered in enabled, disabled, and rollover states using a standard button height and font. Rendered buttons are cached at the server for future use.

Toolbars without buttons are not rendered.

The icon can be specified in the toolbar configuration as a built-in icon ID or as a url to an external image file. The following built-in icon identifiers are supported:

- save
- saveas
- send
- track
- submit
- pdf
- help
- get
- post
- custom

```
e.g., <modify buttonID="NewToolbarButton">
        <text language="en-US" label="Mail" toolTip="Send to Claims Dept." />
        <icon type="id" src="send" />
      </modify>
```

Data Sources

A data source provides the form data that is used when the form is opened. A form can have multiple or no data sources. The first data source is called the “primary” data source. The data from all other specified sources is merged, in order, into the primary data source to provide the data that is used to pre-fill the form when it is opened. (See “Data Source Merging” on page 22.) The primary data source also provides the following as required:

- Template ID
- Form title
- Form is new or existing
- Data ID (local data source only)

Data Source Types

The following data source types are supported.

Arguments that can repeat zero or more times are displayed using brace ({}) symbols. Optional arguments are shown in brackets ([ ]). See also "Syntax Considerations" on page 26.

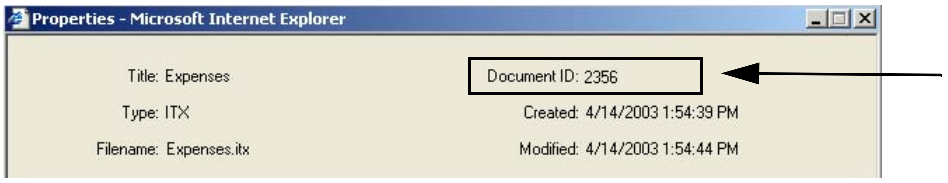
Data Source Type and Syntax	Description
`local`,***`dataID`***	The form data is obtained from a form stored in the library with the data ID***dataID***. The form is opened as an existing data document with the title stored in the eForms Central library. If the data ID is 0, the user's personal information is opened as a new form with the title "Untitled." The document title is obtained internally from the data document. The user's personal information is treated as a new data document. All other form data documents are treated as existing form data documents.
`url`,***`url`***	The form data is obtained from a website at the specified ***url***.
`form`,***`param`***	The form data is obtained from the HTTP post parameter specified by ***param***. The request must be an HTTP POST request.
`file`,***`param`***	The form data is obtained from the HTTP post parameter specified by ***param***. The request must be an HTTP POST request with enctype="multipart/form-data."
`inline`, [{,***`name`***,***`value`***}]	The form data is specified inline as a list of name-value pairs where ***name*** and ***value*** represent the cell name and cell value respectively. This type of data source cannot be used to pre-fill table cells.
`cache`,***`cacheID`***	The form data is obtained from a server cache instance with the specified ***`cacheID`***. This type of data source may only be used internally by the Forms Manager Server.

Examples

Local

```
Form.aspx?template=local,Sample&data=local,0;local,2356&title=Sample&state=1
```

This example specifies that a data document with the user's personal information from the local template with template ID "Sample" is to be opened and merged with a local data document with document ID 2356. Personal information can be excluded by omitting "local,0". The document is opened as an existing document with the title "Sample."



Title: Expenses
Document ID: 2356
Type: ITX
Created: 4/14/2003 1:54:39 PM
Filename: Expenses.itx
Modified: 4/14/2003 1:54:44 PM

```
Form.aspx?template=local,Sample&data=local,0
```

This example opens a new data document that's pre-filled with the user's personal information from the local template with template ID "Sample." The document is opened as a new form instance with the title "Untitled."

Url

```
Form.aspx?template=local,WCExpenses&data=url,http\://hostname/Expenses.xml&title=Expenses&state=0
```

This example specifies that a form is to be opened with data obtained from a URL data source.

Form

```
Form.aspx?template=local,Sample&data=form,formData
```

This example specifies that a form is to be opened from the data provided by the form parameter called "formData" from the local template with template ID "Sample". The template source must be provided explicitly.

Inline

```
Form.aspx?template=local,User&data=inline,FName,Bob,LName,Smith,Age,42
```

This example specifies that a form is to be opened using the local template with the template ID "User" pre-filled with the specified inline data. The form will be opened as a new data document with the default title "Untitled."

NOTES

- Detailed examples are provided in the last section of this document.
- Inline data sources containing time formatted values must be shown with the escape character '\' before the colon, e.g., 11:30 should be entered as data=inline,StartTime,11\:30

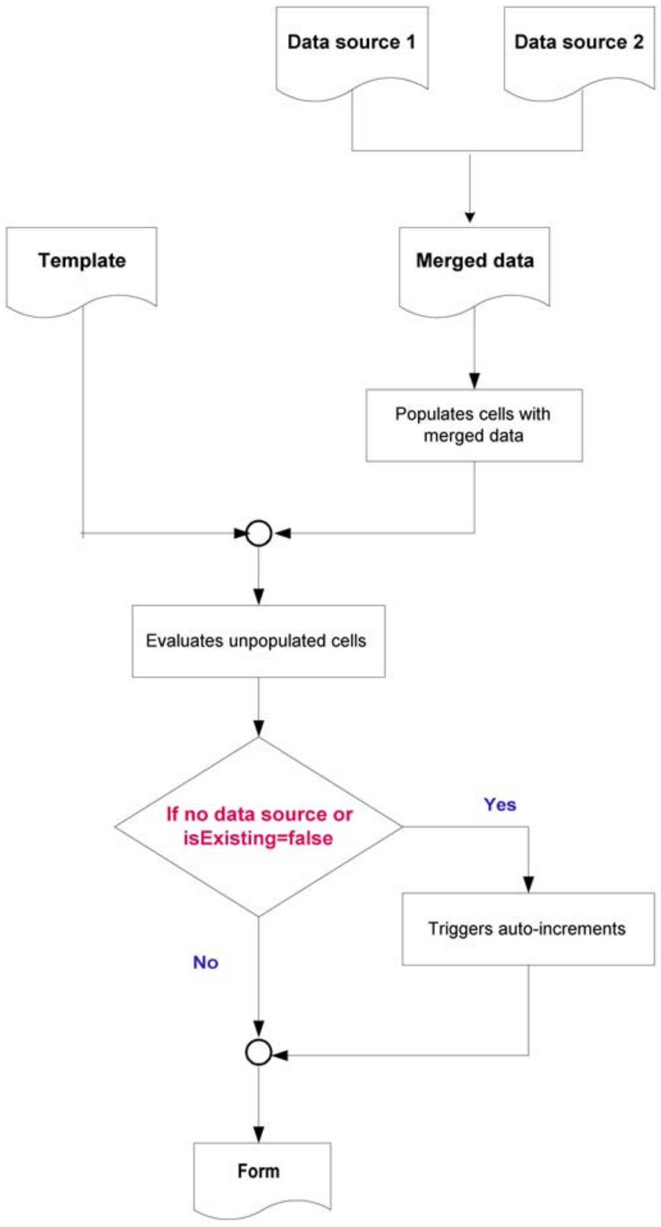
What Happens When a Form is Opened

The following steps take place when a form is opened:

- If one or more data sources are used to open a form, they are merged into a single instance of data. See "Data Source Merging" on page 18 for more information about how data is merged.
- Any cells explicitly specified in the merged data are populated. This includes cells whose values are empty.
- Any cells defined by the template whose values are not explicitly populated are evaluated according to the configuration of the template.
- If no data source is specified, auto-increments and startup scripts are triggered.

NOTE Scripts are not supported with this implementation. However, script functionality is reserved for a future release.

The following diagram illustrates this process.


Data source 1
Data source 2
Template
Merged data
Populates cells with merged data
Evaluates unpopulated cells
If no data source or isExisting=false
Yes
No
Triggers auto-increments
Form

Data Source Merging

A form can be opened with data that is obtained from more than one data source (can be different data source types). The data from all data sources is automatically merged, in order, into a single data instance, starting with the primary data source. As each data source is merged, the previously merged data is preserved.

This table illustrates how data is merged.

Data Source 1	Data Source 2	Merged Data
`Cell1=` “`Edmonton` “		`Cell1=` “`Edmonton` “
`Cell2=` “ “	`Cell2=` “`Alberta` “	`Cell2=` “ “
	`Cell3=` “`Canada` “	`Cell3=` “`Canada` “
`Cell4=` “`North America` “	`Cell4=` “`Europe` “	`Cell4=` “`North America` “

In the following example, all the data from the primary data source is preserved and only the data from `cell4` and `table2` are merged.

NOTE Although the secondary data source has more rows for `table1`, these rows aren’t included in the merged data source.

Primary Data Source

```
<xml version=”1.0” ?>
<xform>
  <model>
    <string name=”cell1”/>
    <string name=”cell2”/>
    <string name=”cell3”/>
      <group name=”table1” maxOccurs=”*”>
        <number name=”Qty”/>
        <number name=”Price”/>
        <string name=”Description”/>
      </group>
  </model>
  <instance index=”1”>
    <cell1>Value1</cell1>
    <cell2><cell2>
    <cell3>Value3</cell3>
    <table1 index=”1”>
      <Qty>10</Qty>
      <Price>123</Price>
      <Description>Testing stuff</Description>
    </table1>
    <table1 index=”2”>
      <Qty>101</Qty>
      <Price>45</Price>
      <Description>Testing stuff</Description>
    </table1>
  </instance>
</xform>
```

Secondary Data Source

```
<xml version=”1.0” ?>
<xform>
  <model>
    <string name=”cell1”/>
    <string name=”cell2”/>
    <string name=”cell3”/>
    <string name=”cell4”/>
      <group name=”table1” maxOccurs=”*”>
```

```
         <number name=”Qty”/>
         <number name=”Price”/>
         <string name=”Description”/>
      </group>
      <group name=”table2” maxOccurs=”*”>
         <number name=”Date”/>
         <string name=”OrderNumber”/>
         <string name=”Description”/>
      </group>
   </model>
  <instance index=”1”>
    <cell1>ValueSec1</cell1>
    <cell2>ValueSec2<cell2>
    <cell3>ValueSec3</cell3>
    <cell4>ValueSec4</cell4>
    <table1 index=”1”>
       <Qty>10</Qty>
       <Price>123</Price>
       <Description>Testing stuff Secondary row 1</Description>
    </table1>
    <table1 index=”2”>
       <Qty>101</Qty>
       <Price>45</Price>
       <Description>Testing stuff Secondary row 2</Description>
    </table1>
    <table1 index=”3”>
       <Qty>101</Qty>
       <Price>45</Price>
       <Description>Testing stuff secondary row 3</Description>
    </table1>
    <table1 index=”4”>
       <Qty>101</Qty>
       <Price>45</Price>
       <Description>Testing stuff secondary row 4</Description>
    </table1>
    <table1 index=”5”>
       <Qty>101</Qty>
       <Price>45</Price>
       <Description>Testing stuff secondary row 5</Description>
    </table1>
    <table2 index=”1”>
       <Date>2003-02-15</Date>
       <OrderNumber>45New01</OrderNumber>
       <Description>Table2 from secondary row 1</Description>
    </table1>
  </instance>
</xform>
```

Merged Data

```
<xml version=”1.0” ?>
<xform>
  <model>
    <string name=”cell1”/>
    <string name=”cell2”/>
    <string name=”cell3”/>
    <string name=”cell4”/>
    <group name=”table1” maxOccurs=”*”>
       <number name=”Qty”/>
       <number name=”Price”/>
       <number name=”Description”/>
```

```
      </group>
      <group name=”table2” maxOccurs=”*”>
         <number name=”Date”/>
         <string name=”OrderNumber”/>
         <string name=”Description”/>
      </group>
   </model>
   <instance index=”1”>
      <cell1>Value1</cell1>
      <cell2><cell2>
      <cell3>Value3</cell3>
      <cell4>ValueSec4</cell4>
      <table1 index=”1”>
         <Qty>10</Qty>
         <Price>123</Price>
         <Description>Testing stuff</Description>
      </table1>
      <table1 index=”2”>
         <Qty>101</Qty>
         <Price>45</Price>
         <Description>Testing stuff</Description>
      </table1>
      <table2 index=”1”>
         <Date>2003-02-15</Date>
         <OrderNumber>45New01</OrderNumber>
         <Description>Table2 from secondary row 1</Description>
      </table1>
   </instance>
</xform>
```

Mapping Forms Manager Data Types to XML Data

The following table lists all Forms Manager data types with their matching XML data types. This is useful especially when generating prefill data content from a database.

Forms Manager Data Type	XML Data Type
Character	String
Number	Number
Name	String
Date	Date
Time	Time
Boolean	Boolean
Picture	Binary
Signature	Binary

Syntax Considerations

ID Values

To be valid, the values for button IDs and toolbar IDs must be upper or lowercase alpha characters.

Delimiters

Multiple template, data, and toolbar sources are separated by semi-colons (;). Each source contains multiple arguments separated by commas (,). Some arguments contain multiple values separated by colons (:).

For example, the toolbar parameter of the following url contains two toolbar sources. The first toolbar source contains two arguments. The second toolbar source contains four arguments. The fourth argument of the second toolbar source contains three values.

Form.aspx?template=local,Sample&toolbar=form,tb1;inline,mod,app:save,en-US:Resave:Resave

Optional Arguments

Unused optional arguments can be replaced with empty strings or with the reserved word `null`. Optional arguments at the end of the set of arguments may be omitted. For example, the following urls are equivalent.

Form.aspx?template=local,Test&data=local,47&toolbar=inline,add,submit,,id:submit

Form.aspx?template=local,Test&data=local,47&toolbar=inline,add,submit,null,id:submit,null

Escape Characters and Url-Encoding

Because they are used as delimiters, all semicolons, commas, colons, and backslash characters that occur in any argument must be escaped using the backslash character. Also, if parameters are provided via the query string, they must be url-encoded.

In the following example, the query string is used to open a form with a local template source, the template ID `T a;b,c:d\e`, and the url data source **http://forms.com/form.asp?id=23.** Each parameter must be escaped as follows.

local,T a\;b\,c\:d\\e

url,http\://www.forms.com/form.asp?id=23

Then each parameter must be url-encoded for the query string:

Form.aspx?template=local%2CT%20a%5C%3Bb%5C%2Cc%5C%3Ad%5C%5Ce&data=url%2Chttp%5C%3A//www.forms.com/form.asp%3Fid%3D23

This form of query string is difficult to read and reproduce. A partially-encoded url such as the following is sufficient for most browsers.

Form.aspx?template=local,T%20a\;b\,c\:d\\e&data=url,http\://www.forms.com/form.asp%3Fid%3D23

Examples

This section contains examples of when and how to use the Forms Manager API syntax for Data Source and Toolbar Source types and simple toolbar configuration.

Before You Begin

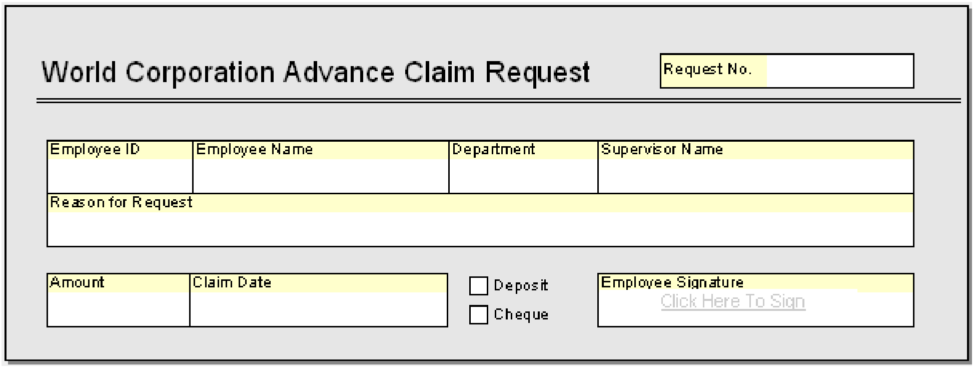
To conveniently generate XML code based on your Desktop Form Template:

1. Open the template in Desktop eForms.
2. Choose **File > Save As.** The Save Data Document As dialog box appears.
3. Choose “XML File (*.xml)” from the ‘Save as type’ drop-down list.
4. Enter a file name and then save it.
5. View the code in any HTML editor program. Do not close this page.
6. Open a new blank page in the HTML editor. Copy and paste the relevant lines of code containing the necessary pre-fill fields into this blank page.

   Using these steps save you time in typing much of the XML code generated from the template.

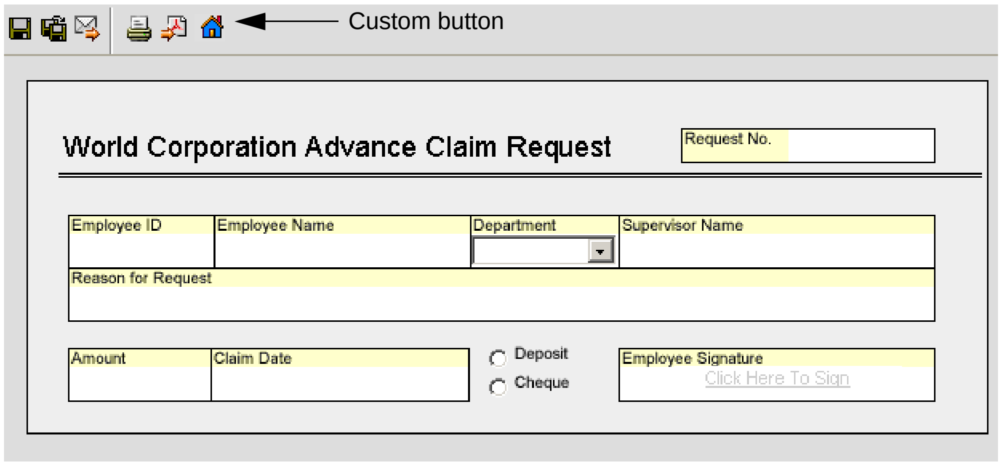
Embed Custom Toolbar Only

In this example, the form requires a custom toolbar that includes a custom “Home” button. Here is the sample template drawn by a form author using eForms Designer, and the form as it appears with the embedded toolbar when opened by the user for the first time.


World Corporation Advance Claim Request

Request No.			
Employee ID	Employee Name	Department	Supervisor Name
Reason for Request			
Amount	Claim Date	☐ Deposit ☐ Cheque	Employee Signature Click Here To Sign

Desktop Form Template shown in eForms Designer.



World Corporation Advance Claim Request

Request No.			
Employee ID	Employee Name	Department	Supervisor Name
Reason for Request			
Amount	Claim Date	○ Deposit ○ Cheque	Employee Signature Click Here To Sign

The form as it appears to the user in an eForms Central library..

Writing the Toolbar Configuration XML

After the template is completed, it can be saved in the ITX Form template (*.itx) format.You can add the toolbar XML directly in the XML by using the following steps:

1. Open the ITX Form template in Notepad or simple HTML editor.
2. Enter the toolbar configuration code at the top level of the XML, between the <revisionInfo> and the <dataModel> elements. For example,

```
<?xml version="1.0" ?>
<template version="2" xml:space="preserve">
.
.
.
<revisionInfo version="3.1">
<toolbarConfig>
     <clear toolbarID="app" />
     <add toolbarID="resources" buttonID="FileNetHome">
           <icon type="url" src="http://hostname/information.gif" />
           <text language="en-US" label="Home" toolTip="FileNet website" />
           <action actionID="get">
           <param>http://www.filenet.com</param>
           </action>
```

```
                </add>
                <add buttonID="send" />
                <add buttonID="saveas" />
           </toolbarConfig>
          <dataModel>
          .
          .
          </template>
```

3. Save the file with an .itx extension.

First Time Deployment to the Forms Manager Server

To add the modified ITX Form template to the eForms Central library:

1. Browse to the appropriate folder in the eForms Central library and click **Add Document.**
2. Click **Browse** and find the ITX template.
3. Click **Add** to add the template to the library.
4. In the Properties dialog box, change the status to Normal and then click **Update.**

NOTE You cannot deploy a template containing modified XML code directly from eForms Designer.

Updating an Existing Template on the Forms Manager Server

If the ITX template already exists in the eForms Central library, you can replace it by:

1. Search for the template in the library, select it and click **Properties.**
2. Click **Browse** and find the modified ITX template.
3. Click **Update** to save the changes.

Data Source and Toolbar Source

Form Type

In this example, the designer applies the form type of the XML data source with toolbar configuration code to create a pre-filled form that loads with a modified toolbar.

The form user selects an Employee ID from a drop-down list in a custom front-end web application. When the matching ID is found in the database (e.g., see code below; `Database=dbname)` HTML eForm opens and loads the form pre-filled with the Employee ID, Department and Phone number information.

The toolbar above the form contains two new buttons, “Home Page” and “Save XML.” Both save and send buttons are removed.

Step 1: Create Web Application for Input Data

```
''''''''''''''''''''''''''''''''''''''''
'Purpose: To be used as a custom front,
'end web app where the user chooses the
'Employee ID from a drop-down list
'and then clicks a button called
'Lookup Employee.
```

```
''''''''''''''''''''''''''''''''''''''''

<%@ Language=VBScript %>

<html>

<head>
    <title>Form Data</title>
</head>

<body>

    <form name="OpenForm" action="CreateXML_FormPost.asp" method=post target="XMLForm">

        <select name="selectEmpID">
            <option value="12345">12345</option>
            <option value="10001">10001</option>
            <option value="10002">10002</option>
        </select>

        <input type=submit value="Lookup Employee">

    </form>

</body>

</html>
```

Step 2: Post Data to Form.aspx

```
''''''''''''''''''''''''''''''''''''''''
'Description: Takes XML data source,
'encodes it in HTML and posts it to
'Form.aspx.
'Remove buttons and add buttons with
'get/post actions.
''''''''''''''''''''''''''''''''''''''''

<%@ Language=VBScript %>
<%

Dim cnn
Dim cmd
Dim rst
Dim cnnSQL
Dim xmlString
Dim getEmpID
Dim formXML
Dim sDepartment
Dim sPhone

'get the Employee ID data from custom front-end web app
getEmpID= Request.Form("selectEmpID")

'ActiveX Data objects required
Set cnn = CreateObject( "ADODB.Connection" )
Set cmd = CreateObject( "ADODB.Command" )
Set rst = CreateObject( "ADODB.Recordset" )

'SQL connection
cnnSQL = "Driver={SQL Server}; Server=hostname; Database=dbname; UID=username; PWD=password"
Call cnn.Open (cnnSQL)

'DB recordset
cmd.ActiveConnection = cnn
cmd.CommandText = "Select * from EmployeeDirectory where DBempID=" & "'" & getEmpID & "'"

Set rst = cmd.Execute ( recordCount )

'get DB data to pre-fill XML data
sDepartment= rst.Fields.Item( "DBdepartment" )
sPhone= rst.Fields.Item( "DBphone" )

'create XML code
xmlString = "<formData version=""2"">"
xmlString = xmlString + "<config>"

'remove the buttons called Save and Send
xmlString = xmlString + "<toolbarConfig>"
xmlString = xmlString + "<remove buttonID=""save"" />"
xmlString = xmlString + "<remove buttonID=""send"" />"

'add a button called "Home Page" that uses a ‘get’ action
xmlString = xmlString + "<add toolbarID=""form"" buttonID=""openHome"">"
xmlString = xmlString + "<text language=""en"" label=""Home Page"" />"
xmlString = xmlString + "<icon type=""id"" src=""get"" />"
xmlString = xmlString + "<action actionID=""get"">"
xmlString = xmlString + "<param>http://www.filenet.com</param>"
xmlString = xmlString + "</action>"
xmlString = xmlString + "</add>"

'add a button called "Save XML" that uses a ‘post’ action
xmlString = xmlString + "<add toolbarID=""SubmitForm"" buttonID=""XMLSubmit"">"
xmlString = xmlString + "<text language=""en-US"" label=""Save XML"" toolTip=""Click here to submit
the form data and save as XML."" />"
xmlString = xmlString + "<icon type=""url"" src=""http://hostname/folderpath/trigger.gif"" />"
xmlString = xmlString + "<action actionID=""post"">"
xmlString = xmlString + "<param>http://hostname/folderpath/CatchXML.asp</param>"
xmlString = xmlString + "</action>"
xmlString = xmlString + "</add>"

xmlString = xmlString + "</toolbarConfig>"
xmlString = xmlString + "</config>"

'model of template cells
xmlString = xmlString + "<model>"
xmlString = xmlString + "<string name=""Department""/>"
xmlString = xmlString + "<string name=""Phone""/>"
xmlString = xmlString + "</model>"

'instance of form data
xmlString = xmlString + "<instance index=""1"" valid=""1"">"
xmlString = xmlString + "<Department>"+ Trim(sDepartment)+ "</Department>"
xmlString = xmlString +"<Phone>"+ Trim(sPhone)+ "</Phone>"
xmlString = xmlString + "</instance>"
xmlString = xmlString + "</formData>"

'HTML encoded to write XML directly to Form.aspx
formXML = Server.HTMLEncode(xmlString)

%>

'takes the encoded XML and posts it to Form.aspx
<html>

    <head>
        <title></title>
        <script language="javascript">
            function LaunchForm()
            {
                window.document.OpenForm.submit();
            }
        </script>
    </head>

    <body onload="LaunchForm()">
        <form name="OpenForm" method="post" action="http://hostname/library/
        form.aspx?template=local,PersonnelActionRequest&data=form,param">
            <input type="hidden" name="param" value="<%=formXML%>">
        </form>
    </body>

</html>
```

Data Source Only

URL Type

In this example, the designer uses a url data source to provide a form data document (e.g., .ifx file) as its content and “text/xml” as its content type.

The form user selects an Employee ID in a custom front-end web application. When the matching ID is found in the database (e.g., see code below; `Database=dbname)` HTML eForm opens and loads the form pre-filled with the Employee ID, Department and Phone number information.

Step 1: Create Web Application for Input Data

```
''''''''''''''''''''''''''''''''''''''''
'Purpose: To allow the user to  choose an
'Employee ID; call form via URL and
'pre-fill fields.
'''''''''''''''''''''''''''''''''''''''''

<%@ Language=VBScript %>

<html>

<head>
<title>Form Data URL Type</title>

<script language="javascript">

    function HandleForm()
    {

    var sID= window.document.OpenForm.selectEmpID.value;
    var sURL= "http\://hostname/folderpath/CreateXML_URLType.asp?queryEmpID=" + sID;
    window.document.OpenForm.action = "http://hostname/library/
    form.aspx?template=local,PersonalActionRequest&data=url," + sURL;

    window.document.OpenForm.submit();
    }

    </script>
</head>

<body >

    <form name="OpenForm" method="post" target="FormWindow">

        Select an ID

        <br>
        <br>

        <select name="selectEmpID" onchange="HandleForm()" >
            <option value="0"></option>
            <option value="12345">12345</option>
            <option value="67890">67890</option>

        </select>

    </form>


</body>

</html>
```

Step 2: Post Data to Form.aspx

```
''''''''''''''''''''''''''''''''''''''''
'Purpose: To open the form and
'prefill it with the data based on a
'match with criteria entered in the
'custom front-end web app.
''''''''''''''''''''''''''''''''''''''''

<%@ Language=VBScript %>

<%
Dim cnn
Dim cmd
Dim rst
Dim cnnSQL
Dim xmlString
Dim getEmpID
Dim sDepartment
Dim sPhone

'get the Employee ID from the custom front-end web app
getEmpID= Request.QueryString( "queryEmpID" )

'ActiveX Data objects required
Set cnn = CreateObject( "ADODB.Connection" )
Set cmd = CreateObject( "ADODB.Command" )
Set rst = CreateObject( "ADODB.Recordset" )

'SQL connection
cnnSQL = "Driver={SQL Server}; Server=hostname; Database=dbname; UID=username; PWD=password"
Call cnn.Open (cnnSQL)

'DB recordset
cmd.ActiveConnection = cnn
cmd.CommandText = "Select * from EmployeeDirectory where DBempID=" & "'" & getEmpID & "'"
Set rst = cmd.Execute ( recordCount )

'get DB data to pre-fill XML data
sDepartment= rst.Fields.Item( "DBdepartment" )
sPhone= rst.Fields.Item( "DBphone" )

'create XML code
xmlString = "<formData version=""2"">"

xmlString = xmlString + "<config>"
xmlString = xmlString + "</config>"

xmlString = xmlString + "<model>"
xmlString = xmlString + "<string name=""Department""/>"
xmlString = xmlString + "<string name=""Phone""/>"
xmlString = xmlString + "</model>"

xmlString = xmlString + "<instance index=""1"" valid=""1"">"
xmlString = xmlString + "<Department>"+ Trim(sDepartment)+ "</Department>"
xmlString = xmlString +"<Phone>"+ Trim(sPhone)+ "</Phone>"
xmlString = xmlString + "</instance>"

xmlString = xmlString + "</formData>"

'write XML to Form.aspx, pass as content type text/xml
'not HTML encoded as URL is called directly
Response.ContentType = "text/xml"
Response.Write(xmlString)

%>
```

Appendix A - Toolbar Configuration Schema

Validate Your XML Toolbar Code

You can use the schema provided here to validate the XML toolbar code that you write for your toolbar configurations. You can check the XML schema using any XML Schema validator website, e.g., http://www.w3.org/2001/03/webdata/xsv.

```
<?xml version="1.0" encoding="utf-8" ?>

<xs:schema targetNamespace="http://[path to website]/
toolbarschema.xsd" elementFormDefault="qualified" xmlns="http://[path
to website]/toolbarschema.xsd" xmlns:xs="http://www.w3.org/2001/
XMLSchema" version="1">

<xs:complexType name="Text">

   <xs:attribute name="language" type="xs:language" />

   <xs:attribute name="label" type="xs:string" />

   <xs:attribute name="toolTip" type="xs:string" use="optional" />

</xs:complexType>

<xs:complexType name="Icon">

   <xs:attribute name="type">

         <xs:simpleType>

               <xs:restriction base="xs:string">

                     <xs:enumeration value="id" />

                     <xs:enumeration value="url" />

                     <xs:enumeration value="none" />

               </xs:restriction>

         </xs:simpleType>

   </xs:attribute>

   <xs:attribute name="src" type="xs:string" />

</xs:complexType>

<xs:complexType name="Action">

   <xs:sequence>

         <xs:element name="param" type="xs:string" minOccurs="0"
maxOccurs="unbounded" />

   </xs:sequence>

</xs:complexType>

<xs:complexType name="AddModify">

   <xs:sequence>
```

```
         <xs:element name="text" type="Text" minOccurs="0"
maxOccurs="unbounded" />
         <xs:element name="icon" type="Icon" minOccurs="0" />
         <xs:element name="action" type="Action" minOccurs="0" />
   </xs:sequence>
   <xs:attribute name="toolbarID" type="xs:string" use="optional" />
   <xs:attribute name="buttonID" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="Remove">
   <xs:attribute name="buttonID" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="Clear">
   <xs:attribute name="toolbarID" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="ToolbarConfig">
   <xs:choice minOccurs="0" maxOccurs="unbounded">
         <xs:element name="add" type="AddModify" />
         <xs:element name="modify" type="AddModify" />
         <xs:element name="remove" type="Remove" />
         <xs:element name="clear" type="Clear" />
   </xs:choice>
</xs:complexType>
<xs:element name="toolbarConfig" type="ToolbarConfig" />
</xs:schema>
```

Appendix B - Capturing Data from a Custom Post Action

Sample Script

```
''''''''''''''''''''''''''''''''''''''''''''''''''''
'Description: Takes XML data from the
'form, writes it to a text file, and
'saves it. A message is returned to the user.
''''''''''''''''''''''''''''''''''''''''''''''''''''
<%@ Language=VBScript %>

<%
Option Explicit
Response.Expires = 0
' Trap errors manually
On Error Resume Next

' Get posted XML data
Dim xmlString, xmlFile, xmlLength
xmlLength = Request.TotalBytes
xmlFile = Request.BinaryRead(xmlLength)

' Convert Binary to string
Dim i
For i=1 to LenB(xmlFile)
xmlString = xmlString + Chr(AscB(MidB(xmlFile,i,1)))
Next

' Set up Constants
Const ForWriting = 2 ' Input OutPut mode
Const Create = True

' Create file and write xml to it
Dim MyFile
Dim FSO ' FileSystemObject
Dim TSO ' TextStreamObject

MyFile = Server.MapPath("Form.xml")

Set FSO = Server.CreateObject("Scripting.FileSystemObject")
Set TSO = FSO.OpenTextFile(MyFile, ForWriting, Create)

TSO.write xmlString

TSO.close
Set TSO = Nothing
Set FSO = Nothing

' Test for errors and return appropriate message to user
If Len(err.Description) <> 0 then
Response.ContentType = "text/plain"
Response.Write "An error has occurred trying to send your request. Please try again. " &
err.Description
Else
Response.ContentType = "text/plain"
Response.Write "Your file has been saved."
End If

%>
```

NOTE You must use XML DOM to extract the individual data fields from the document and insert them into a database. For example:

```
    Dim xmlDoc

    Set xmlDoc = Server.CreateObject("Microsoft.XMLDOM")
    xmlDoc.async = False
    xmlDoc.Load(Server.MapPath("Form.xml"))

    ' Read XMLData field from XML Document
```

```
Dim firstName, formNumber

firstName = xmlDoc.getElementsByTagName("FirstName").item(0).text
formNumber = xmlDoc.getElementsByTagName("FormNo").item(0).text
```

Please refer to your XML Document Object Model documentation for details on how to modify an XML document.

Notices

Trademarks

Program Number: 5724-R89

Printed in USA

GC31-5499-00