ViewsFlash 7 Release Notes

ViewsFlash 7 introduces significantly enhanced capabilities for creating sophisticated questionnaires and forms and interoperating with existing systems.

Notes for system administrators

New training and learning aids

- Revised Examples are available right off the ViewsFlash home page. They contain copious notes about the techniques used to construct them.
- Instructional Videos are now available on the web at **www.cogix.com/docs/vf70/video**

New questionnaire and form features

- A new method for **authenticating questionnaires** using a generic hidden field with a unique ID is now the default. This method is well suited for anonymous and non-user centered questionnaires.
- A new way to **arrange questions in a layout** provides precise control over placement in two-dimensional layouts.
- A new way to display **custom validation alert messages** for conditions such as required fields missing or invalid characters in numeric fields.
- New **Question Styles** for a Calendar date picker, validating text fields that conform to a Regular Expression, and checking that items are ranked in a unique order. The **DisplayOrHide** and **Calculate** styles have been made more general and flexible, and can now be used with most question types and perform logical and conditional operations.
- The **SQLSelect**, **SQLUpdate** and **DynamicDropDown** actions can now work with data in a **different JNDI datasource**, even on a different brand of database. They offer more control over error messages and logging, and over conditions when there is no data. The documentation for each action describes its options in detail.
- New action **SQLDisplay** allows retrieving data from multiple rows in a database table using a SQL query and displaying the data in tabular form in a questionnaire.
- New Action **CascadingMenus** allows constructing multiple linked drop-down menus, such as for presenting car Make, Year, and Model. The values for each dropdown depend on the values of the previous one. All the data for the dropdowns comes from a database table and can be personalized using a dynamic database query.
- The **Script** action now includes built-in functions to redirect the browser,get the HttpRequest, HttpResponse, and HttpSession objects, to write to the ViewsFlash log, to display questions conditionally, to send personalized emails conditionally, and to display error messages at the top of a page and for multiple questons. It also includes a way to create Scripts that are executed only when using the data review API.

System administration

- System administrators should review the **Upload** technical notes. Documents that are uploaded with questionnaires are now stored in the database by default, instead of the file system.
- If you are using Oracle, or using the Upload Attachments feature, **you must review** the **Upgrading to ViewsFlash 7** notes.
- When using J2EE declarative security for application and questionnaire authentication is needed, a new **OtherAuthentication** class is provided that is easily configured with servlet parameters.
- New servlet properties **showadminservleturl** and **usingsslproxy** are provided for use with reverse proxy servers.
- New servlet properties **dbcreatetablespacestatement** and **dbtablenamesuffix** are provided for when using **DB2** for specifying a CREATE TABLESPACE statement and allocating tables in specific table spaces.
- When saving questionnaire data in its own database table, **indices** can now be created to speed up data retrieval by other applications.
- The ColumnBegin, ColumnBreak and ColumnEnd styles have been deprecated and are no longer included. Use the Styles page to see if they are used, and delete them if they are not. If they are used, they will continue to work as they used to.

Miscellaneous

- **E-mail invitations** now include options for retrying failed emails and for seeing who has been invited and who has completed the questionnaire. The options for **sending e-mai**l with each questionnaire now include sending a test e-mail and allowing an e-mail to be sent when a questionnaire's content is revised.
- When including an item from a **Question Library**, and renaming the question names by prepending or appending a

name mask, a new option allows for references to those question names to be automatically modifed by prepending or appendin the name mask to references to those question names automatically, rather than having to rename them manually.

- **Staging** files can now be stored on databases on different servers and exchanged without using the file system

---

ViewsFlash 6.2 release notes

ViewsFlash 6.2 introduces several new features. Data reports now allows displaying the human-readable text of the **respondent's answer**, rather than the value stored in the database. Another ehancement to Data reports allows appending an additional filter in the report URL.
An enhancement to the **Participants API** allows including additional fields in email-invitations. An enhancement to the Web Services API allows for programmatic **survey creation** and for retrieving a complete questionnaire definition **as XML**.

ViewsFlash 6.1 introduces several new features. **Script** actions now have easy ways to access Java request and response objects, as well as methods for redirecting the browser after saving and submitting entered data.

ViewsFlash 6 introduced numerous usability enhancements, including a simpler **Question Editor**, consolidation of the Live Tally and **Summary** pages, and the removal of the Bulk Vote, Remove Entries, and Set Rank legacy polling functions which are no longer in general use (but **can be enabled** when needed). A Level option is added to **MyViewsFlash**, which allows beginner uses to take advantage of the most commonly used features while hiding others that are often not needed.

The entire application, from user interface, to reports and questionnaire look and feel, have been visually upgraded for a more pleasant and effective experience, the HTML editor has been integrated throughout the application, and the Matrix and Grid wizards have been upgraded for ease of use and detailed control over presentation. Altogether, these enhancements make ViewsFlash significantly easier to use without compromising its power.

The standard **Response** page and the standard Results Style have been updated so that the image, header, and footer fields from the General page are used automatically in the Response page by default. This creates a more standard look throughout the questionnaire and its response page. A new section in the Response page allows for presenting the response message in the body of the page, between the header and footer.

The Data, Univariate, Cross Tabulation, and Live results reports have been redesigned. **Univariate** Reports now can be enriched by adding text and HTML using questions created with the **ReportText** question style. **Cross Tabulations** include a new option for shading report cells according to their content, so that cells with a low percentage are light while cells with a high percentage are strong. This pattern helps spot relationships visually in a cross tabulation.

The **Publish** page includes the URL of the final questionnaire. When using Question or Captcha authentication, the URL now shows the URL modifications that are appropriate for that method.

The **Date** and **Number** question styles now handle worldwide date and number formats. This was made possible by using a new capability in question styles, which can now specify additional parameters for a style such as the decimal separator character.

The **Matrix Wizard** can now create Likert and Semantic Differential scales. Their appearance has been enhanced by using fewer lines. The matrix is automatically sized for evenly spaced columns, and exact control over column widths is also available by editing the first question (the header) in a matrix.It is even possible to render these matrices using a custom CSS style.

The **Grid Editor** has been enhanced by allowing control over the generated question names, thus making them more meaningful. Grids now have an "elastic" mode, where only a few rows in a grid are shown, and a "More rows" link is provided that allows for entering additional rows of data. Because most grids use only a few rows, this allows for a more compact presentation while leaving open the option to fill out the entire grid when necessary.

A new **DynamicDropDown Action** creates drop down menus whose content is determined dynamically when the questionnaire is presented to the respondent, based on either the result of a database query or the content of questions asked on previous pages.

A new **Upload style** allows respondents to attach documents to a questionnaire, such as a spreadsheet, a document or a picture. The uploaded documents are retrieved using the standard Data reports by clicking on a link.

A new **XML Import and Export** option allows creating and exchanging questionnaires in XML. This option is also useful for creating questionnaire translations.

New **Saving Data servlet parameters** allow the Administrator to forbid saving questionnaire as text files, in their own database tables, or in the Normalized tables. This can simplify user choices and enforce database and file system usage

policies.

A new **cross-site scripting** servlet parameter, provides additional checks for preventing cross-site request forgeries and cross-site scripting. Other standard practices for preventing cross-site scripting and SQL injection attacks have been strengthened and are always enabled.

ViewsFlash 5.8 introduces an **HTML editor** to make it easier to enter formatted text in headers, footers, and questions, using easily customizable styles. The editor checks English spelling in Internet Explorer and in all languages in Firefox. The **User Guide** has been revised.

This release introduces DisplayOrHide, a new Question Style that dynamically **reveals or hides** questions on a page depending on the value of earlier questions. It now allows including a **"None"** button in radio button questions to indicate no choice. The **Installation** and **Database** documentation have been thoroughly updated. **Internationalization** now provides a way to create a questionnaire in one language with localized messages, as well as questionnaires in multiple languages.

Administrators please note: when this release is deployed using User Security, the Examples and Default places will be **locked** so that only Administrators can create questionnaires and modify their settings. A new public Practice place is created for users to learn how to use the software. The Examples and Default places can be unlocked by an Administrator.

ViewsFlash 5.7 provides a new way to authenticate visitors, using **only their email addresses**. This scenario is very useful when there is a membership list with email addresses, but no passwords, and there is no other way to authenticate visitors. To support this, a customized login page can be specified in the **Security** page.

ViewsFlash 5.6 introduces **Calculate**, a new Question Style that performs live calculations in the browser. For example, given a total salary and a number of employees, this will calculate an average:
average=total_salary / employees;
The result can be optionally formatted with currency symbols, decimal point and digits, and grouped in thousands: $1,234.56, €1.234,56. The calculated field can be made read-only.

New features to help with **Questionnaire Printing** have been added, and the Standard styles have been enhanced to allow easily specifying your own CSS style attributes, such as page borders and colors and font attributes such as small-caps.

A new **internationalization** facility provides a common language page for all translations of a questionnaire. A single URL can be provided to all participants, and they choose the language they want to respond to the questionnaire in from a drop-down menu with choices in their own language.

A new customizable **CAPTCHA** log in page for authenticating anonymous questionnaires is available.

A new section on **How to use the Web Services API** section has been written, including examples for using it in JSP pages and AJAX.
This API is now disabled by default, and must be enabled as described.

ViewsFlash 5.5 introduces a new **Ratings API** that allows easy construction of ratings systems for content such as articles, videos, photos, products.

ViewsFlash 5.4 allows using the **Data View** query and reports to examine responses that are still in progress and that have not been submitted yet. The Question library is **enhanced** to work with Grids.

ViewsFlash 5.3 includes a **Grid Editor**, allowing very powerful grids of similar question to be built. Releases 5.1 and 5.2 were maintenance releases.

ViewsFlash release 5 is a major release with powerful capabilities, including a **Visual Editor**, **Charts**, Reports, **Filters**, accessibility standard compliant questionnaires (section 508, WCAG, XHTML 1.0 Transitional), a streamlined user interface, a **Matrix Wizard**, question help, randomizing and hiding questions, and a MyViewsFlash preferences section.

A note on terminology for existing users who are upgrading: "Polling places" has been shortened to "Places".

Administrators who are upgrading from earlier releases: please read first **Upgrading to Release 5**.

New capabilities in release 5

- A new **Grid Editor** provides a way to create question matrices made up of similar rows of the same questions. An example is included in the documentation.
- A new **Visual Editor** provides a much simpler, intuitive way to create and edit questionnaire forms. The survey is shown

in its final form, and a mouse click opens up the question designer to revise or create questions and question matrices. The entire application user interface has been reviewed and polished making the application dramatically easier to use. The traditional List Editor is still useful for more complex operations such as copying and moving multiple questions and using the clipboard.

- A new public place, Examples, provides ready-made examples of basic and advanced questionnaires. They are locked but can be copied for further study and modification. A new **Question Library** contains questionnaire snippets for frequently used operations such as **branching**, **scoring**, and **hiding questions**.
- The entire application user interface now has a more appealing and functional design, including a **My ViewsFlash** page for showing survey dates in your local time zone and locale for multinational users, time-stamping all objects and sorting object lists by clicking at the top of each column, and using AJAX techniques in the user interface to reveal features only as needed.
- A **Matrix Wizard** provides a foolproof way to construct ranking matrices, Likert matrices, and matrices of questions that share common answers.
- **Question help** allows entering additional explanatory text for any question. A [?} link appears in the questionnaire, which reveals the question help text when clicked. Other renderings are possible by creating a custom style template.
- Saving very long Text fields (larger than 4K) is now possible in the Normalized mode.
- A new **Randomize** capability allows presenting the questions in a page in to different survey takers in a random order. The **Script** custom action now includes functions that hide questions. The hidequestion("questionname") function hides questions that should not be presented to a respondent based on the answers to earlier questions. These two capabilities together significantly enhance the ability to personalize a questionnaire and to randomize the effect of question presentation.
- **Accessible** questionnaires are now created by default, without requiring any special options. The new default styles, named Standard_, create questionnaires that comply with XHTML 1.0 Transitional, section 508, and WCAG priority 2 and many priority 3 requirements. The new Matrix Wizard works with accessible questionnaires as well.
- The new Standard_ **style templates** can be used in both stand-alone surveys and in Portals. If the portal is 508 or WCAG capable, created surveys are accessible in the portal. Prior to release 5, special styles were needed for Portals.
- Analytical query tools (Univariate and Cross tabulation) now create pie **data charts**, bar charts and 3D stacked bar charts. Charts can be customized and viewed in PowerPoint and Excel as well as the browser. Commonly run queries can be saved as Reports and retrieved with one click.
- Queries and reports can now **filter** what data to use, such as state = CA. A new **Data View** query and report allow displaying saved data as HTML, Excel, and XML, in all character sets. In HTML, the data can be sorted and resorted by simply clicking on the top of each report column.
- Alerts, such as "Please answer this question", are now available in **internationalized surveys** in English, French, Spanish, German, Italian, Japanese and Arabic. Explicit support has been added for including right-to-left, language, and charset tags when needed.
- A new database change watchdog alerts the survey creator who modifies a questionnaire after data has already been collected. If the modification requires a change in the table definition, the questionnaire will be closed until it is published again, which will alter the table appropriately. Without this safety feature, storing responses would result in loss of data due to a database exception.
- Data is now saved in questionnaire tables before the survey response page is shown to the respondent. While this introduces a small delay, it lets the respondent know immediately when a database error occurs. In such as case, live tallies are not updated with the record that could not be captured.
- Secure and anonymous questionnaires can now be done by simply selecting Basic authentication or Cookie authentication in the **Security** settings.

Previous release notes

Next: **Introduction**

Introduction

This documentation is designed to be read online, in one window, with the software running on another window, ideally side by side. A PDF version is available in the Support section of **www.cogix.com**.

The software is designed to be largely self-documenting. Each screen includes instructions on its use and available options. This guide focuses on explaining concepts and operational information in greater depth.

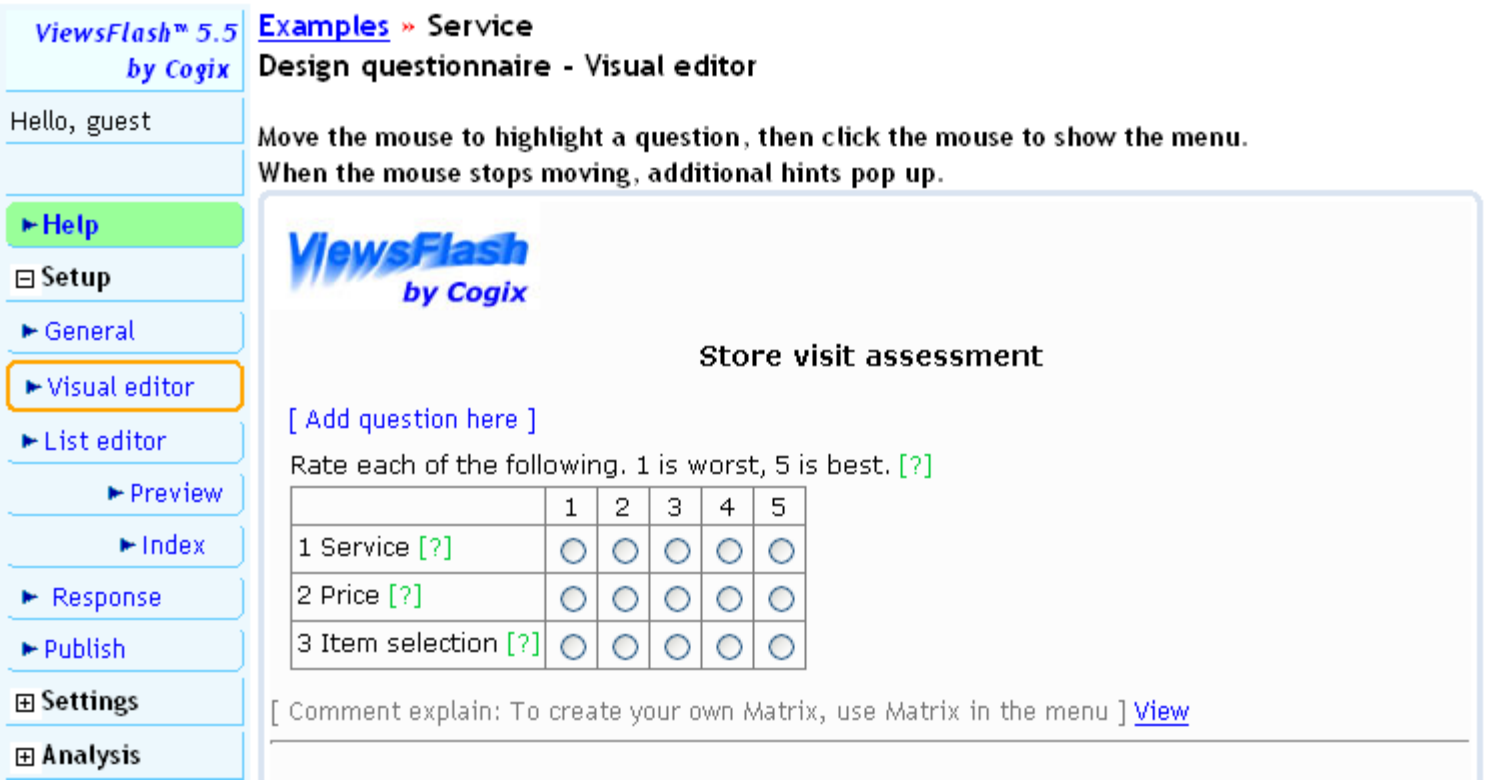
In this documentation, the term "questionnaire" is used as shorthand for survey, poll, assessment, or form. From ViewsFlash's point of view, these are all variations on the same theme of asking one or more questions, tabulating the answers, and displaying results. Sometimes "poll" is used for its brevity.

Concepts

The software deals with **Places**, **Questionnaires**, **Styles and Invite Lists**. A place is where a user creates a series of questionnaires, protected by Access Controls. For instance, a company might have HR, Marketing, Sales, and Public places, where different users create surveys that run independently of each other. All questionnaires in each place must share the same Styles; a user who wants to create questionnaires with different styles needs to use more than one Place.

The ViewsFlash application user interface

The HTML user interface is simple, with a trail-crumb along the top and a menu bar down the side. To navigate, use the links at the top of the page or on the side of the page. Actions are performed by filling out the form fields on a page and pressing the Go, Submit or other buttons. For example:



This page is used during questionnaire creation. This questionnaire is called Service, and it is hosted in the Examples place. Clicking on Examples in the crumb trail at the top of the page navigates to the Examples place. The links on the left-side menu bar vary from page to page. The Help link opens up this documentation at the corresponding page. The links with ⊞ and ⊟ open up further options. In this case, the Setup option is open, showing seven pages that are used to set up a questionnaire. The current page is highlighted in orange. Pages also use links to illustrate the most common next steps, like this:

⇒ Next: Setup the response

Clicking on Next will go to the recommended next page.

Skills needed

There are three sets of skills needed to use the ViewsFlash software. In a large organization, they will usually be represented by the IT department, the Design and Production department, and the users who construct and analyze questionnaires. In a small organization, they might be one or two people. In this documentation, we personify these skills as the Administrator, the Designer, and the User, respectively. Here are their respective responsibilities and skills in more detail:

Person	Administrator	Designer	User
Responsible for	Installation, configuration, web server, directories, permissions, security, backups. Interoperation with databases, JSP pages, and web services.	Look and appearance of polls and surveys; integration with content management system, if any	The questions to ask, when to ask them, what to do with the responses and results
Skills	System administration, database administration	HTML, content management system knowledge	Editorial, analytical
Effort required	Moderate, one-time.	Depends on degree of customization desired; one-time	Ongoing
Further reading	**System administration**, dealing with installation and setup, how to set configuration parameters, and databases. **Production** is recommended.	**Production**, dealing with the production cycle, creating Style Templates, how to embed them in containing pages, how to interface with content management systems	**Creating questionnaires**, where the creation process is explained in this and subsequent pages.

Everyone should read the **Architecture White Paper** for information about System Architecture, Performance and Requirements.

Next: Places

Places

A Place contains a series of surveys, polls, or quizzes. For example, Human Resources can conduct its employee surveys in one place, while Marketing conducts customer surveys in another. All the questionnaires in a place share certain common settings, including Styles that give them a uniform look and feel, whether they run simultaneously or one at a time. Typically one person is responsible for all activity in one place.

When using **User Security**, a Place is visible only to those users who have Access Rights to the place; it is invisible to other users. The Administrator is responsible for creating Places and initially assigning Access Rights to them. The Administrator may delegate some of those rights to users in a place.

The Places page lists all places by name. The Current and Archive columns indicate the number of questionnaires in that status. The Latest column gives the title of the most recent questionnaire. The N column indicates how many questionnaires have been gathered in the Latest questionnaire. The Modified column indicates when each Place was last changed. Clicking on the column headers at the top of each column sorts the table by that column. Clicking on a Place allows managing its questionnaires.

ViewsFlash uses a Default place, whose settings are used as a default for other places that are created later. After becoming familiar with Settings, configure the Default place and the Default questionnaire so that their settings will be used by default.

To create a new place, chose New place, enter a name for it, and press Go. Places are named uniquely, using letters, numbers and underscores.

To copy the settings from an existing place, choose Copy Selected, check the box next to the place to copy, and press Go.

To remove one or more places, choose Remove Selected, check the box next to each place to remove, and press Go.

Administrators can see who has what access rights and can assign access rights for one or more Places.

Next: Place Settings

Place Settings

To create a questionnaire, first click on the **Current** link at the top of the page.

The Place page has three sections, chosen by the links at the top of the page: Current, Archived, and Settings. Current lists questionnaires that are in progress. Archived lists closed questionnaires. Settings changes Place characteristics.

1 Holds a place description for your later reference.

Choose a form Style to use when composing the questionnaires in this Place. Standard is the default. The Appearance link opens up a set of defaults that can change the width of the questionnaire, the font, size, and color used, whether a progress bar should be included, whether a border should appear around questions, and whether an image or logo should appear on each questionnaire's page.

Click here *for additional technical information when not running ViewsFlash on a database and embedding polls on web pages*

2 Choose a Response Style to use when composing what a respondent will see after completing the questionnaire. Results is the default style.

The Appearance link opens up a set of defaults that can change the width of the response, the font, size, and color used, and the color of response bars when revealing results to participants is chosen.

Click here *for additional technical information when not running ViewsFlash on a database and embeding results on a web page.*

3 You can provide a list of email addresses to notify when a survey is opened and closed.

4 For most uses, choose "Concurrently". Multiple questionnaires can be opened simultaneously with this setting. Choose "One at a time" when scheduling only one current poll at a time in a place, such as when creating a daily poll for the home page of a web site.

When using password security, the Place and its questionnaires can be protected by choosing this option and entering a password twice. Note that the ViewsFlash Administrator password gains access to all places, and can be used to change the password in a place. When using User Security, this field is not shown.

When logged in as an Administrator, an option to Lock this place is shown. Checking this box prevents users from creating or deleting questionnaires and from accessing this page. Only administrators can change this setting. This option is used to protect the Examples and the Default places, making users use the Practice place or their own places instead. This option is only available when using User Security.

5 If there is no questionnaire scheduled, and an attempt is made to display the "current" questionnaire, the message in this field will be displayed instead. Normally, a single space will suffice. You can also display a message like "There are no polls open at this time." You can use HTML in the message.

After the options have been chosen, press Submit and continue with the **Create Questionnaire** link.

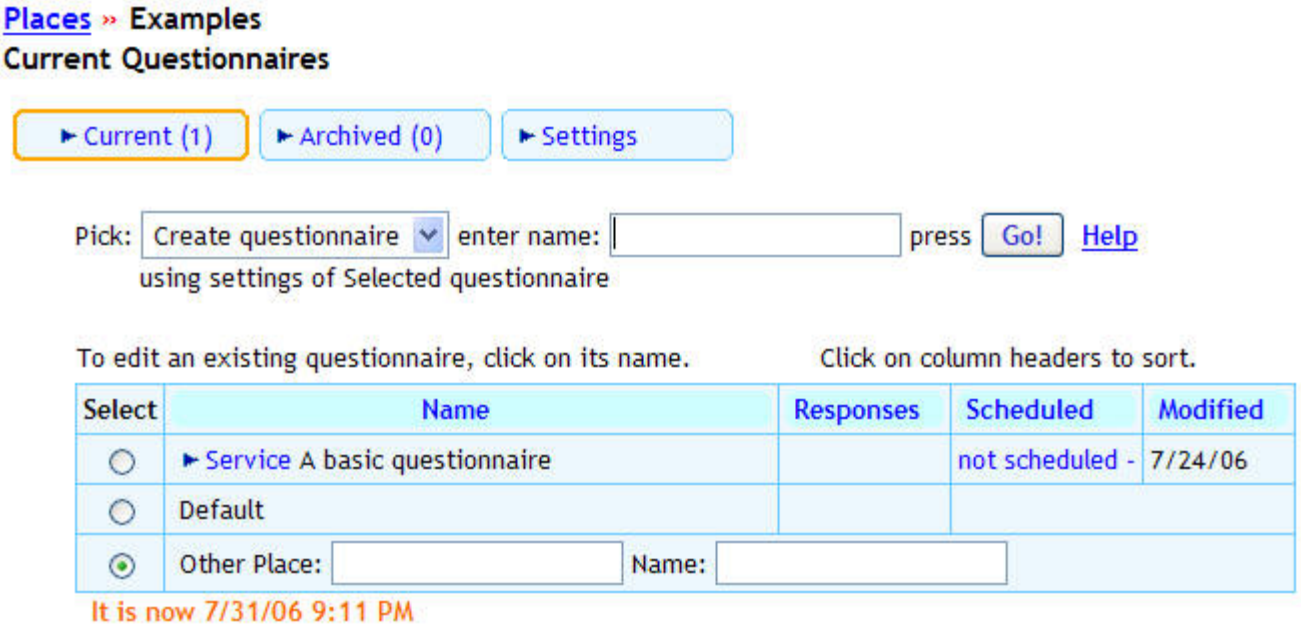
Next: Current Questionnaires

Current Questionnaires

This Place view displays the questionnaires that are currently in progress. After the Name, the Responses column shows how many responses have been received. The Scheduled column indicates the dates and times when the questionnaire is open. Modified indicates when the questionnaire was last changed. Sorting on the column names at the top sorts the listing by that field.

Clicking on a name will take you to that Questionnaire, where you can change its questions and settings. Clicking on the number of Responses will pop up a page with results at this moment. Clicking on the scheduled dates will take you directly to the Publish page, where you can change the publication dates.

Creating a new questionnaire
This description refers to the illustration below.



To create a new questionnaire, select Create Questionnaire from the Pick drop-down menu, and enter a name for it. Names must only use letters, numbers, and _ (underscore). To use the same questionnaire settings as a previously created survey, select it by clicking on the radio button next to it in the Select column. Then press Go!

To create a simple one-question poll, select "Create quick poll "from the Pick drop-down menu, give the poll a name, and press Go. An abbreviated set of steps will allow creating a poll very quickly.

Pick Copy selected to copy all questions and answers in the Selected questionnaire. as well as its options. This is very useful for creating a survey which is repeated periodically, or for creating variations on a previous survey or variations off a survey pattern.

The Other line allows copying a questionnaire or its settings from another Place, by entering the place name and the questionnaire name. This provides a powerful facility for testing a questionnaire in one place and then copying it to another, or for creating Examples that can be shared. You will need to know the exact name of the Place and questionnaire. If User Security is active, you can only copy surveys from a place where you also have Survey Creation access rights unless you are logged in as a ViewsFlash administrator.

When a questionnaire is closed (i.e., when its scheduled time expires or its quota is full), it is automatically removed from the Current list and is sent to the Archived page.

To remove a questionnaire, check the box next to it, pick Remove from the drop-down menu and press Go.

To append a questionnaire to an existing one, pick Append to **Questionnaire**, enter the name of the **questionnaire** to append to in the name field, select the questionnaire to be appended by using the radio buttons, and press Go!

Export selected as XML allows checking several questionnaires. After pressing Go, a Download zip file appears right under the Pick menu. The downloaded zip file contains one XML file for each questionnaire selected. These files can be used for backup, or sharing and exchanging questionnaires.

Import questionnaire as XML allows uploading an XML file created with Export as XML. If a name is entered in the name field, the imported questionnaire will be renamed accordingly. The imported questionnaires are stored in the Place where they are imported, regardless of ther original Place. The uploaded file can be a single XML file extracted from the zip file created by Export as XML, or it can be a zip file with multiple questionnaires in it, in which case the names of the questionnaires cannot be changed. If the name of an imported XML file is already in use,the file will not be imported.

Next: Archived

Archived Questionnaires

This place view displays questionnaires that have been closed. After each questionnaire's name, the Responses column shows the number received. The Scheduled column indicates the period when the questionnaire was online.

Clicking on a questionnaire's name will take you to its Design page, so you can change its questions and settings. Clicking on the number of Responses goes to the summary page. Clicking on the scheduled dates goes directly to the Publishing page, where a questionnaire can be reopened by changing its publication dates. It is legitimate to reopen a questionnaire and gather additional responses. It is even possible to change the questionnaire and gather additional data.

To remove a questionnaire permanently, click on the Remove check box next to it and press Submit. WARNING: After removing a questionaire, there is no way to restore it.

Archive Display pages

It is possible to construct automatically an Archive Display page that displays links to previous questionnaires and lets the visitor browse through historical results. The remaining options on this page deal with these settings.

Next to each questionnaire, a marked check box indicates that it should be included in the Archive Display. Only questionnaires that are checked are included in the Archive Display. When a questionnaire is archived, it is automatically included in the display.

The Archive Display is prepared using the Archive Style chosen in the pull-down menu. The display lists the archived questionnaires to be shown, in either list of links form or in menu form. When a visitor selects a questionnaire, the Archive Display is refreshed, with the results of the selected questionnaire displayed above the listing using the Response Style chosen from the pull-down menu of Response Styles.

After selecting the suitable templates and removing the checkbox in the Show column from any questionnaires that should not be shown for editorial reasons, press Submit.

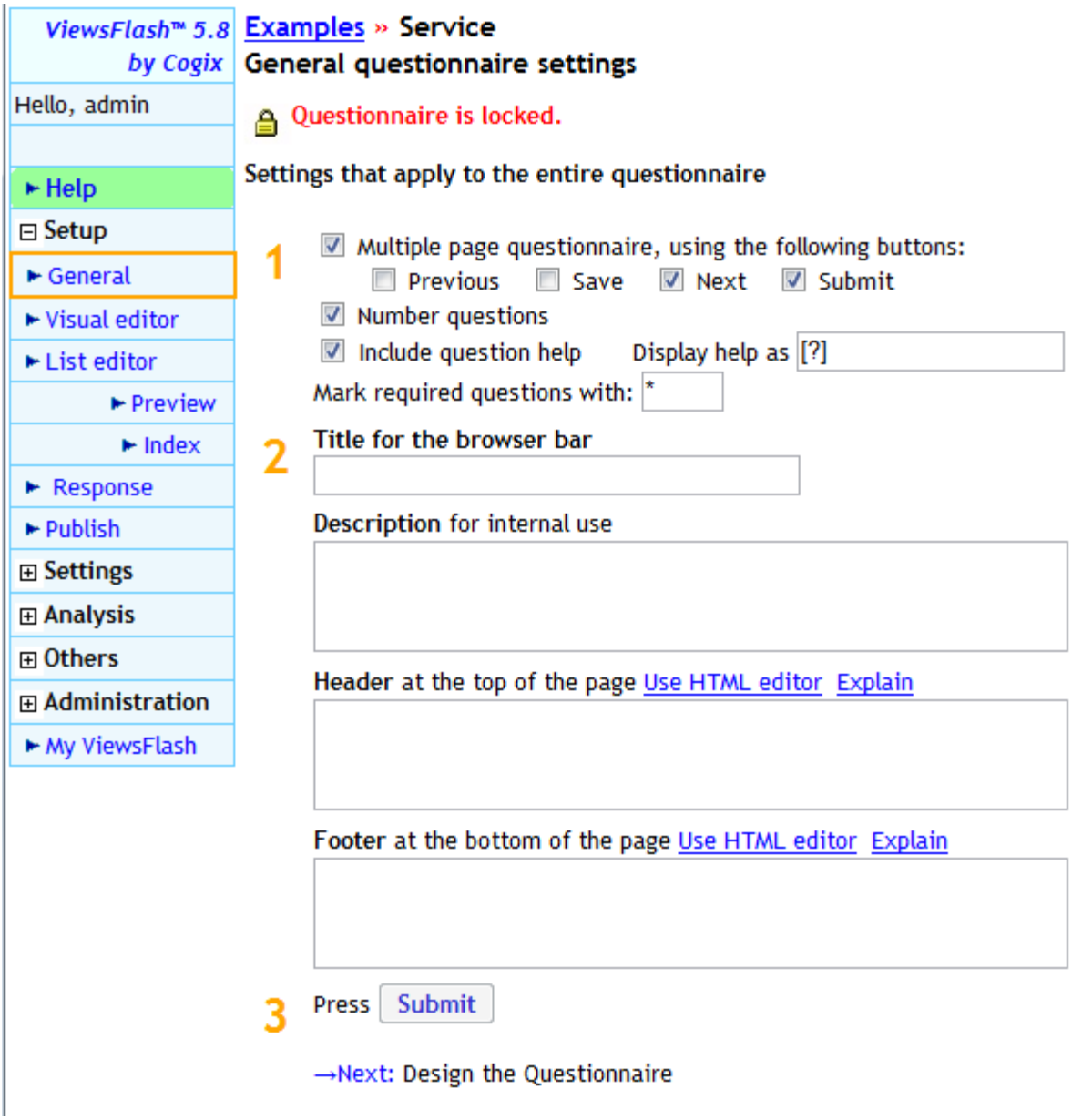
Clicking on the Archive Report URL will pop up a window with the live Archive Results display. This URL can be copied and used in the site to link to this archive page.

Note that Archive Style Templates should always be a whole page.

Next: Setup Questionnaire

General Questionnaire Settings

This page sets options for all questionnaire pages. This is the first page shown when creating a new questionnaire, and it looks like this:



The buttons on the left side, which change depending on what is possible on the page, provide navigation to other pages. The plus and minus symbols open up further options in the tab. In this example, the Setup tab is open, and the current page is highlighted in orange.

1 Check "Multiple page questionnaire" if the questionnaire will span **more than one page**. If so, indicate what buttons should be used on each page. The Previous button allows respondents to go to previously answered questions. The Save button allows respondents to stop in the middle of a long questionnaire and to resume later where they left off, at the URL presented to them when they press Save. The Next button proceeds to the next page in the questionnaire. The Submit button writes the responses to permanent storage. Normally, the Next and Submit button are checked.

To number and renumber questions automatically, check "Number questions".

Checking "question help" allows entering additional information for each question that explains in greater depth what the question is about or offers hints about how to answer. When the help indicator [?] next to the question is clicked, the help text is shown; a second click hides it. Try it by clicking on the [?] in this example:

1 How do you get your TV? [?]

- ○ Off cable
- ○ Off the air
- ○ Using Tivo
- ○ By satellite

To modify how question help is rendered, create new **Style Templates** by copying the Standard style and select the new style template in the Place's settings page.

2 The Title field is displayed in the page's title bar. The Description field summarizes what this questionnaire is about, and is only displayed in the Place page. The Header field is rendered at the top of each page. The Footer field is rendered at the bottom of each page.

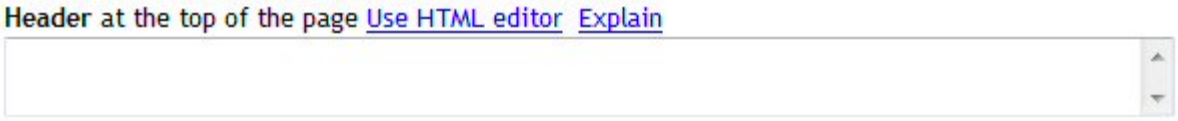
The header and footer, like many other fields, have a Use **HTML Editor** link; click on that link to compose formatted text. The Explain link next to it explains how. There are many Explain links throughout the application, which deserve close reading and provide quick and to the point explanations.

3 After making all changes, press Submit. Then go to "Design the Questionnaire", which goes to the Visual Editor.

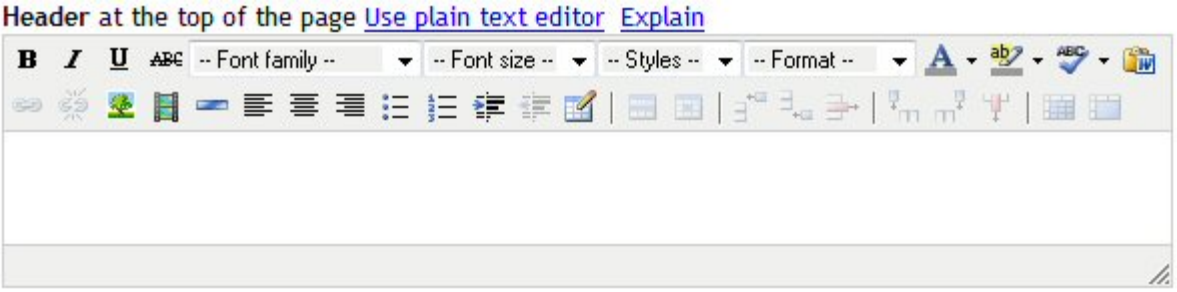
Next: HTML Editor

Using the HTML Editor

Many fields allow entering plain text or formatted (HTML) text, like in a word processor. These fields include a "Use HTML editor" link, like this:



Clicking on Explain brings up this page in a small window. Clicking on Use HTML editor transforms the editing region into a fully functional text editor, and the link changes to "Use plain text editor". You can switch between the two editors at any time; any HTML you enter with the editor can be edited further as plain text.



Use the keyboard just like a word processor. The following special characters are often useful:
Enter or return starts a new paragraph. Shift-Enter inserts a line break within the paragraph.
Shift-space inserts an extra "non-breaking" space. To enter a single non-breaking space, enter two and backspace once (IE 7 only).
In Firefox, use the plain text editor and enter

To resize the editor, drag its lower right hand corner with the mouse.

Move the mouse over the buttons to see their functions, described here from left to right.

To format, select the text and then apply formatting by using one of the buttons, such as **bold**, *italic*, underline and ~~strike through~~, as well as the Font family and font size drop-down menus. The Styles menu applies the pre-defined styles Medium, SMALL CAPS, **Large**. The Format drop down menu applies HTML tags P, H1-H6, and pre. The text color and background colors set the selected text to the specified colors. For most consistent results across browsers, use Font family, Font size, and Styles rather than Format.

The spelling button is only seen in Internet Explorer. In Firefox, use the built-in spell checker. See **spelling** below.

To paste HTML text from a web site, select it in the browser with the mouse and copy it to the clipboard with Ctrl-C, then paste it into the editor with Ctrl-V. To paste formatted text from Microsoft Word, select it and copy it with Ctrl-C and paste into the editor using the Paste from Word button.

To insert a hyperlink, enter the link's text, select it, and click on the Insert/Edit link button. Fill in the pop-up menu with the complete URL, including http://, such as http://cogix.com and other information. To remove a link, place the cursor inside the link and click the Unlink button.

To insert an image, place the cursor where the image should be inserted, press the Insert Image button, and provide the image URL, including http://, such as http://cogix.com/vf2/cogixtoknow.gif . The image dimensions will be entered automatically. To find the URL of an image on the web: using Firefox, right click on the image and choose Copy Image Location. In IE, right click on the image and copy the URL from the Properties dialog. ViewsFlash does not provide a way to upload images to a server.

To insert other media such as Flash, QuickTime, Shockwave, Windows Media or Real Media, place the cursor where the image should be inserted, press the Insert / Edit embedded media button, and fill out the form. In the File/URL field enter the URL of the media, including http://, such as http://www.cogix.com/filename.swf . ViewsFlash does not provide a way to upload media to a server.

Additional formatting options include inserting a horizontal rule, aligning text left, center, or right, and creating bulleted, numbered, or indented text blocks.

The Table icon allows entering a table, and the remaining icons allow manipulating the table.

Spelling

In Firefox, use the built-in spell checker, which checks spelling as you type and suggests correct spellings by clicking on a misspelled highlighted word with the right mouse button. In IE, use the Spell icon, which highlights misspelled words, and double-click on a misspelled highlighted word with the left mouse button to see suggested spellings.

HTML Design notes
The Styles menu can be customized by modifying the cogix.css file in /ViewsFlash/styles/cogix.css. The P, H1-H6 and pre tags are not defined and are interpreted by the browser. To define them, the best place is in a copy of the Standard form style.

Use the "Paste from Word" button to import formatted text copied from Microsoft Word, as well as HTML text.

Next: Visual Editor

Design Questionnaire - Visual Editor

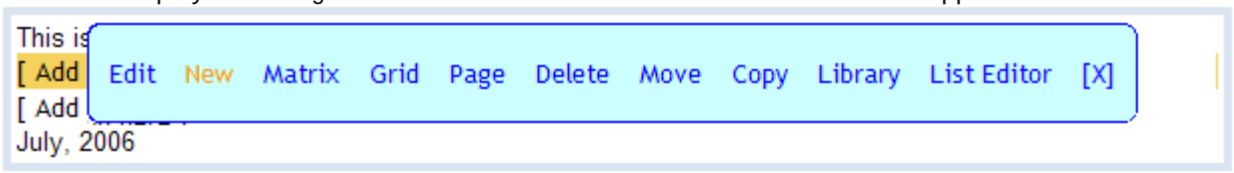
When constructing a questionnaire form, the easiest way is to use the Visual Editor. A few more advanced questionnaire creation options are available in the **List Editor**.

A questionnaire is composed of questions, organized into pages; pages are delineated by page breaks. The Visual Editor provides a way to create, edit, remove, move and copy questions and page breaks, and to bring in question sets from the question library.

Every question has a name, which are used for reference in operations such as branching and scripting. Names also are used to name the columns in database tables.

The Visual Editor displays the questionnaire the way that the respondent will see it, surrounded by a light blue box, with [Add here] at the top and bottom, between the header and footer, like this:

As the mouse is moved over this area, different regions and questions are highlighted in orange, and a tool tip that says "Click for menu" displays. Clicking the mouse once causes the Visual Editor menu to appear where the mouse is located:



To close the menu, click on the [X] on the upper right hand side.

The menu has the following options:

- Edit. When the mouse is over a highlighted question, such as a radio button, clicking on Edit lets you revise the question.
- New. Lets you add a new question below the question which is currently highlighted. You will start by giving the question a name.
- Matrix. Lets you insert a matrix of questions right below the highlighted question. A matrix is a series of questions with identical answers, such as Likert rating scales. See **Matrix Wizard**.
- Grid. Lets you create a grid of similar questions repeated multiple times, such as financial data over multiple years. See **Grid Editor**.
- Page. Inserts a page break after the currently highlighted question. This indicates the end of one questionnaire page and the beginning of another.
- Delete. Removes the highlighted question or page break.
- Move. After clicking on Move, the highlighted question will be marked for moving. When the mouse moves again, it will highlight questions in a red color. A second mouse click moves the question below the red highlighted question. To cancel a Move, press the space bar.
- Copy. Same as Move, but a copy of the question is created below the red highlighted question. To cancel a Copy, press the space bar.
- Library. Allows inserting commonly used question setups from the question library.
- List editor. Switches away from the Visual Editor to the **List Editor** questionnaire designer, which allows moving and copying multiple questions, cutting and pasting questions to and from a clipboard, and renaming questions.

Other useful editing tools are the Preview button on the left hand side, which gives an even clearer picture of the finished questionnaire, and the Index button, which gives a summary listing of all questions, which is specially handy when creating branching scripts.

Hidden questions are displayed like this:

[Hidden total: Running total field]

This includes the question name and its description.

Action questions are displayed like this:

[Action hide_why_visit: Script when=before] Hide

```
if ( score > 3 ) {
hidequestion ("why_visit");
}
```

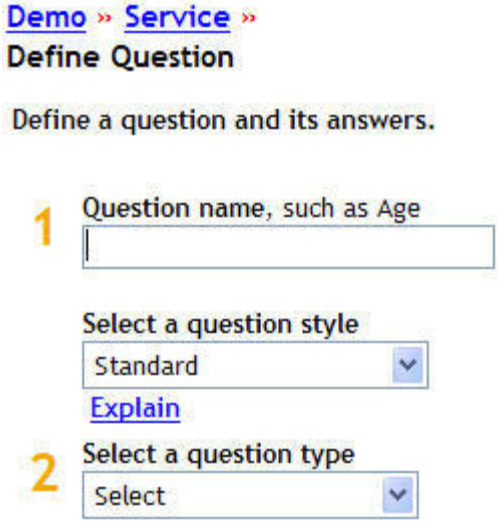
This includes the question name and its Action. A View link opens up a box that shows the script details; when the box is open, the link turns into a Hide link that closes the box. This allows very quickly getting an idea of what an Action does without having to open the question for editing.

Next: Define a Question

Define Question

This page defines one question and its answers. The contents of this page change depending on the question's style and type, as explained here.

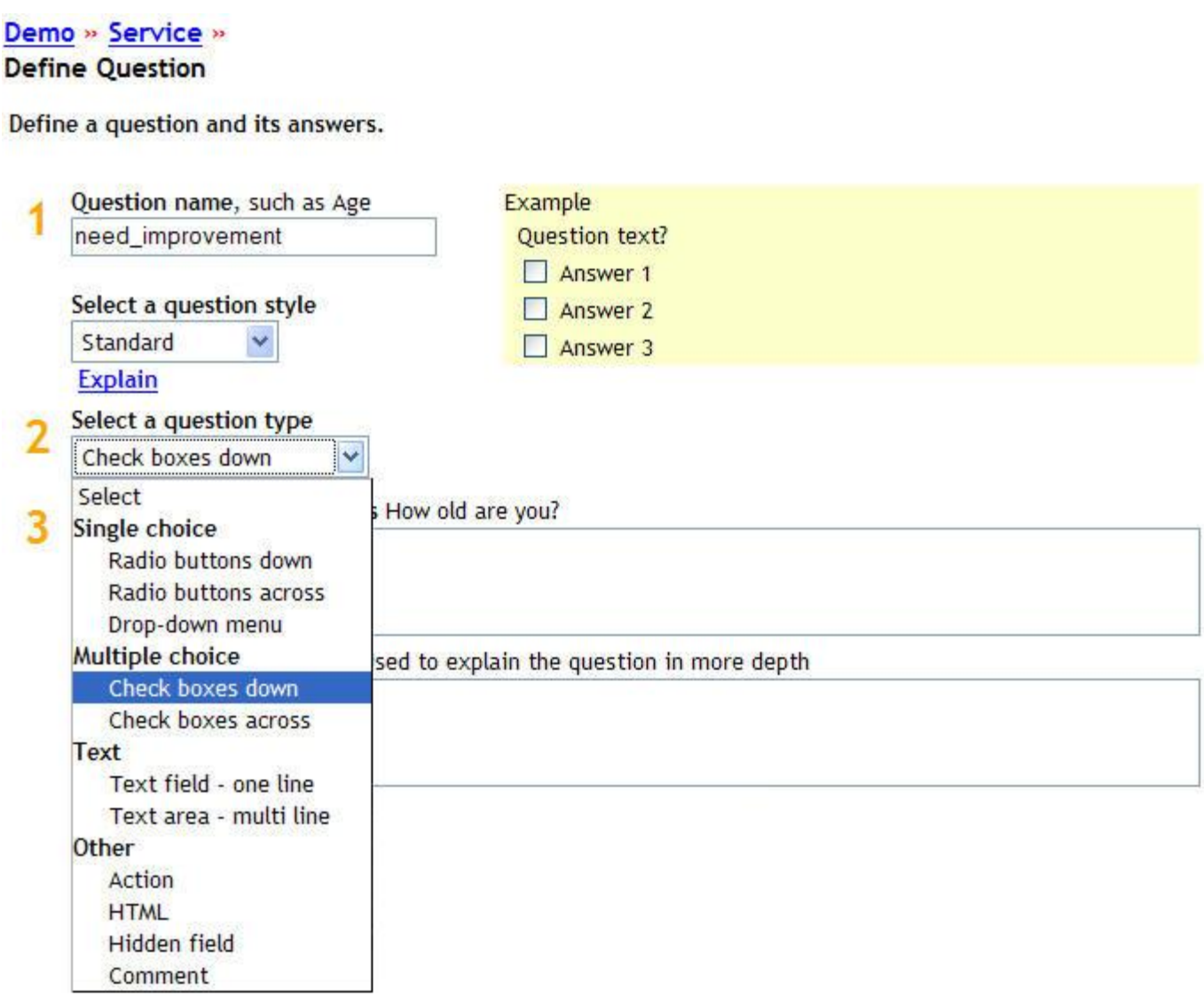
When a new question is created, the page looks like this:



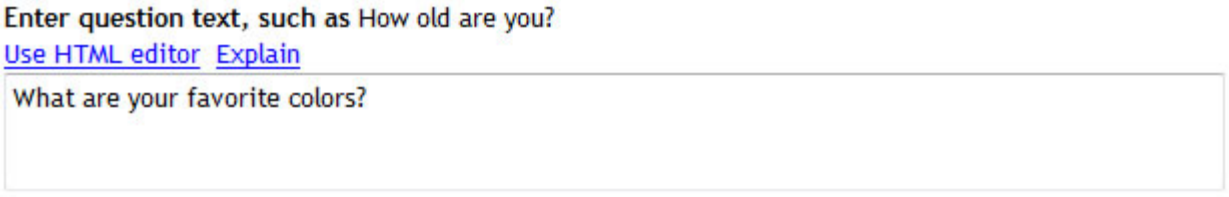
Every question must have a "Question name". Besides identifying the question, this name is used as the column name when data is saved in a database table. Question names can only use letters, numbers, and an underscore (_) and can be up to 30 characters long.

The "Select a question style" menu determines the appearance and behavior of the question. The Standard question style is used most of the time, and allows creating all standard questionnaire and form elements. Other **Question Style**s allow displaying a question in a particular format or using a specific behavior, such as verifying that the text entered is a valid number. To see what other styles do, pull down the menu, pick a Question Style, and then click on "Explain" to get specific instructions for how the style works; some of them display an illustration on the right hand side of what they look like. For more information, see **Question Style**s.

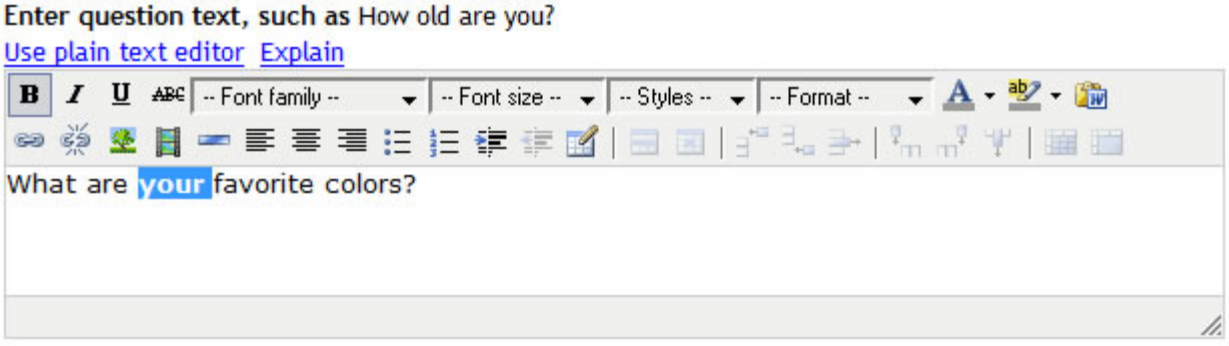
The "Select a question type" menu determines its kind: radio button, check boxes, drop down, text field. Cycle through the question types with the keyboard or mouse to see a sample of what each type looks like on the right, like this:



The "Question Text" field is where the question to ask is entered, such as "What are your favorite colors?", as shown below. There is no limit to how much text you can enter; the field will become larger automatically as you enter more text.

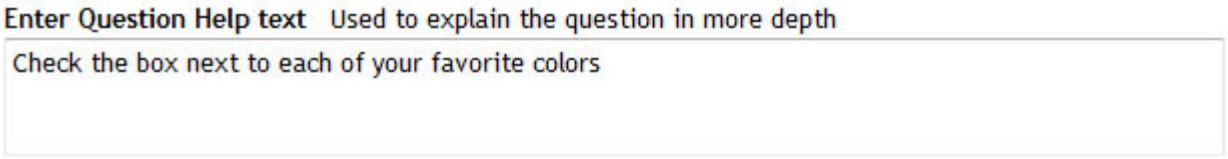


Clicking on "Use HTML Editor" activates the **HTML Editor**, which is being used here to display the word "your" in bold type:

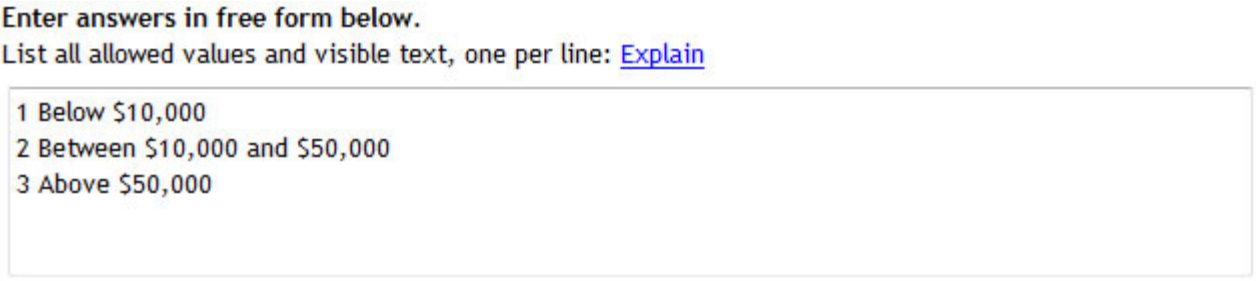


When using the HTML editor, you can enlarge it by dragging its lower right corner.

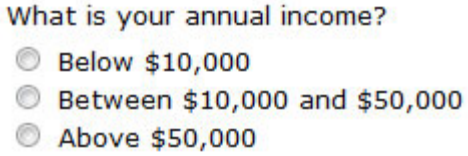
If Question Help is selected in the General settings, a "Question help text" field is available. Text entered here will be shown when the respondent clicks on the [?] help link. When not using Question Help, this box is not present:



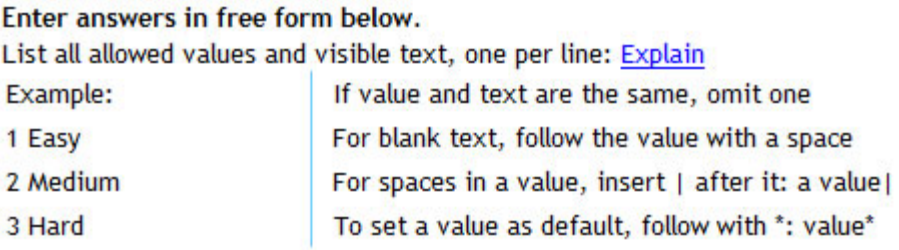
Several question types require specifying each possible valid answer. They are radio buttons, drop-down menus, and checkboxes. When using a radio button or drop-down question, this section of the page looks like this:



In the above example,there are 3 valid answers: 1, 2 and 3. When the questionnaire data is saved, the values 1,2,3 will be written to the database. The questionnaire that respondents see will not show 1,2,3; instead, it shows:



Clicking on the Explain link brings up a quick reminder of how to fill out this section:



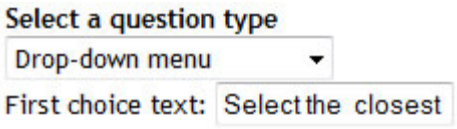
Rules for how to enter answers:

- Enter one value per line; blank lines are ignored
- Begin each line by entering the allowed value, which can be a number or text. When creating scales, or if the question will be analyzed numerically, enter numbers rather than text.
- Next to each value, separated by a space, the text to display can be entered. If no text to display is entered, and only a value is entered, the text displayed will be the same as the value. For example, 1,2,3 will display as 1,2,3.
- However, it is possible to display nothing for a particular value, by following the value with one extra space. This is useful, for example, when creating scales where only the first and last values in the scale matrix should be shown.
- Usually, values are a number or a single word of text. Some special applications might call for values with multiple words. In that case, separate the value from the text using the | character, like this:
  cherry vanilla|Ben & Jerry's Cherry Vanilla
- To make one of the answers the default, add an asterisk * to the value, like this: 2* Medium. If using the | separator, use:
  cherry vanilla*|Ben & Jerry's Cherry Vanilla

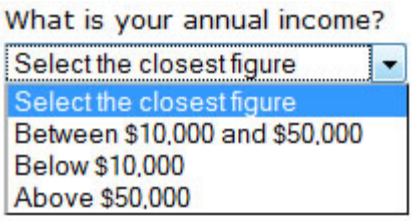
In a question of type checkbox, it is recommended to fill the Value column with meaningful alphabetic codes. For example, if asking about favorite ice cream flavors, enter a short abbreviation on the left hand side, such as choc, strawb, vanilla, and the full text that respondents will see on the Full text column, such as Chocolate Champagne, Strawberry Smooth, Vintage Vanilla. Database columns will be named using the name of the question. For example, if a question is named

"need_improvement", then the column names will be need_improvement_choc, need_improvement_strawb. need_improvement_vanilla (this is the reason for keeping the values short). Values must be made up only of letters, numbers and underscore (_).

Questions of type Drop-down menu include an additional field for entering the text to show in the drop down. By default, this text is "Choose one". For example:

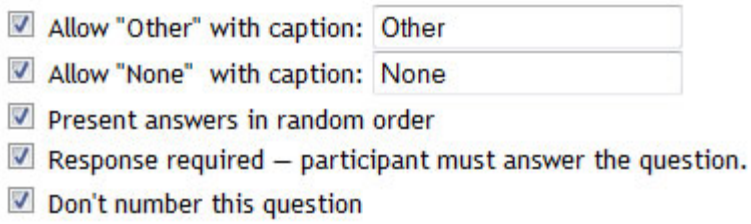


When the questionnaire is presented, the menu will be:



In addition to specifying each value in a drop down field, the contents of a drop down menu can be set dynamically for each respondent based on previous answers or on database lookups. See **Dynamic Drop downs.**

Additional fields



Some question types use an "Other" option, which adds an extra button and a text field for entering other values, using a caption. The caption must be entered explicitly. For screen readers for the visually impaired, add a label for the reader to read by using a | before the reader text: Other|Enter other value. The reader field is not visible and is only read by screen readers.

Radio buttons also allow a "None" option, which adds an extra button with a caption; when a respondent select None, a blank value is entered as a response.

The "Present answers in random order" box, displays the answers to different respondents in a random pattern.

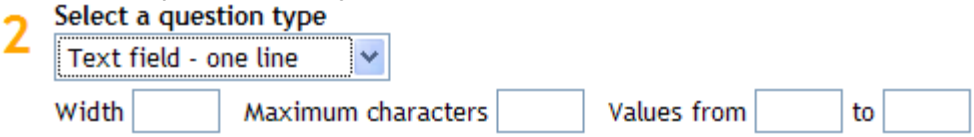
The "Response Required" box will require the respondent to make a choice in a multiple-choice question, or enter data in a text field before proceeding. If they do not, the entry will be rejected, as described in **Data Validation**.

Check "Don't number this question", present only when using question numbering, to avoid giving a number to this question. Question numbering will resume with the next question that doesn't have this box checked.

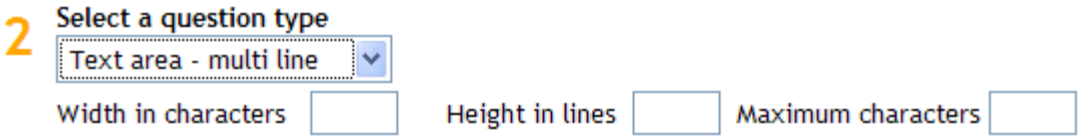
Extra fields are **explained below**.

Other question types do not present the Answers field.

A Text Field question can adjust its width, the maximum number of characters accepted, and a range of acceptable values:



A Text Area question can adjust its width, its height, and the maximum number of characters entered in the box. A safe limit is 2000, which is accepted by all databases. See **Options / Save** for additional information on how to capture very large text fields.



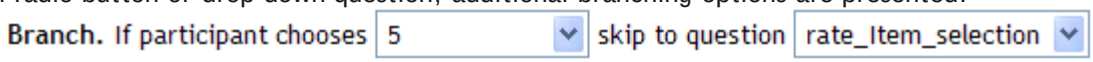
Questions of type Comment provide a way to enter a note about how the questionnaire is constructed, about the questionnaire content, or anything else. A comment is only visible to the questionnaire designer and is never seen by respondents.

Questions of type HTML allow entering text and optionally HTML. This can be used for many advanced applications, including **Question Piping**, and for visual effects, such as a horizontal rule, created by entering <hr>. The **HTML Editor** is very useful for these kinds of questions..

Questions of type Hidden field are used to store information gathered by **pre-populating fields**, by calculations made using the **Script** action, and by values retrieved from a database using the **SQLSelect** action. They can also be set to a value by opening the Answers box with the Edit Answers link that appears below the Submit button, and entering a default value in the first line, like this: value* or like this: A multi word value*|

Questions of type Action specify special processing, such as calculations and complex branching. See **Actions** for a full explanation of this very powerful capability.

In radio button or drop down question, additional branching options are presented:

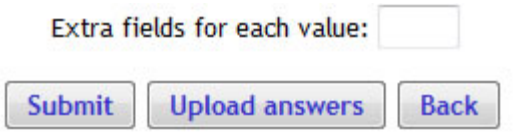


The "If participant chooses" drop down menu allows selecting a value. When a respondent chooses with that value and submits his response, the questionnaire will continue on the page that holds the question selected in the "skip to question" drop down menu. Questions are organized into pages, which are created by adding page breaks in the visual editor and the list editor. It is best to create all questions first and then edit the questions where branching is possible to add branching logic. This option allows for simple branching. For more complex cases, see **Branching**.

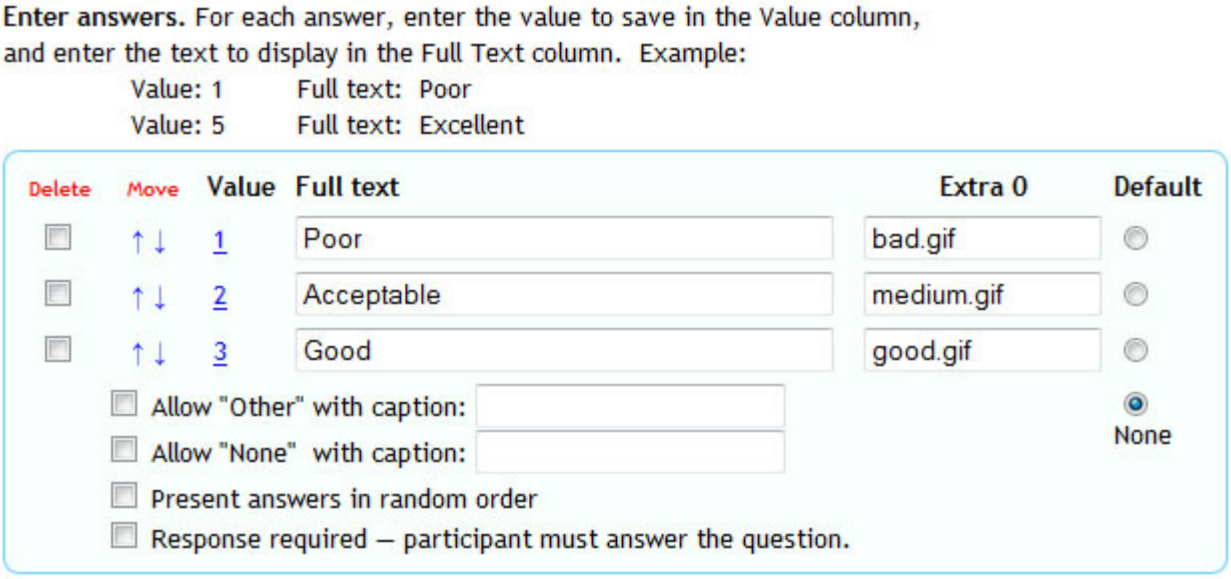
After completing the fields in the form, press Submit to create or revise the question.

Advanced features

Extra fields
When a question of type radio button, check box or drop down menu is created, it is possible to specify a number of Extra fields for each value at the bottom of the page:



After entering a number, press Submit and an Answers box contains extra fields that can be filled out, like this:



In this example, the extra values are used to specify an image that will be shown next to each answer. Other uses are speciying audio or video files for playback, and in general for associating an extra data item with each value. To use Extra values, a specialized Question or Form **Style** must be used.
Extra values are also useful for identifying correct and incorrect answers in tests and quizzes. Extra values can also be used with **Question Piping**.

Uploading answers
If there are a lot of values to insert as answers, such as a list of department codes, this option can be used to set values, full text, and the content of extra fields by uploading a comma delimited file. The file should contain the fields in that order, separated by commas. Each field should be surrounded by double quotes. To use a double quote inside a field, use two double quotes:
"1","Poor ""man"" of La Mancha "
Instead of entering answers and values in the Answers box, press Upload Answers, browse to the file, and press Upload. To upload answers with Extra fields, specify how many extra fields to use and press Upload Answers without pressing Submit. The upload answers button and extra fields are only presented when a question is being created.

In addition to specifying each value in a drop down field, the contents of a drop down menu can be set dynamically for each respondent based on previous answers or on database lookups. See **Dynamic Drop downs.**

Next: List Editor

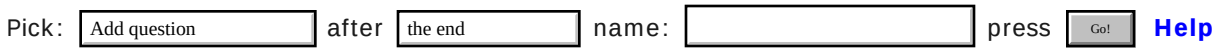
Design Questionnaire - List Editor

The List Editor lists all questions in the questionnaire in tabular form.

The "Name" column shows a link to the question. The "Text" column shows the question's text and an indication of a Page Break if that question is the last on a page. The "Style" column shows the name of the Question Style used for that question.

Clicking on a question's name allows **editing** that question. Clicking on a question's text switches to the Visual Editor, positioned at that question.

The descriptions below refer to the following settings at the top of this section:



To add a question, pick Add question, enter a name and press the Go! button. A question name should be brief yet descriptive, such as "age", "fulfillment_satisfaction". Use only letters, numbers, space, and underscore (_) in question names. Besides identifying questions, these names are used as column names in the database table where the questionnaire is saved. To insert a question after a specific question, or at the beginning, use the second drop-down and select a question; the new question will be inserted right after the selected question.

Use Add page break to indicate the end of a page and the beginning of a new page. Use the second drop down menu to point to the question after which the page break should be added, and press Go.

Use Create matrix to start the **Matrix Wizard**, which allows creating matrices of questions that have similar structure, including Likert scales.

Use Create grid to start the **Grid Editor**, which allows creating a grid of similar questions repeated multiple times, such as financial data over multiple years.

See **Question Library and Clipboard** for an explanation of those items.

To remove one or more questions, check the Select check box next to each question to be removed, pick Remove selected, and press the Go! button.

To move one or more questions, check the Select box next to each question to move, and pick Move selected. Open the second drop-down menu and choose the question after which the selected questions should be moved to. Press the Go! button.

To copy one or more questions, check the Select check box next to each question to copy, and pick Copy selected. Open the second drop-down and choose the question after which the selected questions will be copied to. When copying a single question, enter the name of the new question in the name field and press the Go! button. When copying more than one question, enter a "Question renaming mask" in the name field and press Go!. Masks are the following:

Mask	Does this	Example	Renames	To
>tail	appends 'tail'	>_morning	importance f1	importance_morning f1_morning
<head	prepends 'head'	<Appearance_	importance f1	Appearance_importance Appearance_f1
old=new	replaces 'old' with 'new'	appearance=performance	appearance_importance	performance_importance

To rename one or more questions, check the Select check box next to each question to rename, and pick Rename selected questions. If renaming only one question, enter the new name in the name field; if renaming more than one, use a renaming mask, as defined in the table above.

Other useful editing tools are the Preview button on the left hand side, which gives an even clearer picture of the finished questionnaire, and the Index button, which gives a summary listing of all questions, which is specially handy when creating branching scripts.

Next: Setup response page

Response page

This page specifies what to display after the responded completes filling out the questionnaire.

1 If you choose "Display results page using settings on this page", a page will be constructed dynamically from your settings.

If you choose "Redirect to", the visitor's browser will be redirected to the specified URL after voting. If you specify a URL, the visitor's browser will go there. If you enter *, the browser will return to the same page as the voting form (the 'referer'). If you enter *?x=y, ?x=y will be appended to the URL of the referring page. This can be very useful with the Web Services API. The remaining items on the page do not apply if you choose this option.
Note that this option cannot be used with questionnaires embedded in a portal using the Cogix Survey Portlet .

2 Enter a title, page header, page body, and page footer text. These fields can be left blank. Page header, body, and footer can include HTML, using the HTML editor. The image used in the Questionnaire, if any, is also used automatically in the Response page. It is customary to include a message such as "Thank you for participating" in the Body field.

3 This option allows presenting respondents with a live tally of results, as is commonly done with online polls. Click on Show Questions and specify which questions should be tallied, and choose the options to present counts, percentages, and a bar graph.

For each question to be displayed, specify whether to display the question's name or the full question text by checking the respective boxes.

For each answer to each question, use the corresponding checkboxes to specify what elements to display: the value, the full text, the count of how many people have chosen that answer, the percentage of people who have given the answer, the average value of the responses (meaningful only when the values are numeric), and a percentage bar between 1 and 100 pixels wide.

The drop-down menu allows selecting whether to display the answers in the order in which they were asked, or most popular or most recent first (or last). Choosing "most popular on top" allows constructing top 10 lists, for example. Choosing "most recent first" allows constructing discussion-board style presentations.

4 After making all changes, press Submit. Then click on "Preview the response page". This will show how the results display will look, using the Response Style Template in use in this questionnaire's Place. Note that all the numbers, percentages and sizes of color bars will be random (percentages will not add up to 100%, for example).

After previewing the page, choose whether to continue by Publishing the questionnaire or whether to set additional options to non-default values. Most of the time, the default values are appropriate. We suggest that you continue reading about Publishing first and then continue with the options.

Note: changes in the Setup Response page do not take effect until the questionnaire is Published.

Next: Publish

Publish

Until this point, questionnaire definitions have been stored in the application and have been previewed in the browser, but nothing has been published that can be used on the web site. This page is used to specify when to make the questionnaire available on a web site.

1 Enter a Starting date for when the questionnaire will be available live. On the Ending date, the questionnaire will no longer be available and responses that are late will be discarded. Settings on the Security page specify what will happen with such responses.

The dates can include a date, a time, or both. The formats of each are as follows:

Dates: MM/DD/YY (use 00 for 2000).
Times: 3:35 PM, 15:35, or just 7 or 15 (for 7:00 AM and 3:00 PM).

If the time is omitted, publishing will happen one minute after midnight. If the date is omitted, publishing will happen at that time today. Dates and times are in the format and time zone of the application server. A user can change the date format and time zone by using **My ViewsFlash**.

You can also enter the word "now", or press the Now links to indicate starting and ending times. If the questionnaire conflicts with a previously scheduled one, you will be prompted to override; if you do, the conflicting questionnaire will be rescheduled automatically.

When a questionnaire is published, it is available during the designated period. Sometimes you may have a period of testing before going live. By returning to this screen, you can set or change publication dates at any time.

2 After publishing the survey, it is likely that it will be first taken by a set of people who will test it and perhaps it will be refined further. Between testing rounds, and before going live, check the "Delete all data" box to delete test data, information about who has participated and who has been invited, and to reset tabulations to zero. If a survey is open, and data is collected, and the survey designer modifies the questionnaire in a way that requires a modification to the database tables (such as adding a new field), the survey will be closed until it is Published again, which will alter the table appropriately.

In a multi-page survey, when the survey is closed, there may be people who have completed the survey partially. Normally, those partial results are saved; checking the "Discard partially completed responses" box throws away partially completed surveys.

When **Staging** is enabled, two additional checkboxes appear:

- ☐ Lock (no changes can be made until unlocked)
- ☐ Stage to production server now

Lock will prevent any modifications to the survey until Lock is unchecked from this page. After locking a survey, it's a good idea to perform final tests, and when successful, then remove all test data. If the test is unsuccessful, unlock the survey, make the changes, and lock it again.

Stage will write appropriate records to the staging area so that the survey can be moved from a staging server to a production server. This operation stages all items necessary for the operation of the survey, as well as any Styles and Invite Lists that the survey uses that haven't been previously staged or that have been modified since being staged last.

When staging, the Lock box should be checked. If a Style or Invite List is modified after a survey is locked, staging will be canceled, and the survey must be unlocked, locked again, and retested before staging it again. The purpose of this is to prevent a situation where a Style or an Invite List is changed after the survey was locked.

These two boxes are only available to a user who has a Staging Access Right. In a high-security environment, these users should not be the same as users with the right to Create Surveys. In this way, the survey creator is responsible for configuring the survey, but only users with Staging rights can lock, unlock, and stage the survey. This grants the Staging user control over when a survey is published, and assures that once a survey is locked it cannot be modified without her consent.

If Staging is not enabled, these two checkboxes do not appear.

3 The questionnaire URL is shown in a highlighted box. Clicking on that URL will pop-up the first page of the survey. This is the address of the questionnaire, and it is the URL to use to link from other web pages, to send in e-mail messages, etc., and to bring people to the first page of the questionnaire. When embedding the questionnaire in another page, of course, this is

not its URL; this is the URL that will be used with the embedding system.

When choosing question authentication in **Security**, the URL is appended with the additional parameters required to acess it, such as &id=. A unique identifier must be appended to the URL for each respondent; to have ViewsFlash automatically assign one, use $*,like this: &id=$*.

When using captcha authentication by setting the session value to CAPTCHA in **Security**, the URL is changed to include /ViewsFlash/start.jsp?... other parameters, which is the URL required to present the CAPTCHA validation page first.

If you are using Invite lists, after the survey is open click on the Invite link to e-mail the invitations.

In a non-database installation, or when writing HTML pages to disk has been enabled, the actual file system locations of the created HTML files are saved are shown.

At the bottom of the Publish page is a URL for displaying a one-page printable version of the questionnaire, suitable for printing, without buttons or progress bar.

Next: My ViewsFlash

My ViewsFlash

The MyViewsFlash page allows you to set certain personal preferences. These preferences are stored in your browser using a cookie. If you use a different browser, you will need to reset your preferences.

The My Locale option allows you to choose the format of the dates and times used for publishing questionnaires. For example, using the English locale, dates are viewed in this format: **6/29/06 1:50 PM**. If using the Chinese locale, dates are shown in this format: **06-6-29 下午1:48**. If you don't see a Locale that is applicable to you, use Custom to create your own Locale by specifying a language code and a country code. For example, in Turkey the language code would be tr and the country code would be TR. You don't have to specify a country code. The application server must be configured to support your locale.

Note that the My Locale setting does NOT affect in any way the format of dates or numbers used and stored in questionnaires.

The My Time Zone option allows you to adjust the dates and times used for publishing questionnaires to your local time zone. If the application server is hosted in New York, for example, and you work in London, the application server uses US/EST time and you use UK time. Select the UK time zone and all date and time displays will be adjusted to your local time zone. This setting does not affect the actual time of publication in any way; it simply displays the actual time translated into your time zone. For example, if a questionnaire is due to open at 8 AM New York time, you will see that time in London as 1 PM, and in the US PST time zone as 5 AM.

The My ViewsFlash level option allows you to see more or fewer options, depending on your level of expertise in using ViewsFlash. If you choose a usage level and don't see the functionality that you are looking for, change to a more advanced level. The default setting is Advanced.

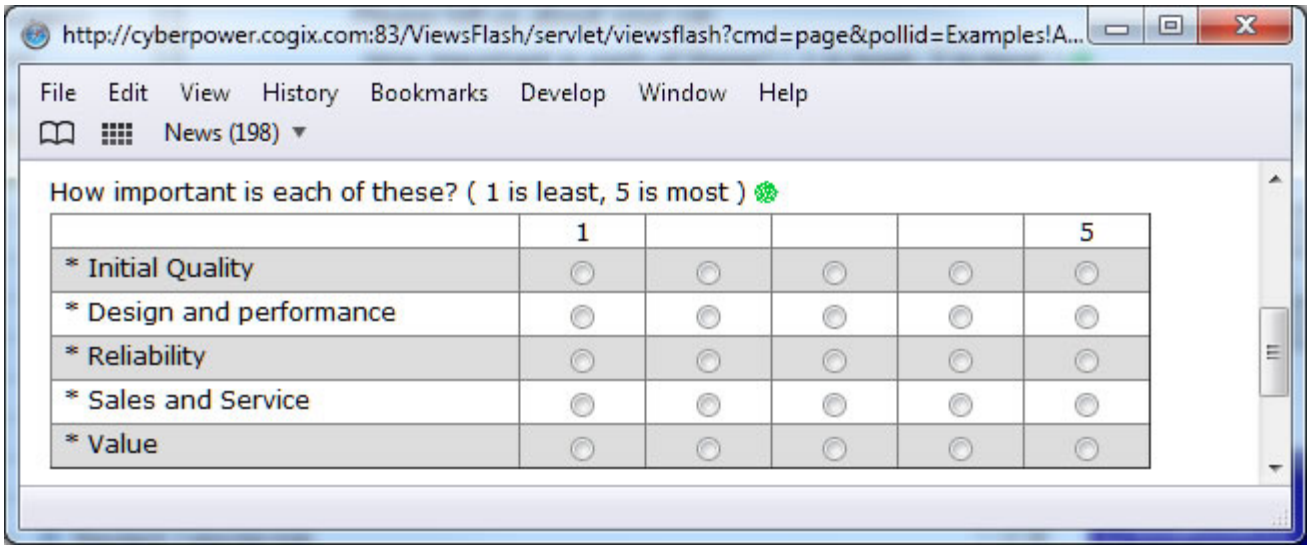
Next: Layout

Arranging questions in a layout

The Layout Style uses scripts that enhance questionnaire layout.
Note that the Visual Editor does not show the effect of these scripts. Use Preview and test-run the questionnaires to see the exact effect.

To enter these scripts, create a question and choose style Layout. In the Script field, enter the appropriate scripts, one per line.
The question **must be located BELOW** the questions affected, on the same page. The scripts only affect questions on the page where they are located.
Create a Practice questionnaire by copying the car_satisfaction2 questionnaire from the Examples place to experiment with these Scripts.

To alternate colors in tables:

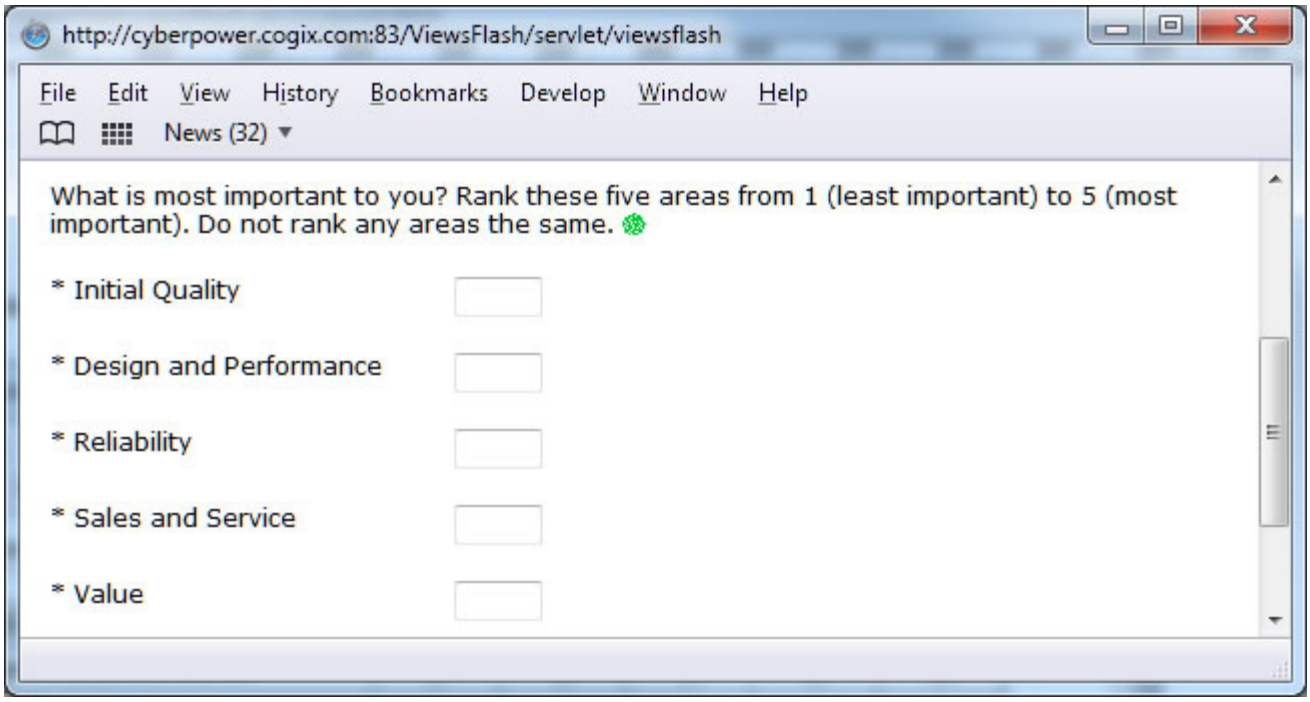


Use this script:

```
CogixColorRows ( '#FFFFFF','#DDDDDD');
```

This script renders all tables on the page in white and grey. Any colors can be used.
To color just one table, look up the table element's ID using View Source and add it to the parameters, like this:
CogixColorRows ('#FFFFFF','#DDDDDD', 'matrixExamplesAdvancedlayoutimp_')
A fourth parameter can specify the table elements class; use null instead of a table ID in that case.

To place text on the same line as the input field:



Use this script:

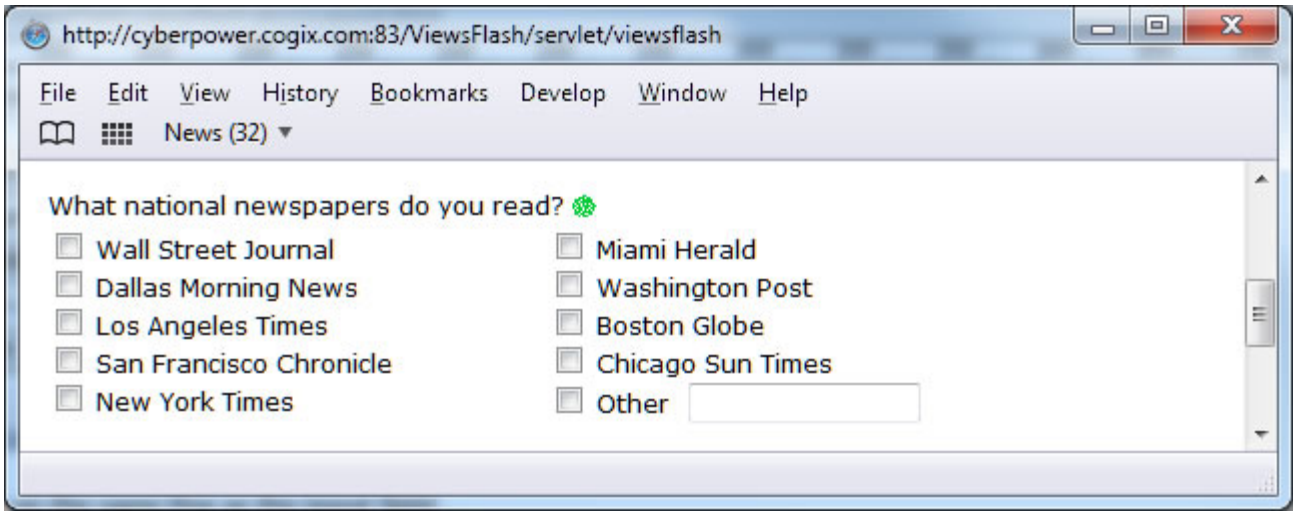
```
CogixLegend ('questionname',200);
CogixLegendAll (200,'t');
```

The first script form places one question's input field to the right of the question text, 200 pixels away.

The second script places all input fields on the page to the right of the question text, 200 pixels away.
The second parameter can be omitted. In that case, all single line text fields, drop-downs and text area questions on the page are aligned.
If present, its meaning is to align:

- t for all single line text fields
- r for radio buttons
- c for checkboxes
- s for select drop-downs
- a for text areas
- x for all elements

To display answer choices in multiple columns:

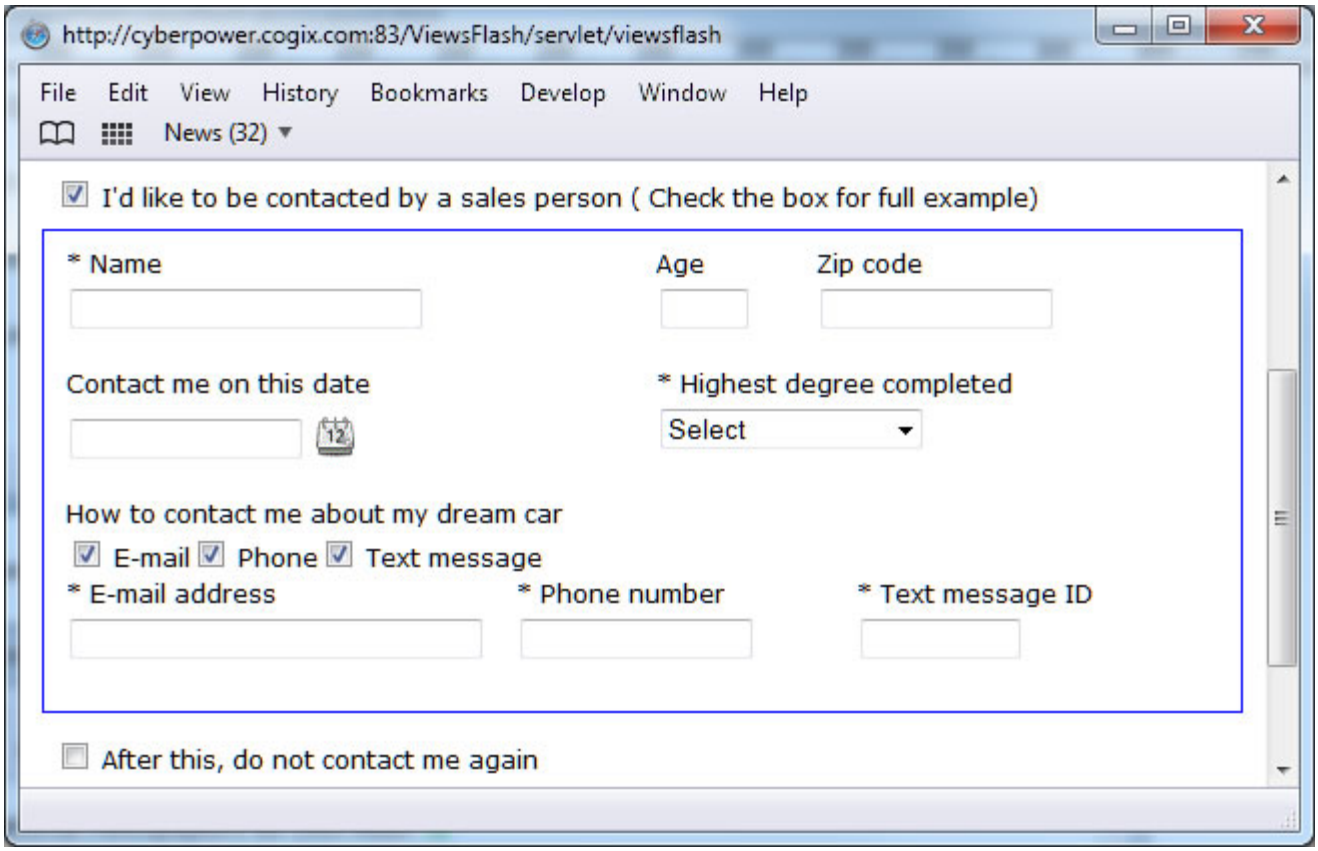


Use this script:

```
CogixAnswerColumns ( 'newspapers',250 );
```

This script displays choices for question 'newspapers' in columns that are 250 pixels wide.
The smaller the number of pixels, the more columns there will be, but if the number is too small, text will wrap and will not look well placed.

To place questions anywhere on the page and create a two dimensional layout:



Use this script:

```
CogixPosition ('full_name',5,0);
CogixPosition ('age',300,0);
CogixPosition ('zip',380,0);
CogixPosition ('contact_when',5,60);
CogixPosition ('education',300,60);
CogixPosition ('how_to_contact',5,125);
CogixPosition ('email',5,165);
CogixPosition ('phone',230,165);
CogixPosition ('textmessage',400,165);
CogixLayout ('full_name','textmessage',240, null, 'border:solid 1px blue;');
```

The CogixLayout script creates a rectangular area around all questions starting between, and including, the questions named in the first and second parameters.
The 240 is the height of the rectangular area, and null is the width of the area; setting it to a number will make it an exact number, while using null makes it adjust automatically.
The last parameter is an optional CSS style that is applied to the rectangular region. In this example, it draws a thin blue line around it.

Each of the CogixPostion scripts places the question whose name is in quotes at the given location, in pixels, within a rectangle.
The first number is the number of pixels from the left side and the second number is the number pixels from the top.

Technical note: the scripts are JavaScript functions that are predefined and available in questionnaires.
You can define your own javascript functions and code using the Layout style.
Do not insert <script> tags; they are inserted automatically for you.

Next: Matrix Editor

Matrix Wizard

The Matrix Wizard provides an intuitive way to create matrices of questions that have similar structure, including Likert scales. Here's an example:

Thanks for visiting our store. Please tell us about your visit.

Rate each of the following, from worst to best. [?]

	Worst				Best
Service [?]	○	○	○	○	○
Price [?]	○	○	○	○	○
Item selection [?]	○	○	○	○	○

1 Begin by providing a name for the matrix. The matrix will be created as a series of questions; all questions in the matrix will begin with this name, followed by an underscore.

From the dropdown menu, select a matrix style, and a description of it will appear below.
For a Likert scale, choose Matrix; for a Semantic Differntial scale, choose Matrix_SD.

A standard Matrix can use radio buttons or checkboxes; select which. A Semantic Differential matrix can only use radio buttons.

If questions are being numbered, a checkbox allows not assigning a question number to the matrix.

2 A text field is provided for entering explanatory text which will appear above the matrix. If question help is being used, a text field is provided for that as well.

In the next text field, enter the value and text to show for each choice. For example, to enter 5 values labels Low, Medium, High for values 1, 3, and 5, enter:

```
1 Low
2
3 Medium
4
5 High
```

Note that after 2 and 4, an EXTRA SPACE must be entered. This signals that the corresponding text to show should be left blank.

In a Semantic Differential matrix, only the values are used.

3 In the next section, a text field is provided for entering a name for each question, the text of the question, and a place for question help. Question names are used to name the fields in the stored data; the text of the question is what the participant will see. Space is provided for five questions. If fewer or more are needed, click on the Fewer questions or More questions links as appropriate.

If a participant must answer a question, check the box in the (*) column. To avoid numbering a question, check the box in the No# column. To reorder a question up or down, use the ↑ and ↓ arrows, and use the X to remove a question.

In a Semantic Differential Matrix, enter in the question fields the text for the opposites of the scale, separated by the | sign. For example:
Appealing|Unappealing
Fresh|Stale
Reasonable|Overpriced

After filling out all the entries, press the Preview button, which shows what the matrix will look like. Press Continue. When satisfied, press the Create Matrix button. A series of questions is created, each with a unique name prefixed by the matrix name and an underscore. After creating the matrix, individual questions in the matrix can be edited further as needed.

The appearance of the matrix is automatically designed so that columns are of equal width. For greater control over the appearance of a Matrix, open the first question in the matrix by clicking on the explanatory text above the matrix in the Visual Editor, or in the first question in the Matrix in the List Editor. This question, which is of type HTML, will have additional options available for setting the width of different matrix elements, in pixels,as well as an option for using a custom CSS style. It also includes a field "Use as report text" which makes the explanatory text available in Univariate reports. Click on the Explain button below the Question Style on the first question of the Matrix for further explanation of these options.

Next: Grid Editor

Grid Editor

The Grid Editor provides an intuitive way to create grids of simlar questions of any type and style. There are two kinds of grids: fixed size and variable size.

Fixed size grids

Here's an example of a fixed size grid, where the number of rows is the same for each section:

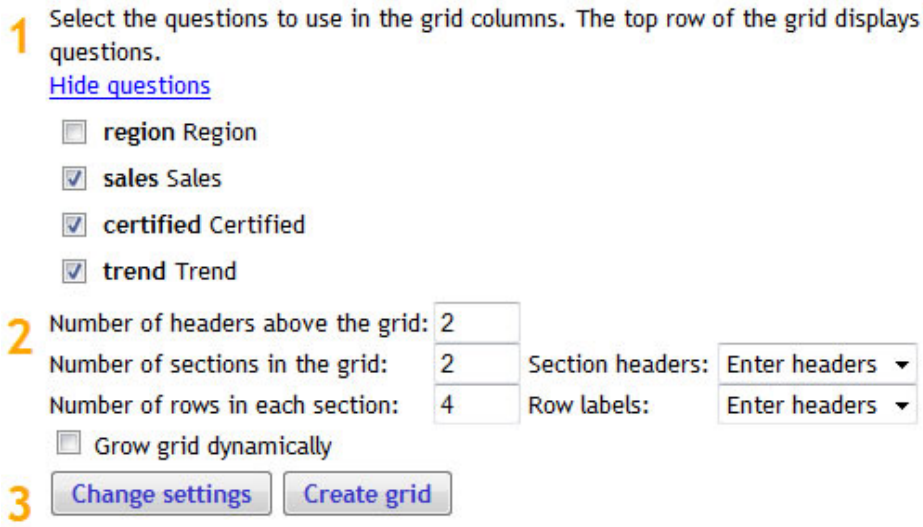
Performance summary			
Area	*Metrics*		
	Sales	Certified	Trend
Domestic			
North		☐	Select
South		☐	Select
East		☐	Select
West		☐	Select
International			
North		☐	Select
South		☐	Select
East		☐	Select
West		☐	Select

In this example, there are three grid columns made from three questions.
The columns are repeated in four *rows*, which have ***row labels*** of North, South, East and West.
The entire grid is repeated in two ***sections***, each of which has a ***section header***, Domestic and International.

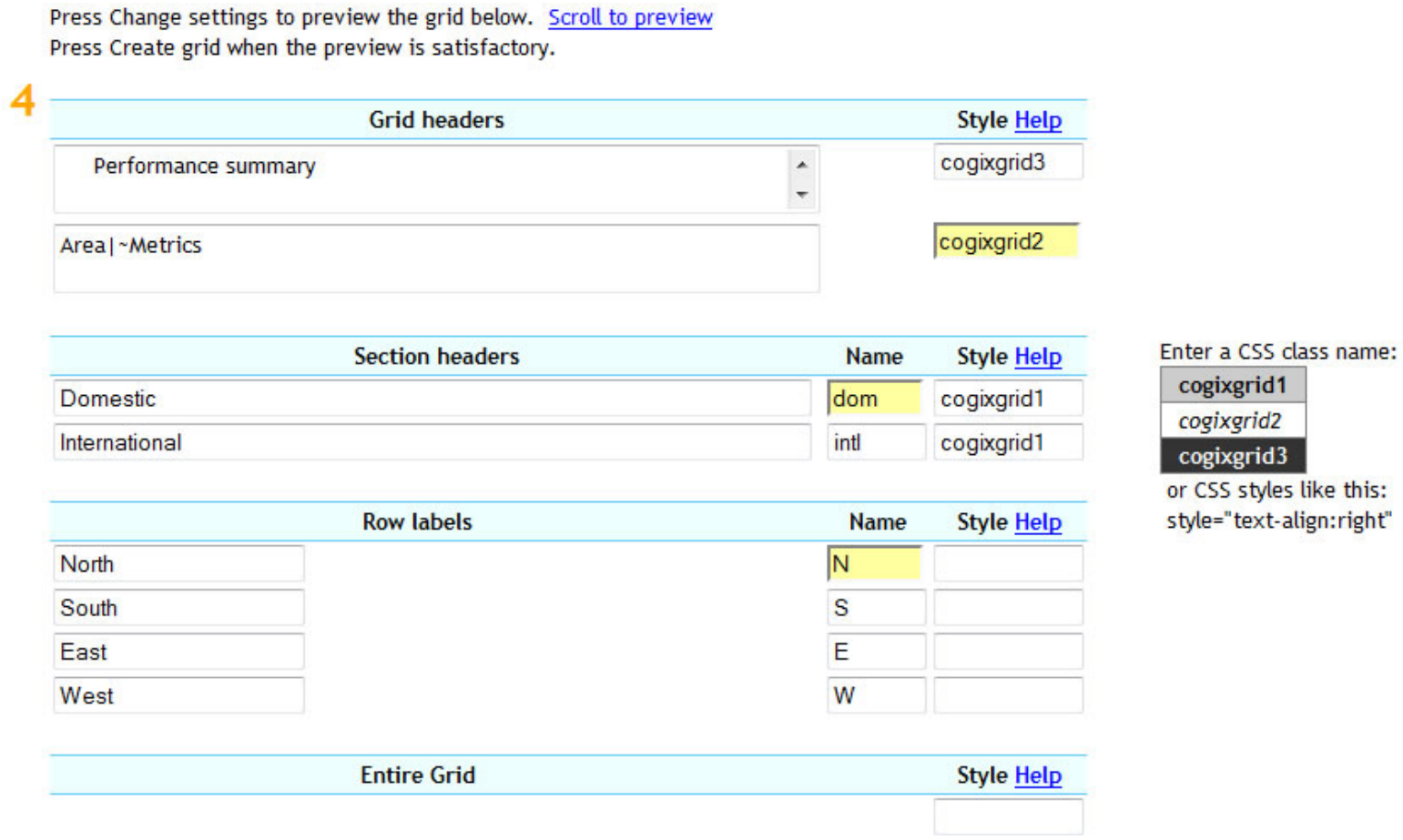
The first two rows of the grid are ***grid headers***, rendered in distinct ***CSS Styles***.
The third row includes each ***question's text***, Sales, Certified,Trend.

In this example, instead of manually creating 24 questions, it is only necessary to create three ***prototype questions***, with names such as sales, certified, and trend.
Then, the Grid Editor is used to create a grid with two sections of four rows each and appropriate headers; when the Grid is created, 24 questions are created and the original three prototype questions are removed.

After creating the prototype questions, select the Grid option in the Visual Editor. Enter a short name for the grid, such as "fin", and fill out the fields as follows:



In the first step, the three questions to use are selected by checking the boxes next to each.
In the second step, the number of headers, sections, and rows in the grid are specified,and the kind of headers.
After pressing Change Settings, a new section allows entering the specific headers, row labels, ***Section Names*** and ***Row Names***:



The two Grid Headers provide text to show at the top of the grid. The Section Headers provide text to show above each of the sections. Row labels are displayed on the left side of each grid row.

The second grid header uses the symbols ~ and |. The | indicates a visible cell boundary, and the ~ indicates an invisible cell boundary. That produces the useful column spanning effects in the header.

The Style column contains CSS style names cogixgrid1, cogixgrid2 and cogixgrid3. To see what these names do, click on the Help link right next to the respective Style column. In this illustration, the styles available for section headers are shown.

The Name column contains short names for each section and each row. The 24 questions created have question names made up of a combination of the grid, section, and row names, such as:

fin_sales_dom_N	Sales Domestic North	fin_sales_intl_N	Sales International North
fin_certified_dom_N	Certified Domestic North	fin_certified_intl_N	Certified International North
fin_trend_dom_N	Trend Domestic North	fin_trend_intl_N	Trend International North
fin_sales_dom_S	Sales Domestic South	fin_sales_intl_S	Sales International South
fin_certified_dom_S	Certified Domestic South	fin_certified_intl_S	Certified International South
fin_trend_dom_S	Trend Domestic South	fin_trend_intl_S	Trend International South
fin_sales_dom_E	Sales Domestic East	fin_sales_intl_E	Sales International East
fin_certified_dom_E	Certified Domestic East	fin_certified_intl_E	Certified International East
fin_trend_dom_E	Trend Domestic East	fin_trend_intl_E	Trend International East
fin_sales_dom_W	Sales Domestic West	fin_sales_intl_W	Sales International West
fin_certified_dom_W	Certified Domestic West	fin_certified_intl_W	Certified International West
fin_trend_dom_W	Trend Domestic West	fin_trend_intl_W	Trend International West

These question names are used in reports and as column names in the database that holds the data, so it is helpful to use meaningful abbreviations.

After pressing the Change settings button, a preview of the grid will appear at the bottom of the page. Experiment with different settings and headings, pressing Change settings each time to preview the result.

When the preview is satisfactory, the Create Grid button will remove the original three prototype questions, and 24 new questions will be created. Additional questions that hold the necessary HTML for displaying table cells and headers are also created.

The grid definition becomes the first question in the grid. After the grid is created it can be revised, by opening the grid quesstion and clicking on the Edit Grid button. This will remove all the created questions and will restore the removed prototype questions. After undoing a Grid, the protoype questions can be revised, different questions can be chosen, and all the headers and other options can be changed. When the preview is satisfactory, Create Grid will once again remove the prototype questions and create the individual questions that make up the grid.

Variable size grids

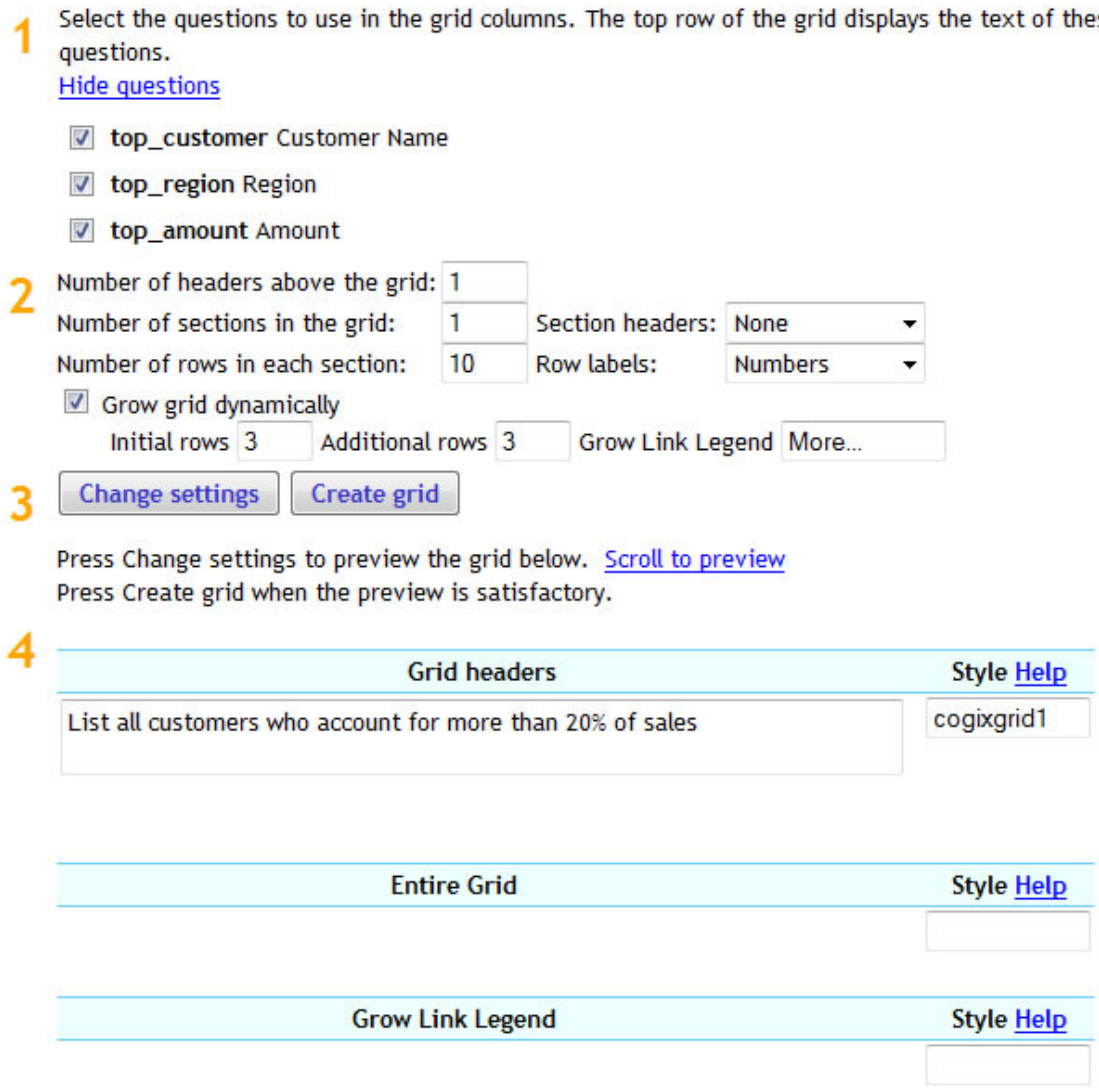
Here's an example of a variable size grid, where a maximum number of rows is created, but it is expected that not all rows will be filled out. This shortens the amount of space required to display a grid considerably.

List all customers who account for more than 20% of sales

	Customer Name	Region	Amount
1		Select	
2		Select	
3		Select	

More...

This is example is created with these settings:



Note that "Grow grid dynamically",right above the Change Settings button, is checked. Three new options become available, which allow setting number of rows initially visible, and how many more rows to show each time that the More link is clicked on. An additional setting, the Grow Link Legend,can be used to modify the style of the More link.

When used in a questionnaire that allows returning to previous pages, or revising questionnaire data, or printing, each variable size grid will display as many rows as necessary to display all information entered.

Advanced Grid Tips

Keep grid, question, section, and row names as short as possible so as to avoid hitting the maximum of 30 characters for question names.This limit is in place because database column names are limited to this size. If the combined name exceeds this length, an adjustment will be made.

To provide additional Styles, the ViewsFlash system administrator can modify the /ViewsFlash/styles/Standard.css file. The Style field can also use inline CSS styles, such as: style="text-align:right"

If using this with the Entire Grid style, specify the class as well: style="border:none" class="cogixgrid"

While creating and revising a grid, it is permitted to modify the prototype questions, and to add extra questions. After pressing Change Settings,click on the Visual Editor link on the left side, modify or add questions, and click on the question where the grid is, which is shown like this:

[Grid best_]

This will open up the Grid Editor again; press Change Settings and the preview will be updated with any changes that were made to the prototype questions.

To add or remove questions from the grid, click on Select different questions in part 1 of the Grid Editor to make the question list visible, select different questions, and press Change settings.

The Grid Editor can only be used with the Standard form style or a style that is compatible with it.

When creating multiple Grids in one questionnaire, each grid must use different prototype questions; prototype questions cannot be shared among different grids.

Copying grids and using a Question Library

Grids and grid questions cannot be copied. However, it is possible to copy and paste a Grid. First, create the grid normally and revise it until satisfied with the results. Then Undo the grid, and copy the grid question and its prototype questions to the clipboard using the List Editor. Once the grid is copied to the clipboard, it can be pasted into the same questionnaire or other questionnaires.

Using this procedure, a Grid can be placed in the question library as follows:

- Create the prototype questions and the grid; add headers, create and test grid. When satisfied, Undo the grid.
- Using the List Editor, copy the (undone) grid and its prototype questions to the clipboard.
- Create a new questionnaire in the Question Library, using a descriptive name such as "Income grid". Using the List Editor, paste the clipboard into the new questionnaire.
- To use the grid from the library, use the Visual Editor or the List editor and include the grid from the Question Library; do not specify a name when including the grid.
- If the grid is included from the library only once, the original question names will be used. If the grid is included two or more times, all questions will be renamed to assure unique question names.

Next: Question Library

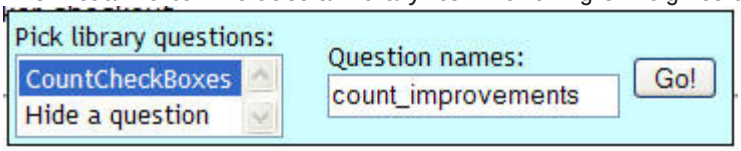
The Question Library and the Clipboard

Question Library Overview

There is often a need to create a set of questions and to reuse the set in many questionnaires. Places whose names begin with "Q_Lib" are treated specially in ViewsFlash. The questionnaires inside these places are listed in the Question Library menus. Each of those questionnaires contains one or more questions that are treated as a unit, or set of questions.

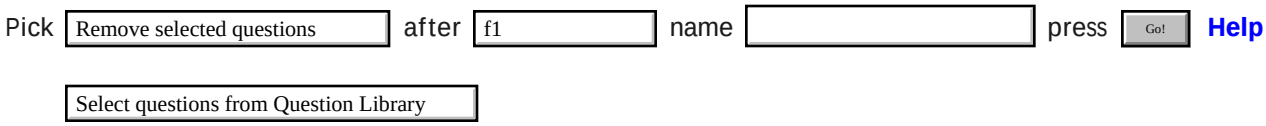
How to copy questions from the Library

The Visual Editor includes a Library item. Clicking on it gives the following:



The menu lists the questionnaires in the question library. By selecting one and pressing Go, all its questions will be copied to the current questionnaire.

The List Editor follows a very similar procedure. Pick "Insert questions from library" and a second drop down appears below it, showing Select questions from Question Library. Chose the questions from the second menu, and use the after menu to select where the questions will be copied:



Note that when copying a set of questions library, only the questions are copied. Option settings in the library are ignored.

Question names

The inserted questions will have the same names as the questions in the library. If those names conflict with question names in the current questionnaire, the inserted questions will be renamed. By providing a "name mask" in the name field, it is possible to rename the questions as follows:

Mask	Does this	Example	Renames from	To
>tail	appends 'tail'	>_morning	importance f1	importance_morning f1_morning
<head	prepends 'head'	<Appearance_	importance f1	Appearance_importance Appearance_f1
old=new	replaces 'old' with 'new'	appearance=performance	appearance_importance	performance_importance

When appending or prepending a name mask to question names, the contents of the library items can be modified as well. Simply include {/} in the question text, Formula, Script or Help text, and it will be replaced with this value. Furthermore, use the syntax {questionname} to automatically rename any questions accordingly. For example, if "_earned" is appended, and the library item contains a question named "amount", which is referred to in a Script, write the Script so that instead of referring to just "amount", it refers to "{amount}". The Script will then contain a reference to amount_earned, so that question references do not need to be modified manually.

How to add additional question sets to the Question Library

Besides Q_Lib, it is possible to create additional libraries by creating new Places whose names begin with Q_Lib, such as Q_LibHR. All questionnaires from all question libraries are shown in the question library menu. The Q_Lib places must use the Standard style.

For users of ViewsFlash versions prior to 5, items from the Question_Library are also included. The name was changed to make it shorter, since the combined place name and questionnaire name cannot exceed 30 characters. Because Question_Library uses the deprecated Questionnaire style, its items are only visible in places that use the Questionnaire style. It may be desirable to copy questionnaires from Question_Library to Q_Lib to allow using them with the newer, more flexible Standard style.

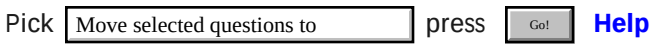
To add a new question set, create a questionnaire inside Q_Lib or one of the new places whose names begin with Q_Lib. Use the Title as the description of this set of questions; this title is what will be shown in the Question Library menus. It is a good idea to think ahead about the question names to be used, since the set of questions may be copied over and over into many questionnaires. For example, a matrix of questions could be named F_1 through F_5, so that they can be renamed using a mask like F=Favorite to produce question names Favorite_1 through Favorite_5.

The clipboard is a useful tool for copying questions from a test survey to the Question Library.

Special care must be used when storing Grids in the question library. See the **special instructions** provided.

Using the Clipboard

The clipboard is available only in the List Editor. Select one or more questions, pick "Copy selected to Clipboard", and press Go!



The selected questions are copied to a virtual "clipboard". When the clipboard contains questions, this menu contains "Paste clipboard at"; selecting this inserts the questions in the clipboard after the question selected, such as "the end", when Go is pressed.

When questions are inserted from the clipboard, if questions by those names already exist in the survey, the new questions are renamed automatically. The questions can be renamed by putting a **Rename Mask** in the name field.

Questions in the clipboard can be pasted into the same questionnaire where they came from or into any other questionnaire. The clipboard is cleared by logging out of ViewsFlash or by picking Empty clipboard and pressing Go!

The clipboard is also useful for copying data from a questionnaire and pasting it into a questionnaire in the question library.

Next: **Multiple page surveys**

Multiple page Questionnaires

Benefits

There are various reasons for breaking up a questionnaire into multiple sections, such as:
- to ask related questions as a group
- to skip around irrelevant questions (if you don't smoke, no point asking how often)
- to allow time to research the answers and to enter them later
- to provide additional instructional screens between survey sections
- the questionnaire is accessed through a single dynamic URL:
`http://yourcompany.com/ViewsFlash/servlet/viewsflash?cmd=page&pollid=pollingplace!poll`

How they work

To navigate between pages, multi-page questionnaire use up to four buttons:

Next	Goes to the next page of the questionnaire
Previous	Returns to the previous page
Save	Stops presenting questions and goes to page with a URL to use to resume taking the questionnaire later. When the visitor returns to this URL, they will resume where they left off.
Submit	Submits the entire survey, saves it and tallies it. After this, if you are checking for duplicate entries, visitors will not be able to fill out he questionnaire again. If there's no Submit button, the Next button will submit the survey when the visitor reaches the last page.

The Standard Form style shows and hides these buttons as appropriate. For example, the first page does not have a Previous button, and the last page has a Submit button but no Next button. The most common combination is to allow a Next and a Submit button and not the others. You should use this template or one derived from it. For details on these templates, see **Multiple page Tags**.

Because multiple page questionnaires require multiple form submissions, establishing the identity of the respondent is essential. Therefore, multiple page questionnaires **require** using **User Authentication**. The simplest method to use is Cookie, which is the default. If there is no need for a Save button, the Servlet Session method is recommended. If a single sign-on system is in place, use Basic or J2EE Authentication.

How to create a multi-page questionnaire, step by step

1. Create a place that uses the Standard and Results styles, or similar templates derived from them. In the Settings page, item 4, choose "Concurrently".

2. Create a questionnaire. In General settings, check the Multiple page questionnaire box. Choose what buttons to show by checking the boxes below; select at least Next and Submit.

3. In Settings / Security, choose an Authentication Method in #1. If nothing better is available, choose "Cookie" and leave its other fields blank.

4. Add questions as appropriate. Separate questions with Page breaks.

5. If you intend to use question branching, it is easier to add branching after entering all the questions. See **Branching** for complete instructions.

6. When finished, Publish the questionnaire and click on the URL below the Submit button. Try out the various options and modify any necessary settings until satisfied. Then remove the responses received so far by using the Delete All Data checkbox, and tell the world about the questionnaire by publishing its URL.

7. The Data Summary page includes a line, Questionnaires Presented, that counts how many questionnaires have been shown to respondents; the Questionnaires Completed line counts how many people have completed all pages of a questionnaire.

Using a question for Authentication

The "Use the following question" option in the Security page allows designating a question in the survey as the unique User ID. For example, create a question of type Hidden and call it Userid. The survey url should now be:
...?cmd=page&pollid=pollingplace!poll&Userid=TomJones
In other words, each recipient should receive a URL tailor-made for them, which includes their Userid.

If a participant saves the survey and resumes later, the URL must include the user id. The settings on #3 of the Security page affect this. The default response page, viewsflash/resume.html, does this automatically, using the [/ifqueryauthentication], [/authenticateduserquestion], [/authenticateduserid] and [/endifqueryauthentication] tags. If you use another template, be sure to start by copying this page. If you choose a redirect, you can use these same tags to pass appropriate information to the redirect URL, which will be responsible for reconstructing the URL at which to resume the survey.

Next: Pre-populating

Personalizing questionnaires

A questionnaire can include information that is unique to the respondent using the techniques described here. The information can be used in a myriad ways, such as to provide a personal greeting, to provide different choices, to control the path taken through a questionnaire. These techniques are illustrated in the Examples.

Using query string parameters

Every questionnaire has a unique URL, displayed in the Publish page. I has the form:
http://yourcompany.com/ViewsFlash/servlet/viewsflash?cmd=page&pollid=pollingplace!questionnaire

By adding extra parameters to the URL, the values of questions can be filled in. For example, if a questionnaire includes questions "name" and "title", the following URL will pre-populate them:
http://yourcompany.com/ViewsFlash/servlet/viewsflash?
cmd=page&pollid=pollingplace!questionnaire&name=Tom+Jones&title=Mr.

Then using the question piping technique, these vales can be displayed, usually in a question of type HTML:
Dear [/title] [/name] becomes Dear Mr. Tom Jones
Note that the + in the URL is replaced with a space by the browser when the URL is included in an email or web page.

This feature is used by **Invite Lists**, which can send personalized emails that include, for example,the participant's name in the URL.
This technique can also be used with any external systems that can create a dynamic URL.

Using database queries

Every questionnaire uses an Authentication method. The method chosen results in the questionnaire having a unique id, except when duplicate ids are explicitly allowed. Using question piping, it is possible to refer to the questionnaire's unique id as [/authenticateduserid]. This value can be used with the **SQLSelect** action to populate questions from a database query that uses the unique id as the lookup key in a SQL statement's WHERE clause. The values that are retrieved can be used in visible questions or in questions of type hidden. All those values are available throughout ViewsFlash as if they had been entered by the respondent.

Using session attributes

Using the Script action, it is possible to retrive HTTP session attributes and store them in questions. This is useful when ViewsFlash works in conjunction with a suite of applications, and those applications leave values in the session for a questionnaire to use. These values can even include the unique questionnaire ID used for authentication.

Next: Data Validation

Data Validation

Data validation is done in the browser, using JavaScript, as well as the server, using built-in validation and **Script**.

Every question can be required by checking the Response Required box in its question page. Some question types, such as text, allow validating that the input is in a certain range, such as 1 to 10000. Both browser and server side validation checks are performed.

For additional validation, use a Question Style such as E-mail, Number, and Date. These styles check for the appropriate content and prevent the respondent from continuing until a valid entry is entered. These styles only check input in the browser and do not validate input on the server.

You can **create your own Question Styles** to check the content of fields in your own ways. Check the Styles Library in the Support section of **www.cogix.com**.

Data validation messages can be customized as described in **Custom Validation Messages**.

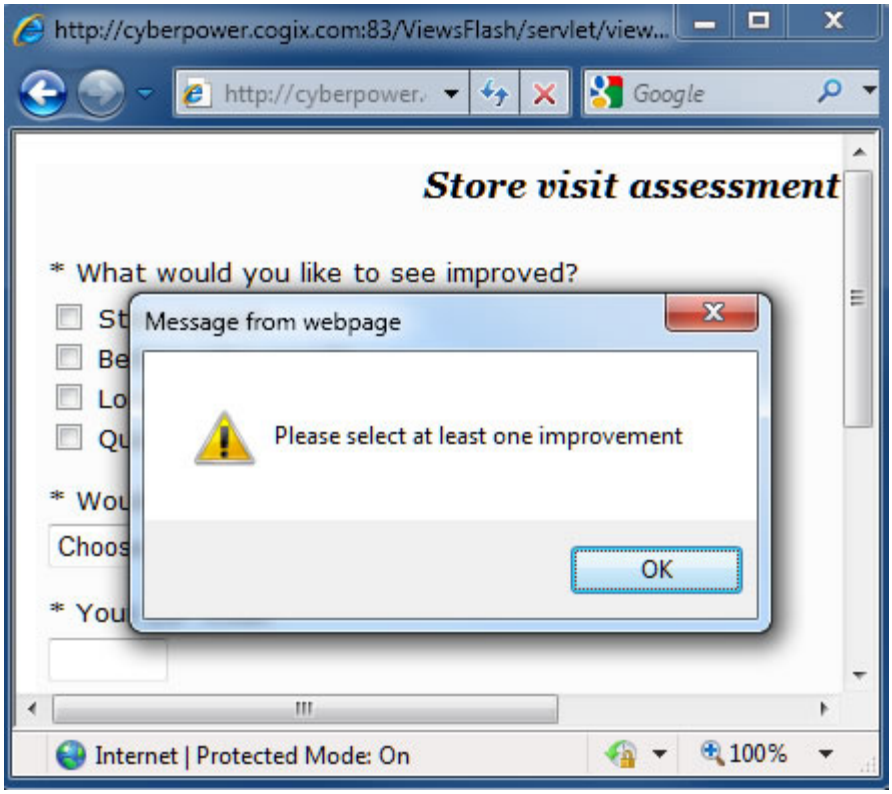
Next: Custom Validation Messages

Custom validation messages

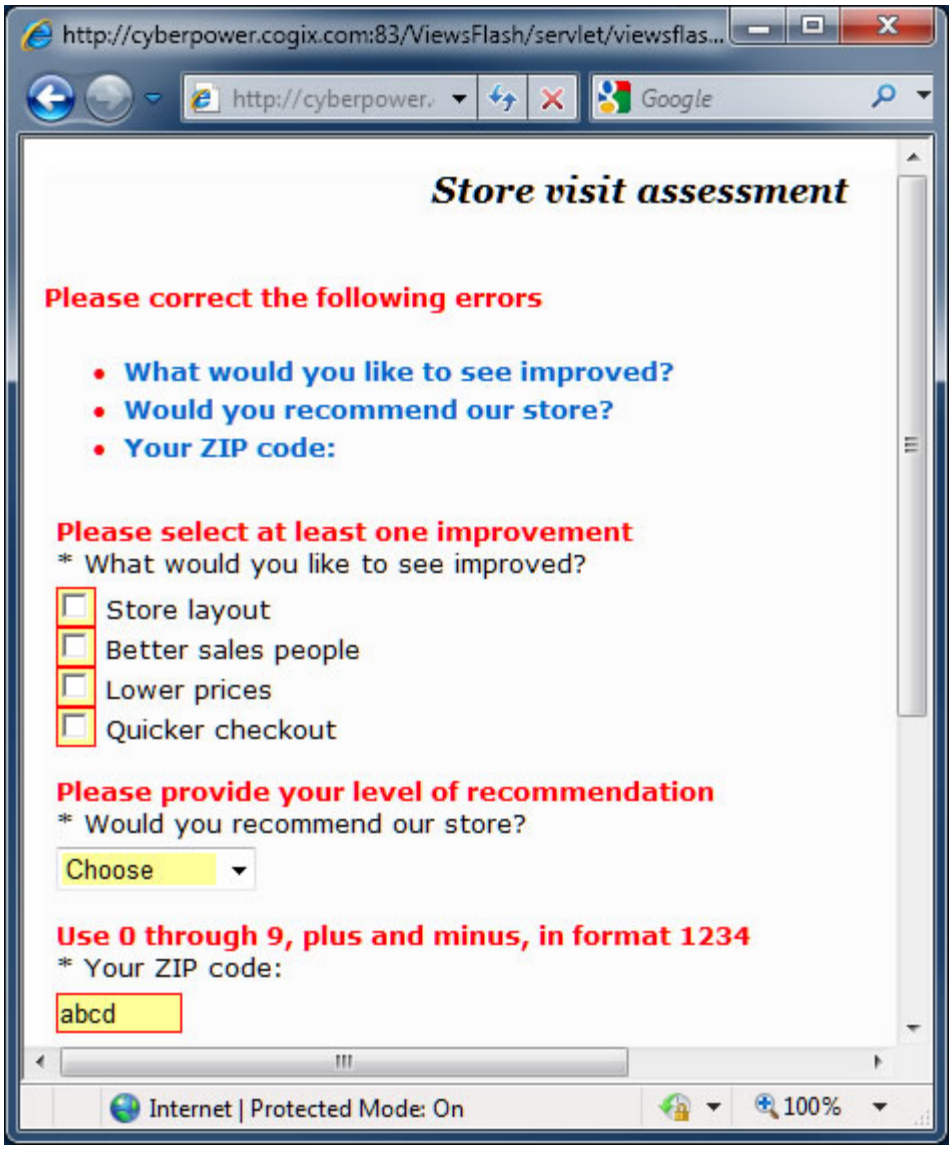
Overview

ViewsFlash provides rich facilities for checking questionnaire fields for correct values and for displaying appropriate error messages. The Style used when the question is created, such as Number, includes characteristics of what can be entered into the field. A question can also be flagged as Required. Each of these error conditions is reported using a standard error message. These messages can be redefined for all questions or tailored for specific questions. For example, one required question can use the message "Zip code required" and another required question can use the message "Please enter an amount", while they both share a message "Enter only numbers" when a non-numeric value is entered.

Validation error messages are presented when the page is sent by pressing the Next or Submit button on a questionnaire. By default, each message is displayed as a Javascript alert box, and the cursor is placed on the field in error when the alert is cleared, as shown here:



The General page includes a setting for [x] Show validation messages inline. With this setting, each message is displayed next to the question in error, and a listing of those questions is displayed at the top of the page, as shown here:



Specifying messages

To customize messages, enter in the Customized messages box one line for each message. Each line consists of a message code, an equal sign, and the message.

To change a default message, enter the message code and the message. Examples:
To change the default message "Please correct the following errors", enter:
correcterrors=Correct these errors
To change "Question must be answered", enter, on a different line:
presence=Answer this question
To change the text of the Next button:
Next=Continue

It is also possible to change the validation messages for a specific question. To adapt the "Question must be answered" message to questions named "name", "address" and "zip", enter:
presence-name=Fill in name
presence-address=Provide address
presence-zip=Enter ZIP code

The Show messages link displays a list of all the default messages. You can cut and paste lines from there.

When translating a questionnaire, Show messages displays messages in the language of the questionnaire, while the Show messages in English link displays the messages in English.
Cut and paste the entire text area to the Customized messages box and translate the messages there.

Next: Question Styles

Using Question styles

When a question is created, the first choice to make is which style it should use, using the "Select a question style" dropdown. As different choices are selected in the menu, a short description of each question style appears under it:

Select a question style:
Email
Check for valid email address
Explain

Note the **Explain** link under the menu. This link opens a popup window with complete details on how to use the Question Style selected. The explanations are more complete than those given here. Open the menu, scroll down to the style you're interested in, and click on Explain.

The Standard style is used for radio buttons, text fields, and other form elements. Other styles are used to:

Calculate	perform calculations with fields in the browser
DisplayOrHide	revealing and hiding questions in the browser
Javascript	insert Javascript for layout
ElasticText	create an automatically resizing Text Area
Calendar	create a pop-up date picker calendar
Upload	upload documents or attachments
Date	create a text field for a Date
Number	create a text field for a Number
Email	create a text field for an Email address
RegularExpression	create a text field with custom validation
Matrix	Likert matrix
MatrixSD	Semantic Differential scale
ReportText	insert text and HTML into univariate reports

Elastic Text is used with Text Area fields, such as Comments. The created form field is rendered small, and as the respondent enters text, it becomes as tall as necessary. This keeps questionnaires compact yet allows a large amount of text to be entered.

Calculate is used with one line text questions, often using the style Number. See **Calculation** for a complete explanation. Example:
average_income/$.2~ = (income_1 + income_2) / 2 ;

This will display as $1,234.00

The **Matrix** style allows creating a Likert scale question matrix like this:

Rank your satisfaction with each item

	Low		High
Service?	○	○	○
Price?	○	○	○

The **Matrix_SD** style allows creating a Semantic Differential scale question matrix like this:

Rate this product's packaging:

Cold	○	○	○	Warm
Serious	○	○	○	Fun
Hard to open	○	○	○	Easy to open

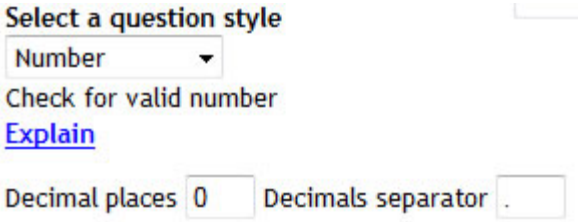
Both kinds of scales are easily created with the **Matrix wizard**. This wizard automatically creates questions that use these styles.

Date is used with a single-line text field. It checks that the date entered conforms to a specific date format, which by default is m/d/y.
To use a different date format, modify the content of the Date Format field:

Use the letters m, d, and y to represent month, day, year.
A single letter allows values starting with 1, two letters require values to start with 01.
yy allows values between 00 and 99. yyyy allows values between 1900 and 2099.
The parts of the date can be separated with a slash, dash, or period.

Examples:
m/d/y allows any length fields such as 12/31/08 or 12/31/2008
mm/dd/yy requires two digits for all fields such as 12/31/08
yyyy-mm-dd requires four digit years and two digits for others such as 2008-12-31
yy.mm.dd requires two digits for all fields such as 08.12.31

Number is used with a single-line text field. It checks that the number entered conforms to a specific format, which by default is an integer without decimal places or thousands separators.
Note that the number is stored in the database as a string, including all the separators. Negative numbers are always allowed.
To use a different number format, modify the content of the Decimal Places and Decimals separator fields:



If Decimal places is 0, the number must be an integer value with no decimals. 0 is the default.
If Decimal places is a positive number, the number must have that many decimals after the decimal separator.
If Decimal places is a negative number, the number may have up to that many decimals after the decimal separator.
The Decimals separator character can be set to a period or a comma. The default is a period.

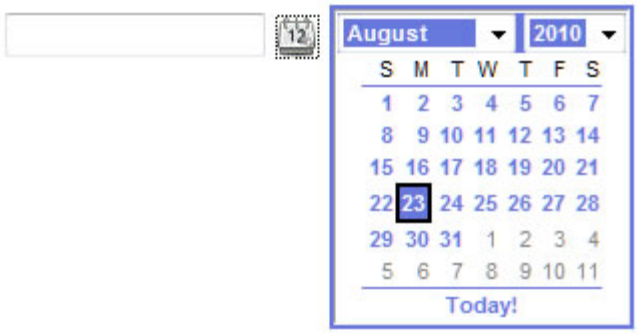
Examples, using a period as a decimal separator
2 allows 1.00 and rejects 1 1. 1.2 and 1.234
-3 allows 1, 1.2, 1.23, 1.234 and rejects 1. and 1.2345

Email is used with a single-line text field. It checks that the e-mail address entered has a valid e-mail syntax. It does not check that the address is a valid email-address.

RegularExpression is used with a single-line text field. It checks that the field contents conform to the RegExp pattern. For example, ^[2-9]\d{2}-\d{3}-\d{4} checks for a North America telephone number, in ordinary dash-separated format with no spaces, starting with a digit between 2 and 9.

Calendar is used with a single-line text field. It displays a calendar icon next to the field. When a user clicks on the calendar, it displays a visual date picker like this:

Target date:



3 Your e-mail

4 Your phone:

5 State:

Select

ReportText allows entering text and HTML. Questions that use this style are never displayed in respondent questionnaires, and they are not visible in the Visual Editor; they can only be created with the List Editor. These questions are included only in **Univariate Reports**, where they are used to add appropriate text at the given place in a report.

See also Creating question styles.

Next: Question Piping

Question Piping

Question Piping is the ability to use the answer to a question in subsequent questionnaire pages.

For example, let's say a question named "how-often" has been created, using one Extra field. The question asks "How often do you drink?" and four values have been defined, as follows:

Answer value	Answer text	Extra field 0
1	Never	not a factor
2	One or two glasses a day	healthy
3	Three to six glasses a day	to be watched
4	More	excessive

With question piping, the answer's value, its answer text, and its Extra fields, if any, can be used in a subsequent pages. To use them, insert the following tags anywhere that text can be used, as follows:

Tag	Shows as
[/how_often]	3
[/how_often,enteredanswertext]	Three to six glasses a day
[/how_often,enteredanswertext,0]	to be watched

So, a sentence like this can be used on any page after the question is asked, such as the next page, or the response page:
Your response to "How often do you drink?" had a value of [/how_often], "[/how_often,enteredanswertext]". Your drinking is [/how_often,enteredanswertext,0].
this becomes:
Your response to "How often do you drink?" had a value of 3, "Three to six glasses a day". Your drinking is to be watched.

These piping tags can be used in question text, including questions of type HTML. They can also be used in headers, footers, Actions, and pretty much anywhere - as long as the value has been entered. If not, the tags are replaced by empty space.

More detailed information about each tag:

[/question_name] - the value entered or chosen by the respondent. In an edit field, this is exactly what they entered. In a radio or checkbox field, it is the value that corresponds to this choice. In our example, [/howoften] is replaced with 1, 2, 3, or 4 or nothing if the question was not answered.

[/question_name,enteredanswertext] - the visible text that corresponds to their answer. In our above example, [/howoften,enteredanswertext] is replaced with "Never", "One or two glasses a day", "Three to six glasses a day", or "More".

[/question_name,enteredanswertext,N] - the text that corresponds to what has been entered in Extra field N. For more information on Extra fields, see **Design Question Page**. If we were using the extra0 field to contain a URL, then <a href=[/howoften,enteredanswertext,0]> would create an HTML link to that URL.

[/authenticateduserid] - the ID of the person taking the questionnaire, using the method specified in the Security page.

[/spotname] - the name of the place for this poll or questionnaire.

[/pollname] - the name of the poll or questionnaire within the place

[/pollid] - both names combined, separated by a '!', eg., DailyPolls!Monday

An excellent use of this capability is in **Personalized questionnaires**. In this example, an email includes a personalized URL like this:
servlet/viewsflash?cmd=page&pollid=pollingplace!poll&userid=12211&username=John+Peters

The first page of the questionnaire includes an HTML question with this text:
Thank you for participating in our questionnaire, [/username].
This would display as:
Thank you for participating in our questionnaire, John Peters.

When used with localized numbers and dates created with the LocalizedNumber and LocalizedDate styles, piping displays numbers and dates in a locale-indpendent format 1,234.56 and MM/dd/yyyy.
To display these numbers and dates in the format appropriate to the locale of the user's browser, use these tags instead: [/localized,question_name]

These elements can also be used in redirects. For example, on the Response page and the Security page redirects, you can redirect to
"*?q=r&t=[/howoften]"
which will redirect to "...?q=r&t=3 in the above example.

Next: Revealing Questions

Revealing and hiding questions

Sometimes it is useful to present one question to the respondent and, depending on how they answer, to present additional follow-up questions. If all these questions are simple and short enough to be presented on one page, using the DisplayOrHide question style is a simple and very effective to do so.
This style provides a way to display or hide groups of questions on the same page depending on the value of another question. Respondents can toggle between the groups, and values are removed from inappropriate questions.
See the Examples place for a demonstration.

For example, a single choice radio button question named "smoke" could have a value of yes, no, and maybe. A drop down question named "smoke_frequency" asks participants how often they smoke. A text area question named "comment" asks respondents to elaborate. When the respondent first sees the questionnaire, they should only see question "smoke" and the others should be hidden. If they click on yes or maybe, the drop down appears. If they click on no, the drop down disappears and the comments box appears instead.

To achieve this effect, first create the questions as described.
Then, following those questions, create a question with style DisplayOrHide, using the HTML question type.
In the "Formula" field, enter the following:

smoke
yes , maybe = smoke_frequency
no = comments

The formula is constructed as follows. The first line of the formula is the name of the **condition question**, whose value determines what questions to display and what questions to hide.
Each of the lines that follows includes an equal sign. To the left of the equal sign is a list of values, comma-separated. To the right of the equal sign is a list of affected questions, also comma-separated.
When the value of the condition question changes to a value listed to the left of the equal sign, the affected questions listed on the right side are displayed. When the value changes to another value on a different line, the previously shown questions are hidden and the new affected questions are shown.
Each condition is entered on a separate line. Multilpe conditions can be entered on the same line and separated by semicolons. Spaces can be used freely between items; smoke, smoke_frequency, and comments are question names.

So, in this example, when the page loads, neither smoke_frequency nor comments are displayed. Selecting yes or maybe displays smoke_frequency, and selecting no displays comments instead.

The list of questions to the right of the equal sign may contain just one question name, or a list of several question names separated by commas, like this:
yes = smoke_frequency, reason_for_smoking, attempts_to_quit_smoking
It can also contain a range of question names, so that all questions between the first and the last are included, like this:
yes = smoke_frequency - attempts_to_quit_smoking

Multiple conditions can be set up in the same pattern. The first line indicates the question name of the condition; subsequent lines indicate what questions to display or hide for different values of the condition.

The condition question can be of type: radio buttons, check boxes, dropdowns, text fields. So can the questions being displayed and hidden, including HTML.

Special values for radio button questions can be entered to the left of the equal sign: -other is used with radio button questions created with the "Other" option. -none is used with radio button questions created with the "None" option, or for when the user hasn't made a choice yet.

When used with a set of one or more check boxes, each value of the conditional question displays or hides the affected question independently of the other values.

The questions on the right side of the equal sign CAN be marked as "required" when created. In the smoking example, both smoke_frequency and comments can be required questions. The validation logic in ViewsFlash automatically disables validation for questions that are hidden by DisplayOrHide, and re-enables the validation when they are displayed again.

Required questions. Affected questions should not be marked as "required". To validate that a question which may or may not be displayed has been answered, use a Script action instead of "required".

To display or hide an entire matrix, refer to it as matrixname_ . A matrix's question names start with a matrix name; use that name with an underscore. Individual matrix rows can be displayed or hidden using the name of the question in that row, such as rate_service. It is not possible to display or hide just the header row or the explanatory text above the matrix.

To display or hide an entire grid, refer to it as gridname_. A grid's question names start with a grid name; use that name with an underscore. Individual grid rows cannot be displayed or hidden. However, individual questions in the grid can be, using the name of the question in that cell.

The set of lines that begins with the name of a condition question followed by a set of lines for each condition is called a condition block. Multilpe condition blocks can be included in one DisplayOrHide question. Blank lines can be inserted for readability.

Special parameters

The first line of a condition block usually includes just the name of the condition question, but it can also include special keywords that alter the behavior in specific ways. Use a colon to separate the name of the question from the keywords, and separate the keywords with commas.

noneshow When the page is first displayed, all questions listed in all the choices are hidden by default. This can be changed with this keyword, and all questions will be displayed instead.
smoke: noneshow;

remove or ***restore*** or ***preserve*** Use only one of these keywords, which control how to deal with selections and values entered into fields that are made visible but are then made invisible when the user changes the value of the condition question. In the smoking example used here, a user selects Yes and then enters a value into the smoke_frequency field; then he changes his mind and selects No. The ***remove*** keyword, which is the default, will remove the value from smoke_frequency immediately; if the user selects Yes again, he will have to re-enter the value for smoke_frequency.
The ***restore*** keyword changes this behavior so that he will not have to re-enter the value and the value will still be there when he changes his mind. The preserve keyword changes the behavior just like restore, with a difference: when using preserve, the value of the choices made in the fields that are not displayed will be sent to ViewsFlash with the questionnaire and therefore be saved in its data, whereas neither remove or restore will send any data to ViewsFlash.

recursive It is possible to have a hierarchical set of DisplayAndHide questions. For example, an initial question about age displays different sets of questions for younger and older people. Within the younger group, a question about the value of education triggers is used as a conditional questions for more detailed questions about school plans, and a similar pattern is used for adults and retirement options. When this parameter is used, the DisplayOrHide questions themselves can be specified in the initial question's choices, and the "nested" DisplayOrHide conditions will be evaluated and carried out. The net effect is that the user will always see only those choices that are meaningful.

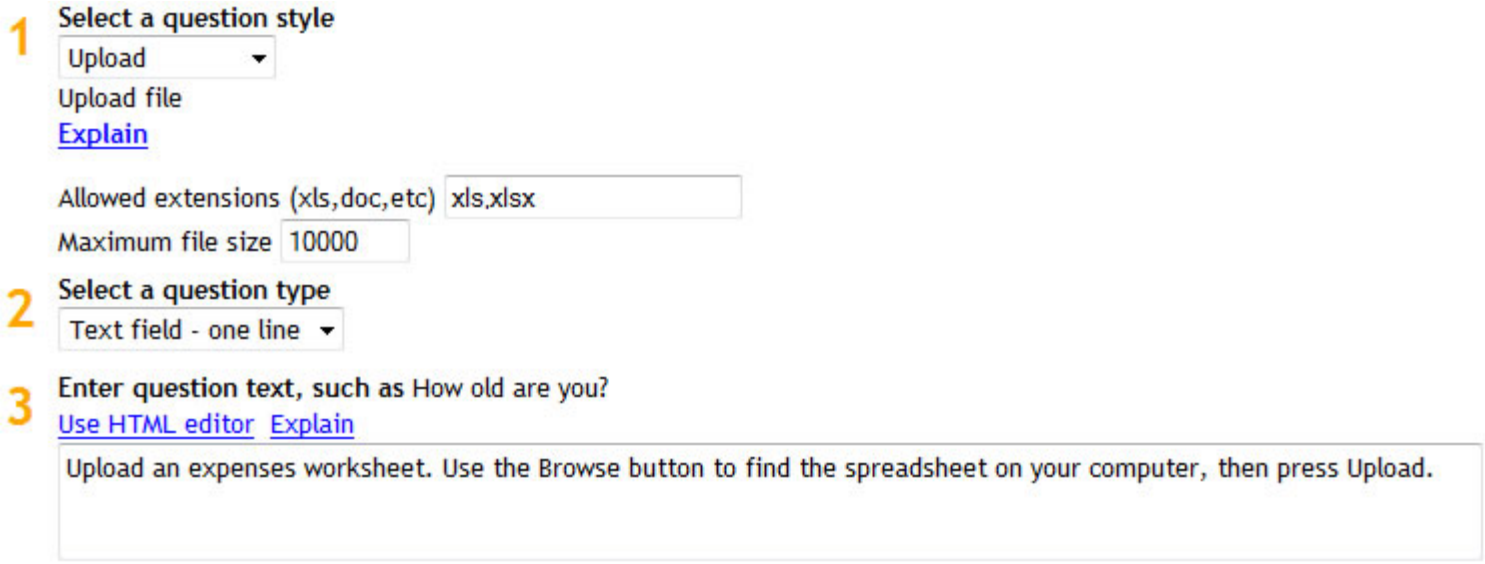
Next: Attachments

Attaching documents to a questionnaire

Documents, images, media, and any file types can be attached to a questionnaire by uploading them from the respondent's computer, and can be retrieved using a Data report.

Creating a question to upload the attachment

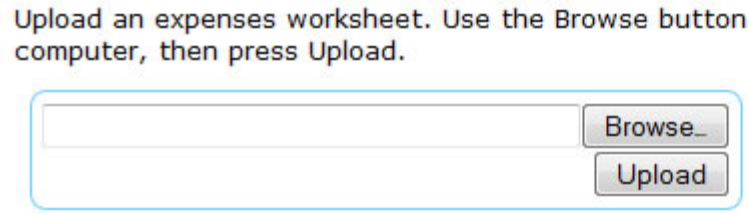
To allow a questionnaire to upload a document, create a question and choose the Upload question style:



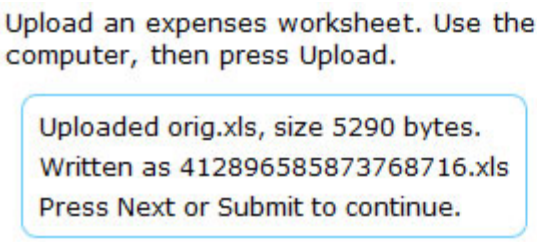
In the allowed extensions field, enter a list of the document extensions allowed, such as xls,xlsx,doc,docx separated by commas.
In the maximum file size field, enter the largest document size allowed.
Both of these parameters are important to prevent upload denial of service attacks.

Respondent experience

These parameters will create a question that looks like this:



The respondent uses the Browse button to find the worksheet, and then presses Upload. The document is uploaded, and the respondent sees a confirmation message:



The respondent can then can proceed to the next page of the questionnaire.

Retrieving the data

When a document is uploaded, a unique identifying number is assigned to it. The original file name and the unique number are stored in the Upload question's field in the database. When the questionnaire data is retrieved using a Data report, responses that have an attached document include a clickable link with the original file name:

Date	employee	report
7/6/08 5:06 PM	Johnson	
7/6/08 5:06 PM	Peters	EuropeExpenses.xls

4 responses

Clicking on the Attachment link will cause the document to be downloaded from the server and loaded with the application that is registered for that document's MIME type, such as Excel or Word.

Technical information

As of release 7, ViewsFlash stores uploaded documents in the database by default, and no additional servlet parameters are needed.
To override this and to store documents in the file system as previous releases did, the system Administrator must include upload.method=file in the ViewsFlash servlet parameters. If a database is not being used, documents are also stored in the file system.

When upgrading from previous releases, including the upload.method=file servlet parameter will preserve the legacy behavior. If the parameter is omitted, all the documents saved in the file system will be migrated automatically to the database the first time that the application starts; note that the application might take a relatively long time to start.

The servlet parameter "upload.maxsize" limits the size of the largest document that can be uploaded. The default value is 1 million bytes.

If not using a database, the system Administrator must include the servlet parameter "upload.directory". If it is not specified,it defaults to /etc/cogix/upload. The upload.directory parameter must point to a directory where the ViewsFlash application server has been granted file creation, directory creation, and read, write and delete access rights. In a single-server deployment, any directory can be used. In a clustered deployment, this directory must reside on a shared volume, so that attachments gathered from all servers are stored in the same directory and can be accessed by all nodes in the cluster.

Uploaded files are stored in subdirectories named SSS!PPP, where SSS is the Place name and PPP is the questionnaire name. The files are saved using names RRRRR.ext, where RRR is a random number and .ext is the original file extension.
The generated file name and the original file name are written to the Upload question's field in the database.

Next: Branching

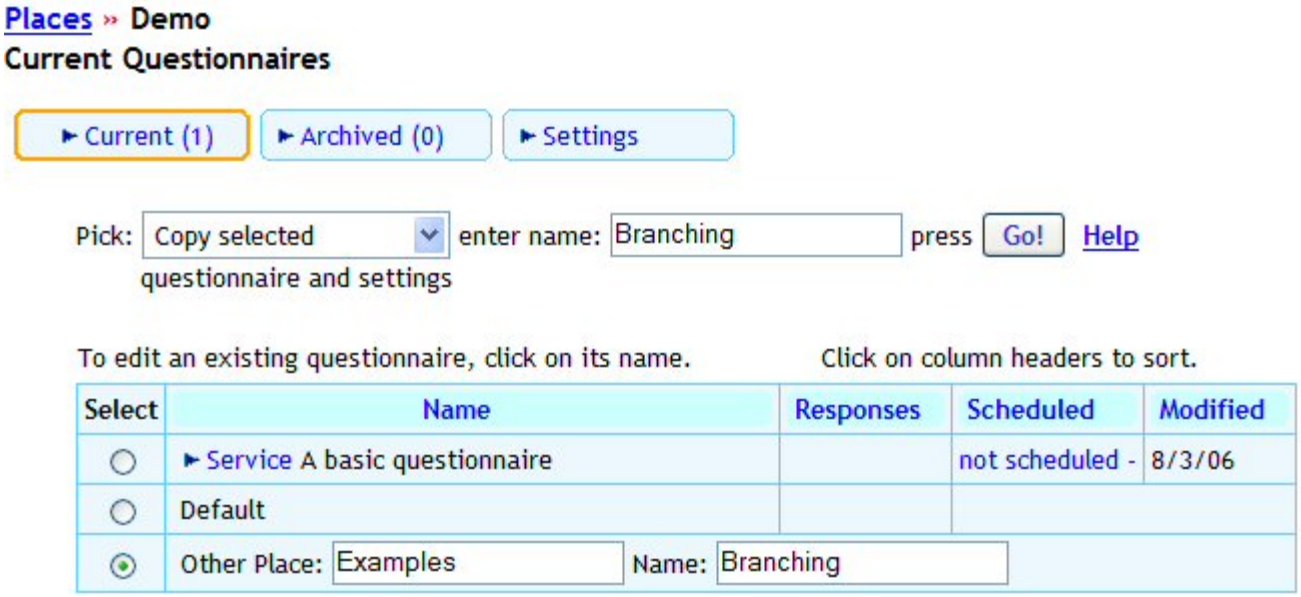
Branching

In addition to Revealing questions on the same page, there are two ways to change the flow of questions shown to a participant from page to page.

First, the Define Question page provides the ability to say, for example, "If a visitor answers Yes, go to question "Art":

If visitor chooses Yes
skip to question Art

This suffices for simple yes/no questions. For more complex branching, use the Script Action. To see them in action, create a Place and Copy the Branching questionnaire from Examples:



The Branching example illustrates setting up a questionnaire with a radio button question, a series of pages, a checkbox question, another series of pages, and the appropriate Action scripts on each page that skip over the page if it is not appropriate depending on how the questions have been answered. See the Examples place.

Each page includes a question of type Action at the beginning, specifying a Script when=before action. This action executes the script before the page is displayed. The script itself is, for example:

if (sex != 1) nextpage();

This means that if question"sex" was not answered with the choice whose value is 1, the respondent will not see this page. On the next page, if there is a similar action, the process is repeated until a condition is met, and then the page will be shown to the respondent.
Please study the example and its comments.

Next: Calculation

Calculation

The Calculate question style performs calculations among questions displayed in the browser, and stores the result in a destination question. It allows integer, decimal and currency formatting, and setting the destination question to read-only status.

To use this style, first create the questions that will be involved in the calculation. Question types text-single line, radio button, dropdown menu, and hidden can be used. Checkboxes and textareas cannot be used. Create a destination question, which must be of type text-single line or hidden.
Then create a question with style Calculate, using the HTML question type.
In the "Formula" field, enter the formulas to be evaluated. Example:

```
total = ( subtotal - discount ) * 1.08 + shipping ;
```

total, subtotal, discount, and shipping are question names.
Constant values can be used, such as 1.08.

Arithmetic operations supported are + - * / ( ), and all JavaScript Math functions such as Math.min(a,b), Math.max(a,b)and Math.round(x).

Logical operators can also be used:
size = a > b; will set size to 1 if the value of question a is greater than the value of question b, or to 0 otherwise.
rate = a > 10 ? 3 : 4; will set rate to 3 when the value of a is greater than 10, or 4 otherwise (reference: the ? and : JavaScript Operators).
Spaces can be used freely. Parenthesis can be used as expected. Multiple formulas can be entered; end each formula with a semicolon.

The destination question is recalculated as soon the value of any question used in the calculation is changed and the cursor leaves that field or a radio button is clicked or a drop down changes in value. A destination question can be used in question piping on subsequent pages. A DisplayOrHide question can also use a destination question for its condition question.

Fields without values are treated as having the value zero. If a field has illegal values the calculated field will show NaN.

Formatting
By default, calculated fields are disabled and display as an integer: 1234
This can be controlled by appending a slash and a format specification to the calculated question name, like this:.

```
average_income/$.2 = ( income_1 + income_2 ) / 2 ;
```

This format specification will display as $1,234.00
If the number is negative, a - is displayed in front of it: -$1,234.00
The characters in the format string must be entered in the following order. All of them are optional:

- $ will display a leading currency symbol.
- . (a period) will separate the decimal digits, if any, with a period, and group thousands wth a comma.
- A single digit number such as 2 indicates the number of decimal places to show; if this number is omitted, or zero, the decimal point is omitted.
- The calculated field will be disabled, so that the user cannot modify its value. To allow modifying the field, use a trailing ~ like this: /$.2~

The leading character(s) are interpreted as the currency symbol,
such as : $, €, £ ¥ 円

The decimal separator can be set to a comma, and the thousands separator will be a period, as in numbers formatted for European usage.
If neither comma or period is specified, the calculated field will not have decimal or thousands separators.
Examples:
average_income/€,2 will display as €1.234,00
average_income/DKK,2 will display as DKK1.234,00

A default format can be set by including this expression before the formulas. The following will display all calculated fields as $1,234.00 :

```
calculate-default=$2.;
```

Note that these calculations are performed in the respondent's browser, and can only use the values of questions on other fields in the same page of the questionnaire. For other calculations, see the next section on **Scoring**.

Next: Scoring

Scoring

A score can be created by using Script action, which cacluates the score and stores it in a hidden question. The Script action may use when=before on the page where all the components of a calculation become available, or when=after on a page where the result may be used or stored. Here's an example of the Script:
health_risk_score = drink + smoke + weight;
There are four questions: drink, smoke and weight have numerical values; health_risk_score is a hidden question.

The hidden question that contains the calculated score can be used like any other question. It can be piped into another, it can be saved, it can be tallied, and it can be analyzed. This section outlines a few additional pointers for how to get best results out of them.

The calculated scores can be integers, decimals, or characters. Most of the time they will be integers. The Script custom action allows specifying how many decimal places will be calculated. If you don't specify any, a calculated score is rounded to the nearest integer. Otherwise, the indicated number of decimal places are stored with the score.

Refer to the **Script** section for complete information on all its options.

To tabulate individual score values, it is necessary to go to the Options / Tally page and turn on tallying for the hidden question. If the score includes decimal places, consider increasing the default limit of 250 different possible answers. Note that more than 250 values can be quite expensive on CPU resources; if you don't need to analyze these fields, then leave tallying off.

When analyzing data with Univariate and Cross tabulation Analysis, a new option allows tabulating Others, both as a group and individually. Which one to use depends on the statistical measurements you want to take. The following table summarizes the recommended settings:

If score is...	Turn Tallying	Analyze Others	Statistics allowed
Integer values	On	Separately	Counts, percentages, mean, standard deviation
Integer values	Off	As One	Mean and standard deviation only
Decimal Values	Off	As One	Mean and standard deviation only

Next: Actions and Script

Actions

Actions are special functions that allow calculation, branching and specialized processing. You can create your own, since they are **Extensible.** The most commonly used Action is Script, described here. See also **Other actions**.

To use an Action, create a question with the Standard style and choose the Action type. In the Action script field, enter the name of the Action, a space, and any parameters it uses. Many Actions require additional information to be entered in the Script details field, which automatically increases in size if necessary.

The Script Action

Using Script, a questionnaire form can score and calculate, validate data, go to or skip around questionnaire pages, hide questions, and submit or end the questionnaire.

Many of these functions are illustrated in the Question Library and in the Examples place. To use the Question Libary, simply pick it from the menu and select one of the questions presented in the drop down menu.

Security Note: If using **User Security**, the administrator must grant the the right to use Script in the questionnaire's Location.

Example

To use a Script, create a question of type Action, and enter in the Action script field:

```
Script when=after
```

Enter the script in the Script details field. For example:

1 **Select a question style**

Standard

Explain

2 **Select a question type**

Action

Action, such as Script

Script when=before

3 **Script details**

risk=0; if (drink >=1) { risk = risk + 1; }

In the example below, two questions are created: "drinking" as a radio button with values 0,1,2,3, and "risk" as a hidden field on a second page. The value of risk is 0 or 1, depending on how a visitor answers the drinking question. An Action question, calculate_risk, on the first page, calculates the value of risk. The Visual Editor shows this:

The question "risk" is defined as a hidden question, and tallying must be turned on to analyze it, as described in **Scoring**.

Additional Script parameters

The Action field must contain the word Script. Additional parameters are of the form name=value. If more than one parameter is used, separate it from the others with a comma. The following additional parameters can be used:

when=before
If present, means perform the script before the page is shown to the participant.

when=after
If present, means perform the script after the participant presses the next or submit button.

If neither is present, and the Action is the first question on a page, when=before will be assumed; similarly, if the Action is the last question on a page, when=after will be assumed. If the question is neither the first nor the last, one of these parameters must be present.

decimals=N
If present, means calculate values with N decimal digits (eg., decimals=2). If absent, values are truncated to the nearest integer. Example: decimals=2

invalid=NNN
If present, when data used in calculations contains invalid characters or numbers, this value will be substituted instead. Example, invalid=0.

separators=yes
If present, calculated values will show grouping separators such as 1,000,000. For example, the value 1,234.56 will be shown as follows:
N=0 1,234
N=2 1,234.56

locale=LL
If present, means format numbers using this locale. If not present, the platform's locale is used. For example, locale=en_US will format numbers as 1,234.56, and locale=de_DE will format numbers as 1.234,56.

type=xxxxxx

If present, this parameter tells Script how to treat the fields in a calculation. Normally, this parameter is absent, and Script considers all fields as numeric, except when the question name includes the word "date", or the actual data is not numeric. When using type=, Script treats all questions involved as the type indicated. This parameter is best used with very short scripts. The possible values are:
type=float, type=date, type=string and type=integer.

reviewing=1

If present, this parameter suppresses execution of the Script during normal questionnaire processing. The Script will be executed (before or after) **only** when the questionnaire is reviewed or updated using Data Review API commands.

error=XXXXX

If present, this parameter determines what to do when an expression evaluation fails. This is most often caused by a syntax error, by misspelling the name of a function or variable. Although many of these potential errors are caught before a questionnaire is published, it is possible for these errors to remain undiscovered until actual data is entered. If that is the case, using this parameter can be very helpful.
The default value for this parameter is 123E.
The error parameter is either NNNI and NNNE, where NNN is a number. NNNI will log the error to the Poll log and the system log. NNNE will also send an email to the ViewsFlash administrator. We recommend using E, as this will get someone's attention more quickly.
This parameter may contain the letter C or Q in front of it. If both letters are missing, and there's something wrong, the Action will be considered Invalid, questionnaire processing will stop, and the action specified in the Invalid Data section of the Security page will be performed. If this parameter begins with a C, as in C123E, processing will continue and the error message will be shown on the next page of the questionnaire at the top. This setting can be useful during early testing. If this parameter beings with a Q, as in Q123E, the error will be logged and processing will continue.
The default value of this parameter is 123E, which stops processing, writes the cause to the log, and notifies the ViewsFlash administrator of the cause.

Script syntax

Scripts use standard Java syntax:

- each statement must end in a semicolon
- curly brackets { } are used to group statements
- /* */ and // are used for comments
- all standard Java operators and functions can be used, including ? and : (if-then-else)
- all Java conditionals can be used, including if () { } else { } , switch { } , for (, ,), while ( ) { }, do { } while () and break
- all Java library classes can be used, **provided that** they are used with their full package names, such as java.util.Date.

Script semantics

The "variables" in the scripts are question names from the questionnaire. Unlike Java, variables must NOT be declared. If you declare a variable explicitly, it becomes a temporary variable instead of a question in the questionnaire:

```
int intermediate_calculation=0;
```

When using question names in expressions, do not put them in quotes. When using question names as parameters to built-in functions such as gotoquestion, put them in quotes. The examples below show explicitly when to use quotes and when not to use them.

Branching

To direct flow to a page **based on the value of a question**, such as a radio button, and to skip around the other pages, use a Script Action with when=before on each of the pages, and use an if statement to not avoid showing the page if it is not relevant. For example, if a question "income_level" has values 1,2,3, add a script like this to each page that contains income_level:

```
if ( income_level != 1 ) {
  nextpage(); // skips to the next page
}
```

On the next page:

```
if ( income_level != 2 ) {
  nextpage();
}
```

And on the next page:

```
if ( income_level != 3 ) {
  nextpage();
}
```

You can also go to a question on any following page by using:

```
if ( income_level != 3 ) {
  gotoquestion ("nextsection");
}
```

To present different pages **based on the multiple selections** made by the user in a in a multiple selection (checkbox) question, use a Script Action with when=before on each of the pages, and use an if statement with the multisin function. For example, if a question "favorite_colors" has values "red","green" and "blue", add a script like this to each page:

```
if ( !  multisin ( favorite_colors,"red")  ) {
    nextpage();
}
```

This tests favorite_colors for the value red; the ! means "not", so if the color red is not selected, this page will not be shown. Add similar code on each page to prevent the page from being shown when it is not appropriate.

The Branching questionnaire in the Examples place illustrates these techniques in action.

Dates

Script can handle date arithmetic. Make sure that question names include the word "date", as in "event_date", "date_last_entered". Dates are stored in the database as strings, in US English format MM/DD/YYYY, such as 06/28/2005.

To calculate the difference between two dates, use the datedifference function, which counts the number of years, months,

or days between two dates. For example:

datedifference (date_created,"01/01/2000","days")
gives the number of days between the turn of the century and the date stored in date_created.

datedifference (today(), last_date_modified,"months")
gives the number of months between today and the date stored in last_date_modified.

datedifference (last_intervention_date,birthdate,"years")
gives a person's age at the time of last_intervention_date.
Note that the first date must be the most recent date; these functions do not return negative values. The third parameter must be a string in double quotes; only the first character is examined, so "months" is the same as "m". The today() function can be used to dynamically compute the date when the form is submitted. To compare against a static date, use the form "MM/DD/YYYY", in quotes, as in "06/28/2005".

To create a date relative to another date, use dateadjust, which calculates a new date a given number of days, months, or years away from the given date. For example:

dateadjust (today(),7,"days")
gives the date a week from today

dateadjust (today(),-1,"year")
gives the date one year ago

dateadjust (first_date,3,"months")
gives a date 3 months after the date stored in first_date
Note that the third parameter must be a string in double quotes. If the second parameter refers to a question, turn it into an integer with:
dateadjust (first_date, (int) number_of_months, "months");

Multiple selection utility functions

These functions are used with multiple selection (checkbox) questions. For example:

multisin (favorite_colors,"red","yellow","green"))
will be true if the user checked any of these choices, and false if the user doesn't check any of them.

multcount (favorite_colors)
will count how many of the choices have been selected (checked) by the user.

Utility functions

Use showmessage to display a message to the user on the next page shown. For example,

showmessage ("The score is " + score.toString());
will show a message like "The score is 3.0".
To get rid of the decimals, use this:
showmessage ("The score is " + ((int)score).toString());

Use WriteLog to write a message to the questionnaire's log, visible in the questionnaire's Administration menu under View Log. For example,
WriteLog ("123I","My Message");
will write My Message to the log with a message number 123I. Message numbers with an E, such as 123E, will also be written to the Administrator's log and will send an emergency email message.
Question values can be used, such as "The value of x is " + x;

Use isin to test if a question has been answered in a particular way. For example,
isin (score,4,5,8,9)
will be true if score is one of those values, and false otherwise

isin (flavor, "vanilla","chocolate","strawberry")
will be true if flavor is one of those. Note that strings are in quotes.

Use answered to test if a question has been answered:
if (answered (age))
will return false if the question was not answered.

Use numberanswered to count how many questions were answered. This will return a value between 0 and 3:
numberanswered (height, width, weight)

Flow control functions

These functions provide branching and other operations that alter the flow through the questionnaire form.

gotoquestion ("questionname");
Displays the page with question "questionname". If you don't use quotes, assign a value to the variable.

hidequestion ("questionname");
The question named "questionname" will not be displayed to the user completing the questionnaire form, whether on this page or a subsequent page. If you don't use quotes, assign a value to the variable. "Wildcard" names can be used to hide questions with similar names, such as "grid_*_1" or gridA_2?A?_1_*"

unhidequestion ("questionname");
The question named "questionname" will be displayed again. This reverses the effect of hidequestion. Wildcards can also be used.

showquestion ("questionname", show);
This function combines hidequestion and unhidequestion, above. When the second parameter is set to true, it acts like unhidequestion, and when set to false, it acts like hidequestion. Wildcards can also be used. Example:
showquestion ("price",true);
Instead of the value true or false, and expression can be used as well:
showquestion ("tax", state.equals("CA"));

invalid ("questionname","message");
Displays "message" next to the question "questionname". This does not stop Scipt processing. Use a "return" statement to stop processing.
When questionname is an empty "" , the message is displayed at the top of the page.
When questionname is a series of question names separated by spaces, the message is displayed next to each question.

submit ();
Post the entire questionnaire as a completed form immediately, without displaying any further questions.

redirect ("http://redirectto");
Equivalent to the user navigating to a URL without submitting the questionnaire form. All data on that page of the questionnaire is discarded. Cannot be used in portals.

reset ("http://redirectto");
Discard any answers received so far and redirect to the given URL. Cannot be used in portals.

disallow ("http://www.yahoo.com"); // redirects to yahoo; always use a full URL, including the protocol
Discard any answers received so far, prevent the user from entering data in this form again, and redirect to the given URL. Cannot be used in portals.

nextpage() and nextpage("http://www.yahoo.com")
When updating a questionnaire in place (rather than adding a new one), will go on to the next page, unless this is the last page of the questionnaire, in which case the questionnaire will be submitted and then this will redirect the browser. When an agument is present, the browser is redirected to that URL; the argument cannot be used in portals.

save ("http://www.yahoo.com");
Saves the data entered so far, as if the Save key was pressed, and redirects the browser. Cannot be used in portals.

submit("http://www.yahoo.com");
Post the entire questionnaire as a completed form immediately, without displaying any further questions, and redirects the browser. Cannot be used in portals.

submit ()
Post the entire questionnaire as a completed form immediately, and shows the questionnaire response page. Cannot be used in portals.

Advanced uses

Methods are provided for direct access to the HttpRequest, HttpResponse and HttpSession objects. Be careful when using these methods in a JSR 168 portal environment, as they are likely to not work in a portal. To redirect, use methods such as reset and submit instead. The methods are, with examples:

```
getRequest().getRemoteUser();
getResponse().setContentType ("text/html");
getSession().getAttribute("attributename");
```

When using custom Java classes, if a class name does not being with a capital letter, it may be necessary to provide a fully qualified name of a class. Examples:

```
String x = new String ("abc"); // is ok
com.myco.myclass x = new com.myco.myclass (); // fully qualified name required
CustomClass zz = new CustomClass (0); // is ok
```

Java environment variables can be read using:
System.getProperty("propertyname")
They can be assigned to hidden fields, which can then be used with question piping anywhere it is allowed.

Some of the Cogix provided examples may use a variable named cogix_vwf_validator_instance. This is used to acess advanced internal capabilities and should only be used as shown in Cogix examples. it is recommended to use getRequest() or the built-in method such as reset and submit described above instead.

Next: Other Actions

Other Actions

The Action question type allows special processing. Some of these functions are built-in, and other functions can be custom programmed (see **Extending Actions**).

To use these, you simply enter in the Custom Action field the name of the custom action, a space, and its parameters, as described in the following table. For more information on each action, click on its name.

Custom Action and parameters	What it does
DynamicDropdown	Creates a drop down menu whose contents reflect answers to previous questions or the result of a database query.
CascadingMenus	Creates two or more drop down menus where the contents of each menu depend on the selection made on the previous menu. The data comes from a database query.
SQLSelect when=before or when=after	Performs a SQL query statement and fills questions from fields in a database table. The SQL statement is entered in the Scipt details field and may use question piping.
SQLUpdate when=before or when=after	Performs a SQL update statement and inserts or modifies data in fields in a database table. The SQL statement is entered in the Script details field and may use question piping.
SQLDisplay when=before or when=after	Retrieves multiple rows of data from a database table using a SQL query and displays the data in fully customizable tabular form.
RandomizeQuestions q1,q2	Displays a range of questions on a page in random order.
IssueHttpQuery url	Sends a URL query, which can include data entered by the respondent, to an application (which can be ViewsFlash or any other application).

Next: DynamicDropDown

Custom Actions: DynamicDropDown

This custom action allows creating a drop down menu whose contents reflect answers to previous questions or the result of a database query.

Create a drop down question no values, or with some values that will always be included in every dropdown, and note its name.
Then create a question of type Action, and fill it out as indicated here.

The Action question MUST follow the dropdown quesion,not precede it.

To fill a drop down menu using a SQL query

In the Action field, enter: DynamicDropDown dropdown=name of the dropdown question to fill out
The Enter Script field contains a SQL statement, which can use **question piping**. The SQL statement selects two fields, one for the values to be stored in the database and one for the visible contents of each option in the drop down menu. If only one of those is available, select the column twice in the SQL query; note that some databases require using an AS clause after the repeated column. Add datasource=JNDIDataSourcename right after DynamicDropDown to execute the SQL query using a different data source. Normally, DynamicDropDown finds tables in the same database that all of ViewsFlash uses. But sometimes, it is useful to find the data in a different schema or database altogether. In this case, create an additional JNDI Data Source and put its name in the datasource parameter. The additional database does NOT need to be the same database brand (eg., Oracle / DB2). More detail **here**. Note that [/authenticateduserid] is the tag, described in question piping, that contains the user id of the current logged-in respondent. Note that [/dbtablenameprefix] will use that paramer if set up in the servlet parameters to provide a prefix for the table name.

Examples:

1 To fill in a dropdown menu question named PickCountry from a table of two letter country codes and country names:
In the Action field, enter
`DynamicDropDown dropdown=PickCountry`

In the Script field, enter:
`select country_code, country_name from country_table`
The table name is country_table, and the two fields in the table are are country_code and country_name.

2 To fill the same menu with only those countries in the continent selected by the respondent when answering an earlier question, use question piping:
In the Action field, enter
`DynamicDropDown dropdown=PickCountry`

In the Script field, enter:
`select country_code, country_name from country_table where country_continent=[/continent]`
The table name is country_table, and the fields in the table are are country_code, country_name and country_continent.
The value entered in response to an earlier question, named "continent", is used in the where caluse to restrict the dropdown to only those rows where country_continent equals continent.
Note that [/authenticateduserid] is the tag, described in question piping, that contains the user id of the current logged-in respondent.
Note that [/dbtablenameprefix] will use that paramer if set up in the servlet parameters to provide a prefix for the table name.

To fill a drop down from values entered or selected in earlier questions

In the Action field, enter: DynamicDropDown dropdown=name of the dropdown question to fill out,list of question names separated by commas

Examples:

1 A respondent is asked to enter the most important life goals using five text questions named goal1, goal2, goal3, goal4, goal5.
On the next page, a dropdown menu question named Goal is filled out from the responses to those five questions.

In the Action field, enter
DynamicDropDown dropdown=Goal,goal1,goal2,goal3,goal4,goal5

2 A respondent is asked to enter the most important academic goals using three single-choice radio button questions named acad_goal1, acad_goal2 and acad_goal3
On the next page, a dropdown menu question named Acad_Goal is filled out from the responses to those three questions.

In the Action field, enter
DynamicDropDown dropdown=Acad_Goal,acad_goal1,acad_goal2,acad_goal3

Note how this example is identical to the first, even though the question types used are very different.

3 A respondent is asked, using a multiple-choice checkbox question named languages_spoken, to indicate all the languages spoken at home.
On the next page, a dropdown menu question named primary_language is filled out from only those choices selected by the respondent.

In the Action field, enter
DynamicDropDown dropdown=primary_language,languages_spoken

Security Note: If using **User Security**, the administrator must grant the questionnaire designer the right to use this custom action.

This Action is always executed before the page that contains the Action is composed.

It is also possible to create multiple-level drop down menus, such as car year, brand, model. See **Cascading Menus**.

Next: CascadingMenus

Custom Actions: CascadingMenus

This custom action is used for creating a set of two or more drop down menus where the contents of each drop down depend on the selection made on the previous drop down. The typical example of these menus is car manufacturer, year, and model dropdowns. The values in the dropdowns come from a database query.

Create a set of drop down question with no values and note their names. Then create a question of type Action, and fill it out as indicated here.

The Action question MUST follow the dropdown quesions,not precede it.

In the Action field, enter:
CascadingMenus dropdown1,dropdown2,dropdown3 where each of these is the name of one of the empty dropdown questions. There is no limit to how many dropdowns can be used. For a single dynamic dropdown, use **DynamicDropDown** instead.

The Enter Script field contains a SQL statement.

The SQL statement selects from a table **exactly twice** as many columns as there are dropdown menus. Each pair of table columns is used to fill the corresponding menu. The first column of each pair is used to fill in the value stored by a menu option and the second column is used for the visible text of that option. If the value and the text are the same, repeat the same column name; note that some databases require using an AS clause after the repeated column. A where clause can be used, and **question piping** can be used anywhere in the SQL statement.

Add datasource=JNDIDataSourcename right after CascadingMenus to execute the SQL query using a different data source. Normally, CascadingMenus finds tables in the same database that all of ViewsFlash uses. But sometimes, it is useful to find the data in a different schema or database altogether. In this case, create an additional JNDI Data Source and put its name in the datasource parameter. The additional database does NOT need to be the same database brand (eg., Oracle / DB2). More detail **here**.

Note that [/authenticateduserid] is the tag, described in question piping, that contains the user id of the current logged-in respondent.
Note that [/dbtablenameprefix] will use that paramer if set up in the servlet parameters to provide a prefix for the table name.

Example. A database table CARS has been created with columns MAKE, MAKE_LEGEND, YEAR, MODEL, MODEL_LEGEND and STATE. Columns MAKE AND MODEL contains a code, wherease the corresponding LEGEND columns contain human-readable names. Column YEAR contains the actual year, and no legend is necessary. Three empty dropdowns are created for Make, Year, and Model, with appropriate explanatory text ("Select the make of the car").
The Action contains:
CascadingMenus Make,Year,Model
The SQL in the Script field contains:
select MAKE, MAKE-LEGEND, YEAR, YEAR, MODEL, MODEL_LEGENDfrom CARS where STATE='[/state]' ORDER BY MAKE_LEGEND, YEAR DESC, MODEL_LEGEND

The table contains one row for every unique combination of make, year and model and state code where that model is certified. The questionnaire contains a field "state" whose value is used in the WHERE clause of the SQL query to limit the records to only those sold in the state. The ORDER BY clause sorts the resulting elements accordingly.

This Action is always executed before the page that contains the Action is composed.

Next: SQLSelect

Custom Actions: SQLSelect

This custom action allows retrieving data from a database table and inserting the values into a questionnaire.

SQLSelect when=before or when=after,question=field,question=field
Use when=before to perform this action before the page is loaded. Use when=after to perform this action when the respondent moves to the next page. If neither of these is used, when=after is the default. Follow this with a list of question names. Each question name can be followed, optionally, by an equals sign and the name of the corresponding field in the database table; if the names match, there is no need for this parameter. The Enter Script field contains the SQL statement. The statement often uses **question piping**. Security Note: If using **User Security**, the administrator must grant the questionnaire designer the right to use this custom action. Other SQLSelect parameters can be added in front of when=before or when=after. See **more parameters** below.

Example: using SQLSelect for personalization

A database table USER_PROFILES contains customer profiles, including logged in user id's, full name, region, and courtesy gift preference in fields named USER_ID, USER_NAME, REGION and GIFT. After creating a questionnaire with fields username, region, and gift, a SQLSelect action looks up the values in the table and inserts them into similar questions in the questionnaire.

A custom action is included in the questionnaire's first page:
SQLSelect when=before,username=USER_NAME,region,gift

Use this SQL statement in the Enter Script field:
SELECT USER_NAME,REGION,GIFT FROM USER_PROFILES WHERE USER_ID='[/authenticateduserid]'

Notes:

- Using when=before on the first page causes these values to be preloaded and available for personalization on the first page.The values are available in subsequent pages as well.
- region=REGION is not required because the name is the same and database fields are not case sensitive; specifying region is enough.
- [/dbtablenameprefix], if added before USER_PROFILES, will provide a prefix for the table name. This parameter is set in the ViewsFlash **servlet parameters**.
- A question of type HTML on the first page displays to the respondent their name and region:
  Hello, [/username] in [/userregion].
- The gift question could be defined as a hidden question that is used in question piping, or as the basis for displaying and hiding certain questions with **Display or Hide**, for example.

More parameters:

Add selectonlyifquestionempty=true in front of when=before or when=after to prevent the SQL query from executing if the first question already has a value. This is useful for filling in fields only once when a questionnaire is set up for revisions, or when it is possible to execute the SQLSelect statement more than once by navigating to its containing page again using the Previous or Save buttons.

Add datasource=JNDIDataSourcename to execute the SQL statement using a different data source. By default, the tables are located in the same database that all of ViewsFlash uses. But sometimes it is useful to use tables in a different schema or database altogether. In this case, create an additional JNDI Data Source and put its name in the datasource parameter. The additional database does NOT need to be the same database brand (eg., Oracle / DB2). More detail **here**.

Add errorprocess=XXXXX to specify what to do when an exception occurs. The format of this parameter can be NNNI and NNNE, where NNN is a number. NNNI will log the error to the questionnaire log and the ViewsFlash log. NNNE will also send an email to the ViewsFlash administrator. This parameter may contain the letter C or Q in front of it. If both letters are missing, and there's something wrong, the Action will be considered Invalid, questionnaire processing will stop, and the action specified in the Invalid Data section of the Security page will be performed. If this parameter begins with a C, as in

C123E, processing will continue and the error message will be shown on the next page of the questionnaire at the top. This setting can be useful during early testing. If this parameter beings with a Q, as in Q123E, the error will be logged and processing will continue.
The default value of this parameter is 123E, which stops processing, writes the cause to the log, and notifies the ViewsFlash administrator of the cause.

Next: SQLUpdate

Custom Actions: SQLUpdate

This custom action allows inserting or updating data in a database table.

SQLUpdate when=before or when=after
Use when=before to perform this action before the page is loaded. Use when=after to perform this action when the respondent moves to the next page. If neither of these is used, when=after is the default. The Enter Script field contains the SQL statement. The statement usually uses **question piping**. Security Note: If using **User Security**, the administrator must grant the questionnaire designer the right to use this custom action. Other SQLUpdate parameters can be added in front of when=before or when=after. See **more parameters** below.

Example: using SQLUpdate for activity logging and audit trails

A table UPDATES with fields ID and DATE_LASTUPDATED is used to log when a questionnaire is entered or updated.

A SQLUpdate action is included on the last page of the questionnaire:
SQLUpdate when=after

Use this SQL statement in the Enter Script field:
INSERT INTO UPDATES (DATE_LASTUPDATED, ID) VALUES (SYSDATE, [/authenticateduserid])

Notes:

- Using when=after on the last page of a questionnaire guarantees that the update will happen only when the respondent reaches the end of a questionnaire.
- [/authenticateduserid] is the tag, described in **question piping**, that contains the ID of the current questionnaire. In this example, that is the user's logged in user ID.
- [/dbtablenameprefix], if added before UPDATES, will provide a prefix for the table name. This parameter is set in the ViewsFlash **servlet parameters**.

More parameters:

Add none=1 when a SQLUpdate could result in not updating any any records. Without this parameter, an exception is reported.

Add datasource=JNDIDataSourcename to execute the SQL statement using a different data source. By default, the tables are located in the same database that all of ViewsFlash uses. But sometimes it is useful to use tables in a different schema or database altogether. In this case, create an additional JNDI Data Source and put its name in the datasource parameter. The additional database does NOT need to be the same database brand (eg., Oracle / DB2). More detail **here**.

Add errorprocess=XXXXX to specify what to do when an exception occurs. The format of this parameter can be NNNI and NNNE, where NNN is a number. NNNI will log the error to the questionnaire log and the ViewsFlash log. NNNE will also send an email to the ViewsFlash administrator. This parameter may contain the letter C or Q in front of it. If both letters are missing, and there's something wrong, the Action will be considered Invalid, questionnaire processing will stop, and the action specified in the Invalid Data section of the Security page will be performed. If this parameter begins with a C, as in C123E, processing will continue and the error message will be shown on the next page of the questionnaire at the top. This setting can be useful during early testing. If this parameter beings with a Q, as in Q123E, the error will be logged and processing will continue.
The default value of this parameter is 123E, which stops processing, writes the cause to the log, and notifies the ViewsFlash administrator of the cause.

Next: SQLDisplay

Custom Actions: SQLDisplay

This custom action allows retrieving multiple rows of data from a database table using a SQL query and displaying the data in tabular form in a questionnaire.

SQLDisplay pattern_question_name
The SQLDisplay action is always exectured before the page is loaded. pattern_question_name is the name of a question that contains an HTML pattern, such as a table. The pattern is filled with the data retreived by from a database query. The pattern_question_name question is of type Comment, not of type HTML. The Enter Script field contains the SQL statement. The statement often uses **question piping**. Security Note: If using **User Security**, the administrator must grant the questionnaire designer the right to use this custom action. Other SQLDisplay parameters can be added in front of when=before or when=after. See **more parameters** below.

Example: using SQLDisplay to list patient data

A database table PRESCRIPTIONS contains patient prescription data including patient ID, drug name, date prescribed, and physician. A patient assessment questionnaire looks up the prescriptions for the patient and displays them in tabular form.

A SQLDisplay custom action is included in the questionnaire:
SQLDisplay prescriptiontable

Use this SQL statement in the Enter Script field:
SELECT DRUG, PRESCRIBED, PHYSICIAN FROM PRESCRIPTIONS WHERE PATIENT_ID=[/patient_id]

The pattern_question_name question, of type Comment, contains the following HTML in the Comment field:

```
[/ifanydisplayrecords]<p>Records for patient [/patient_id]</p>
<table border="1" cellspacing="0" cellpadding="3">
  <tr> <th >Drug</th> <th >Prescribed</th> <th >Physician</th> </tr>
[/alldisplayrecords]
  <tr> <td>[/DRUG]</td> <td>[/PRESCRIBED]</td> <td>[/PHYSICIAN]</td> </tr>
[/endalldisplayrecords]
</table>
[/endifanydisplayrecords]
[/ifanydisplayrecords,not]No records for patient [/patient_id][/endifanydisplayrecords]
```

This produces a display like the following:

Drug	Prescribed	Physician
Lipitor	8/31/2010	Johnson
Lunesta	7/1/2009	Johnson
Lupron	5/15/1998	Jones

Notes:

- This HTML is a table with one row for table headers and one row for the fields being retrieved and displayed. Both have three columns, one for each field.
- The [/alldisplayrecords] and [/endalldisplayrecords] tags indicate to repeat the content for each row retrieved from the database.
- The [/ifanydisplayrecords] and [/endifanydisplayrecords] indicate what content to use when the database query returns rows.
- The [/ifanydisplayrecords,not] and [/endifanydisplayrecords] indicate what content to use when it does not.
- The [/DRUG], [/PRESCRIBED], and [/PHYSICIAN] tags are replaced with the values from the fields in one row of the database table.
- Instead of column names, column numbers can be used: [/1], [/2] refer to the first and second field in the SQL

qwuery.

- Question piping can also be used, such as [/patient_id]
- Other [/...] tags used in a **Form Style** can also be used.
- [/dbtablenameprefix], if added before PRESCRIPTIONS, will provide a prefix for the table name. This parameter is set in the ViewsFlash **servlet parameters**.\

More parameters:

Add datasource=JNDIDataSourcename to execute the SQL statement using a different data source. By default, the tables are located in the same database that all of ViewsFlash uses. But sometimes it is useful to use tables in a different schema or database altogether. In this case, create an additional JNDI Data Source and put its name in the datasource parameter. The additional database does NOT need to be the same database brand (eg., Oracle / DB2). More detail **here**.

Add errorprocess=XXXXX to specify what to do when an exception occurs. The format of this parameter can be NNNI and NNNE, where NNN is a number. NNNI will log the error to the questionnaire log and the ViewsFlash log. NNNE will also send an email to the ViewsFlash administrator. This parameter may contain the letter C or Q in front of it. If both letters are missing, and there's something wrong, the Action will be considered Invalid, questionnaire processing will stop, and the action specified in the Invalid Data section of the Security page will be performed. If this parameter begins with a C, as in C123E, processing will continue and the error message will be shown on the next page of the questionnaire at the top. This setting can be useful during early testing. If this parameter beings with a Q, as in Q123E, the error will be logged and processing will continue.
The default value of this parameter is 123E, which stops processing, writes the cause to the log, and notifies the ViewsFlash administrator of the cause.

Next: RandomizeQuestions

Custom Actions: RandomizeQuestions

This custom action presents the questions in the current page in a random order.

RandomizeQuestions question1,question2
This action is always performed before a page is composed. If used with no parameters, all questions on the current page are randomized. If used with two question names, only the questions between them are randomized, and the rest are presented in the original order; the two questions are included in the shuffle.

Example: a page contains questions named intro, convenience, speed, price. To present the last three questions at random and to keep the introductory question at the top, use:
RandomizeQuestions convenience,price

Next: IssueHttpQuery

Custom Actions: IssueHttpQuery

This custom action allows sending a customized URL to another application or to ViewsFlash. The customized URL can contain data entered by the respondent by using **question piping**.

IssueHttpQuery url
Sends an HTTP command to "url". "url" may use question piping such as [/fieldname] to take a value entered by the visitor and substitute it in the URL. The URL must be accessible to the ViewsFlash server.

Example:

A course evaluation survey has, among others, two questions: facultyid and facultyrating. They are used in this survey, but it would be desirable to also enter these values into another survey automatically.

When this custom action is encountered, often in the last page of a survey, the following entry in the Custom Action field will send the data from facultyid and facultyrating into a different survey whose pollid is pollingplace!poll, using **question piping**.

IssueHttpQuery <space>
http://www.yourcompany.com/ViewsFlash/servlet/viewsflash?
pollid=pollingplace!poll&cmd=tally&faculty=[/facultyid]&rating=[/facultyrating]

Note that there's only one space, after "IssueHttpQuery".

Next: Legacy Actions

Legacy Actions (formerly Custom Actions)

The following Actions are no longer recommended. Their functionality is now provided by the Script action. They continue to be supported, however, for backward compatibility.

To use these, you simply enter in the Action field the name of the Action, a space, and its parameters, as described in the following table. For more information on each action, click on its name.

Action and parameters	What it does
BeforePageComposedGoTo destinationquestion,testquestion,value,value,...	This Action is executed before a page is composed. If 'testquestion' contains any of the values in the list, the next page displayed to the recipient is the one that contains 'destinationquestion'. A ! before 'testquestion' causes branching when the condition is not true. Leaving out testquestion and values causes an unconditional branch. To accomplish this with Script, see **Branching**.
WhenPageReceivedGoTo destinationquestion,testquestion,value,value,...	This Action is executed when a page is submitted by pressing the Next or Submit buttons. If 'testquestion' contains any of the values in the list, the next page displayed to the recipient is the one that contains 'destinationquestion'. A ! before 'testquestion' causes branching when the condition is not true. Leaving out testquestion and values causes an unconditional branch. To accomplish this with Script, see **Branching**.
WeightedScore targetquestion[.decimalplaces], {sum|average},question*weight, question*weight,...	Takes the values of each question in the list multiplied by its corresponding weight, and stores the sum or average in targetquestion, using the indicated number of decimal places if present or none otherwise. To accomplish this with **Script**, use: targetquestion=(question1*.5+question2*.3) / numberanswered(question1,question2);

Next: Invite and Track

Invite and Track

Invite and Track invites participants to take a survey, tracks whether they have or not, and allows sending reminders to those who haven't.

Invite and Track sends e-mail to people in an **Invite List.** The e-mail consists of an invitation or a reminder, or the survey itself. A review of each field on this page follows.

Composing the Invitation

Select invite list. Choose from a defined list in the drop-down menu. If no lists have been defined, or to create a new list or modify an existing list, click on the Open the List link, right next to the selection box, if one is available. If no link is present, that means that the Administrator is the only person who can create and modify lists; contact your Administrator and, once the list is created, it will be visible in the Select invite list menu.

Compose E-mail. Enter a valid e-mail address in the From address field. Optionally, enter a valid Reply to address. These must be legitimate addresses to which people can reply. If they are not, there's a good chance that this e-mail will be classified as spam. In the Subject field, provide a meaningful subject line.

In Format, select Plain Text or HTML. Which one to use depends on the survey's audience. In an Intranet survey, where a company has standardized on an HTML-capable e-mail client, HTML mail is a good choice. For other audiences, Plain Text is the only safe choice.

If the survey is a single-page survey (as specified in the Setup / General page), a Body option allows choosing between Invitation and Embedded Survey. If this option does not appear, only Invitations may be sent. When a single-page survey is embedded in the e-mail, visitors will see the actual survey form in the e-mail and can respond by simply answering the questions and pressing the Submit button in the survey. Because embedded surveys must be sent in HTML format, they are recommended only for Intranet surveys where the e-mail client is expected to display HTML.

When mailing an Invitation, a text box allows entering the invitation text as either Plain Text or HTML, as chosen. Click on the write sample body link to see a draft of a very simple e-mail with ALL the correct parameters, such as the survey URL, included in it. This option is HIGHLY RECOMMENDED, as it is takes into account the type of survey and the Authentication method you are using and composes an e-mail that is just right.

To send a complex e-mail, such as an HTML-formatted invitation in a corporate standard, after pressing Submit, use the Upload button to upload a file; after uploading, the file contents will appear in the box where it can be edited further.

Authentication and Tracking

An Invitation list used for mailing must contain an e-mail address. It may also contain a User Name, which can be used to personalize an invitation or survey (see **Personalization**, below).

In order to track survey-takers progress and send them reminders, an appropriate Authentication option is required in the **Security** page. The choice of method to use depends first of all on whether the Invite List contains User IDs, as well as e-mail addresses, as follows. Please note that Basic authentication refers to both *Basic HTTP Authentication* and to *Web Application* also known as *Container* Authentication. Most single-sign on systems work with this authentication method.

In Portals. An invitation list may contain just a User ID, to specify who is allowed to take a survey. It may contain email addresses as well. When only an ID is used, the portlet must use the "Show all available" mode. The questionnaire should use Portal authentication.

No User ID's available. When the Invite list contains e-mail addresses and does not contain User ID's, it is still possible to track, by using the "Question" authentication method, in which a hidden question is created to hold the user ID, and the user ID is pre-populated using the e-mail address. To see how this works, create a hidden question, then select it in the Question authentication method, and click on Write Sample Body. You will see that the email uses a special URL for the survey, like this:
http://www.yourcompany.com/ViewsFlash/servlet/viewsflash?userid=[/email_in_list]&pollid=pollingplace!poll&cmd=page
This makes it possible to track a survey's progress and send reminders when all that is available is a list of e-mail addresses. The drawback is that, because the entire URL is visible inside the e-mail, there is no guarantee that the person who is taking the survey is actually that person, since the e-mail can be forwarded to anyone, and the recipient can take the survey simply by clicking on the survey URL. For that reason, when using Question authentication, we rate the security of this method Low.

It is also possible to choose No authentication, or Anonymous authentication, or any other method when the Invite list doesn't include User ID's. In this case, because there is no known relationship between e-mail addresses and User ID's, the Tracking options are turned off, and there's no way to send reminders to people who have not completed a survey. It is still possible to

send a reminder e-mail to everyone, keeping in mind that the e-mail will be received both by those who have completed the survey and those who haven't.

User ID available. When the Invite list contains User IDs as well as e-mail addresses, it is possible to set up highly secure surveys. Begin by choosing the Basic, IIS, Cookie, Header, Attribute or Role authentication methods in **Security**. All these methods rely on system-wide or network-wide authentication systems already in place. When the visitor is presented with the Survey URL, they will be asked to authenticate appropriately, usually with their user ID and password, biometric information, magnetic card, etc. The ViewsFlash application, after the visitor is authenticated, will know the user ID for the visitor. It will then check further to see that the visitor is on the Invite list, unless a Security page option specifically allows visitors not on the list to take the survey.

With an authenticated user ID on the list, Tracking is enabled, and the survey process is Highly secure.

UserID and Passwords. Occasionally, a situation arises where there is no central authentication authority, but reasonably strong authentication is desired. In this case, make sure the Invite list contains both UserID's and passwords, copied from some central repository. Then choose the Form Authentication option in Security. When using these options, visitors will be presented with a Login page at the beginning of the survey, asking them to provide an ID and password that must match the entries in the Invite list. The Invitation may or may not include the User ID and the password, depending on the likelihood that the visitor will know what it is. For example, if the User ID is the employee's social security number and the password is their birth date, there is no need to include them in the e-mail Invite and the survey security is High. Conversely, if the User ID and password are not likely to be known by visitors, then both must be included in the e-mail, whose security will now be Low. To see how to include User IDs and passwords in the e-mail see **Personalization** below.

EMail addresses only. It is not unusual to have a list of members, with e-mail addresses, that need to fill out a questionnaire, but there is no web server authentication or single sign-on system available that they can use to register before completing the questionnaire. For this case, use Form Authentication, and set up an invite list the uses the email address as a user ID. When the login form is displayed, there will be no password field, the entered email address will be compared against the entries on the list, and the user will be rejected if the email address is not on the list. It is also possible to send inviation e-mails to these users by using the same field for both the user ID and the e-mail address.

Here are some examples of common authentication scenarios. The suggested authentication methods are listed most recommended first. The security level refers to the ability to impersonate a survey taker; a level of High can be achieved by configuring the web server and application server environments appropriately.

Scenario	User ID's on Invite List	Typical authentication method	Security level
Employees authenticated on Intranet	Yes	Basic, IIS	High
Members who have an e-mail address	Yes	Form, without password	Medium (email address can be guessed)
Special purpose IDs and passwords	Yes	Form	Low-High (depending on UserID and Password visibility in invitation)
Extranet visitors using single-sign-on	Yes	Basic, IIS	High
Portal with logged in visitors	Yes	Portal	High
Portal with anonymous visitors	No	Portal	Low
Visitors who left an e-mail address	No	Question	Low; use Form instead
Anonymous	No	Cookie	Low
Anonymous	No	Session with captcha	Low, but better than just Cookie

Please note again that the write sample body link takes all these settings into account. Select an Invite List, select an Authentication method in Security, then use this link to produce a sample message with the correct survey URL for the options chosen.

Inviting and Tracking

The Survey Tracking section includes buttons for sending the e-mail prepared above to different groups of people.

New. Click on this button to e-mail people who have never been invited before. If the list has changed (e.g., employees added since the first time mail was sent, or a different WHERE clause selects a different pool of recipients), only those who have not received an e-mail before will be sent one.

Retry. Click this button to try sending the current e-mail text to those people who were invited but whose emails failed to deliver. This is useful in situations where a class of recipients, such as people inside or outside an intranet, were not reachable by the mail server because of a configuration error.

Remind. Click on either of these buttons to send the current e-mail text. The first Remind button e-mails those people who have not started completing the survey at all. The second Remind button e-mails those who have started the survey but haven't completed it yet; in the case of a single-page survey, only one of these buttons is present.

Everyone. Click on this button to send the current e-mail text to everyone, regardless of whether they have already started or finished the survey.

Cancel. When working with a large list, if you realize you have pressed one of these buttons in error, pressing Cancel will stop further e-mails from being sent.

If the person using an Invitation List has the right to modify it, two links appear: **View List** and **View List in Excel**. These links provide a detailed view of who has been invited to participate, who has completed their questionnaire, and their characteristics.

Personalization and Mail Merge

All fields in the Invite List can be used to personalize the body of an e-mail invitation and the survey as well. To use them, simply include the following tags in the body of the invitation, or in a survey title or a survey question:

Field	Tag
User name	[/username_in_list]
User ID	[/userid_in_list]
E-mail address	[/email_in_list]
Password	[/password_in_list]

For example, consider the following invitation:
Dear [/username_in_list],
We would like to invite you to take a survey.

When this is sent to Peter Duchin, it will look like:
Dear Peter Duchin,
We would like to invite you to take a survey.

The same notation can be used inside a survey, a survey question, or a survey response page. The mechanism used is essentially the same as for **Piped Variables**.

Additional fields can be used in the body of the email when the **Participants API** is used. Use the names of the additional columns available in the VWFTRACKING emailing table like this:
The survey concernts building [/custom], located in region [/region].

How to Test and Run the survey

After pressing the Submit button in step 3, send yourself a test e-mail by filling in your e-mail address and pressing Send test e-mail. An e-mail will be sent to that address. The user name, if used in personalization, will be "Tom Jefferson". This is only a preliminary test, designed to see any obvious problems. The complete testing sequence is shown below.

1. Construct the survey questionnaire and response pages and check with Preview. On the Publish page, open the survey by clicking on Now next to the opening date.

2. Select an Authentication method appropriate to the survey. See **Authentication** above.

3. Create an Invite List with fields that are appropriate to the authentication method. See **Fields to Use**. If you have the right to modify the list, create a small comma-delimited list of test users; if not, ask your administrator to create two lists with the same fields: one for testing and one for the production survey. Use View List to make sure that the list contains who and what you expect.

4. Create an e-mail invitation, selecting the test Invite List and other parameters. Examine the Status field for any warnings and correct any problems indicated.

5. Send a test e-mail to yourself and correct any problems you see.

6. Use New to send an e-mail to the test population of people who will examine and sign-off on the survey. Gather feedback, make any modifications required.

7. To repeat the test, go to the Publish page and be sure to check "Delete all data" and press Submit. This will remove all information about the test surveys gathered so far or in progress.

8. Once everything is ready to go, go to the Publish page, check "Delete all data" again and press Submit.

9. Go to the Invite page. If using a test list, switch to the production list and press Now to send the initial invitations.

10. After a while (days usually), return to the Invite page, and send a Reminder if appropriate.

11. Once you are satisfied with the response, close the survey by entering a closing date in the Publish page.

12. Analyze the survey data and results.

Next: **Invite Lists**

Invite Lists

Invite Lists specify a list of survey participants. A survey can use an Invite List to send e-mail to participants inviting them to take the survey, or include the actual survey in the e-mail. It can also track who has started and who has completed the survey and send e-mail reminders to those who haven't. An e-mail can include an invitation or the actual survey itself. For complete information on how to use an Invite list with a survey, see **Invite**. An Invite List can be shared by many surveys.

Invite lists are defined using the Invite Lists page. Access to this page is normally restricted to the ViewsFlash administrator, who can relax this restriction from the Administration page. The Invite Lists page allows you to create a new list and to modify or remove an existing list.

Defining an Invite List

An invite list can come from a database, from a comma-delimited file which is uploaded, or from a comma-delimited file which is entered in right in the form. An invite list can contain more than one of these and can select a different source at different times; for example, the list in the form can contain a set of test participants, while the actual list resides in a database.

After the list is defined, the number of people in the list is displayed at the top of the page. When logged in as an administrator, two menu items on the left hand side, View List and View in Excel, allow previewing the list before using it. View List shows the list in the browser; View in Excel shows the list in that application.

Database lists

The first option in the Invite List page allows defining a list that resides in a database table, one person per row. Spaces are provided for entering the table name and the names of the columns that contain the e-mail address, person's name, user ID, and password. See **Fields To Use** below for more information about what fields to use when. If the list doesn't contain passwords, for example, leave that column blank.

The WHERE clause allows limiting the list to only a subset of people in the table. For example, if the table contains a field named "DEPARTMENT", then specifying:
DEPARTMENT='Human Resources"
will include only people the HR department.

LDAP Directory lists

The second option allows defining an invite list based on the contents of an LDAP directory. Spaces are provided to enter the attribute names of each person's email address, name and UserID. See **Fields To Use** below for more information about what fields to use when.

The Base name field is used to specify the LDAP base name of the directory containing the personal entries. In addition, the Filter field allows specification of an LDAP filter expression, if required to locate a person's LDAP entry. A checkbox, Search subtree, should be checked if the search for personal entries should recur into subdirectories. Note that in order to use an LDAP server, some application parameters must be configured. See **Using LDAP**.

Comma-delimited lists

The third option allows defining a comma-delimited list, one person per line. A comma delimited list uses the standard CSV format, where fields are separated by commas and can be optionally surrounded by quotes, as in these two examples:

Tom.Jones@yourcompany.com,Thomas Jones,TomJones
"Tom.Jones@yourcompany.com","Thomas Jones","TomJones"

The check boxes indicate what fields the list contains, in that order. First is an e-mail address, if used. If a user name is available, check that box and that becomes the second field in each line. If a user ID is available, that becomes the next field, and so on. See **Fields to Use** below for more information about what fields to use when.

The comma-delimited list can be entered right in the form, by selecting "Enter the list here" and typing or cutting and pasting the list right into the form.

For larger lists, create the comma-delimited list first, and upload it from your workstation by selecting "Upload a file". If the file is larger than 1MB, change the maximum size. Before uploading the file, be sure to press Submit first. Then press Upload, and in the Upload File page, browse to the location of the comma-delimited file and press Upload. Large files may take a

while to upload.

Even though we refer to these lists as comma-delimited, if your list is tab-delimited, ViewsFlash will interpret that properly. A tab delimited list looks like this:
Tom.Jones@yourcompany.com(tab)Thomas Jones(tab)TomJones
A tab-delimited list uses only tabs to separate fields; any quotes become part of the field, so normally quotes aren't used. One notable exception is this form:
"Peter Duchin"<peter.duchin@yourcompany.com>(tab)Peter Duchin(tab)PeterDuchin

Fields to Use

An Invite list may contain an e-mail address. A User Name is optional, and can be used to personalize an invitation or survey with the person's name; if this type of personalization is not desired, there is no need to provide a User Name.

A User ID, when present, is used in combination with a survey's Authentication method, to track who has taken a particular survey and send e-mail reminders to those who haven't. Normally, using an Invite List restricts survey participants to only the people on the Invite List, but this can be overridden in a given survey. A password can be used when using the "Form" type of Authentication. See **Invite Authentication** for complete information on setting up a survey with different kinds of authentication and the interactions between the fields on the Invite List.

Dynamic Lists

A database list, specially with a WHERE clause, is not static. For example, one can send a survey invitation on one day to people in one department, and next week to people in a different department, by changing the WHERE clause. See **Invite** for more information.

Next: **Accessibility**

Accessibility

Questionnaires created using the Standard ViewsFlash 5 styles meet very high accessibility standards. By using these style templates, the forms created are:
- WCAG Priority 2 compliant, with many Priority 3 elements as well
- Section 508 compliant
- XHTML 1.0 Transitional compliant
- i18n capable; accessible multiple language surveys can be created.

These templates accomplish these goals by using the following techniques:
- using CSS styles exclusively
- using <label> and <fieldset> tags with all form elements, and <alt> tags with images
- extra highlighting for currently active form fields in browsers that support it, like Firefox
- judicious use of tables, with row and column headers
- javascript form field validation backed by server side validation for screen readers that turn off javascript
- form field validation errors reported with a link that takes user directly to field in error

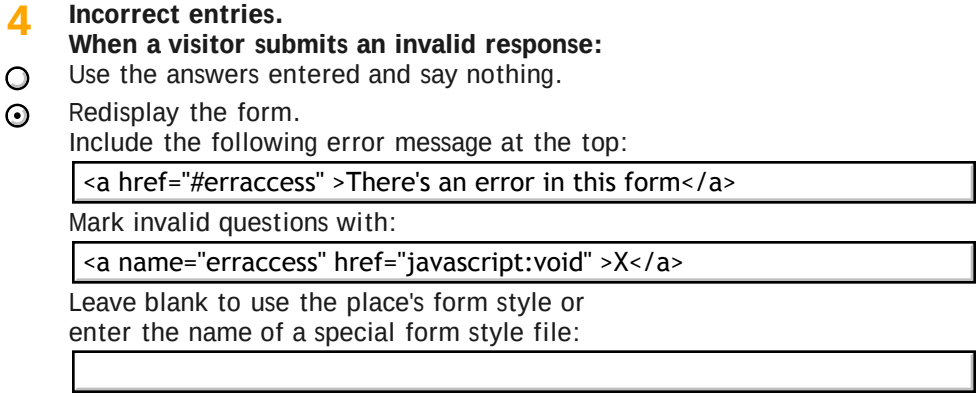
The templates can be further refined, as usual.

To ensure the highest level of compliance, create questionnaires with the following settings.

1 Select the Standard_ style in the place settings. This is the default setting.

2 In the Question page, use the [x] Question must be answered check box , and the "values must be between" fields in Text boxes. Additional arbitrary server side validation can be done with the Script Action.

3 In the Security page, include the following values in item #4, Incorrect Entries:



4 The Language page now includes fields for describing the left to right or right to left orientation of the language used in the questionnaire. If not filled, the servers' default Locale settings are used.

All these procedures can be used in languages other than English. The standard techniques for creating multiple language surveys work with these styles.

Resources
W3C Content Accessibility Guidelines
Section 508 Access Board
Comparison WCAG and section 508 by Jim Thatcher
Accessible HTML Forms by Ian Lloyd
Understanding Web Accessibility by Shawn Lawton Henry
Bruce Lawson
Accessify.com

Style templates enhancements used
A new [/answerno] tag can now be used in form and question styles. It is used to create element IDs.
The new [/direction] tag can be used in form styles. It is used to create the direction HTML tags.

Next: **Sampling**

Sampling and pop-up surveys

Introduction

Polls and surveys created with ViewsFlash are usually published on the web site in places where they can be seen and voted on as often as possible. There are times, however, where you may want to be as discreet as possible, such as when collecting marketing data. For example, you may want to ask visitors what they think about this section of the site and use that information to improve it. Or, you may want to know why they pressed the Cancel button instead of the Order Now button. You could even ask them, perhaps offering them an incentive, to provide you with personal information for demographic analysis or profiling.

In all of these uses, it may not be desirable, at least at the outset, to force every visitor to answer these questions. You can always invite them, by offering them a "take a survey" link or banner. Another effective method involves having a survey, or a survey invitation, pop up for some, but not all, visitors. ViewsFlash provides the capabilities needed to both pop up a survey, and most importantly, to present the survey to a statistically valid sample of visitors.

Sampling on the Internet has often been shortchanged, without sufficient care being paid to its fundamentals. First, ViewsFlash helps you decide how many samples you need to have a satisfactory confidence level, paying due attention to correcting sample size for small populations, which can affect the results of sites or sections of sites that are not heavily visited yet -- which is precisely one of the better times to ask people for feedback, for example. Second, ViewsFlash is careful to present the survey only once to each visitor, and most importantly, gives every visitor the same chance of being chosen as any other visitor, thus fulfilling the fundamental premise required for a representative sample.

Other software and services often miscalculate the sample size required when the number of visitors is small, typically claiming that size doesn't matter. More dangerously, they also present a survey every Nth page view, which will result in a sample heavily biased towards the views of frequent visitors, who will be much more likely to be chosen for the survey. While the views and habits of frequent visitors are important, any survey that tries to understand those views from all visitors to the site must be able to give equal consideration to infrequent visitors, whose views may be just as important to the growth of the site. Worse, not being aware of the significance of this sampling methodology can cause you to make decisions based on information that can be completely wrong, as the views and profiles of frequent visitors are likely to be significantly different from those visitors who are just trying out the site or use it not very often.

Methodology

Recapping, ViewsFlash's two objectives are to provide a sufficiently large sample, and to provide a representative random sample. The Calculate button on the Define Sample page calculates the correct sample size, given what you know about visitors. For situations where the number of visitors is small, this calculator can give very different answers from other sample size calculations which ignore the correction required for small populations. For example, here are the recommended sample sizes to be 95% confident that the answers to a question will be within 5% of the answers if everybody had been surveyed instead of asking only the people in the sample.

Visitors expected	100	500	1000	5000	10000
Sample size, corrected for population size (ViewsFlash)	80	217	278	357	371
Sample size without correction (others)	384	384	384	384	384

As expected, the correction is negligible for large numbers of visitors, but when the number of visitors is small the correction is very significant. If you only expect 100 visitors, the standard answer of 384 can only mean that you must interview everybody, whereas you only need to ask 80 people. For 500 visitors, only 217 interviews are needed, as opposite to 384, and so on.

To make the sample random, it is not sufficient to write JavaScript that pops up a window when a random number produces a certain result. This will result in two problems: a visitor may be asked for an interview more than once, and frequent visitors are more likely to be asked. ViewsFlash solves this problem by providing an HTTP API that, when implemented on a web page, causes a window to pop up for every Nth visitor, and tagging visitors with cookies as they go by so that once a visitor has been a candidate for a survey, he is only given one chance, no matter how often he comes by.

Configuration Summary

Fill out the settings on the Define Sampling page and submit it. Then, add JavaScript to the page or pages where the survey may pop-up. Last, Publish the poll. A detailed explanation follows:

How to do it, step by step

1 The **Define Sample** screen first asks to specify the Sampling Method. There are three choices:

Don't sample. This is the default setting, which disables these options.

Fast Sampling presents every visitor with the survey in a pop-up window once, until the desired number of *completed surveys* have been collected or the Publication ending date is reached. This is a terrific way to get a representative sample of a desired size, provided that the site does not exhibit a pattern of different use during the time it will take to collect the sample. Since this is not usually the case, this method should be used only when gathering the information as fast as possible is more important than having results that are representative of visitors' views over a given period, such as a week. For example, after installing a new video capability on the front page which is technically challenging, it would be useful to know what the next 1,000 people think about it so that it can be removed if it is providing a less than satisfactory user experience.

Besides specifying the number of surveys to collect, you will need to set up JavaScript, as explained below. You can also use the calculator in Random Sampling and copy the calculated sample size to this section.

Random Sampling.

The first two fields, collect __ responses and pop up the survey every __ th visitor, can be filled in by hand, or the can be calculated by filling in the other fields in this section and pressing the Calculate button. Here's how that works. You chose the confidence level (90, 95, or 99%) for your results from the pull down menu, and the degree of precision for your answers as plus or minus a percentage. Then you estimate how many unique visitors will be viewing the page where the pop-up survey can appear during the period when the survey will be Published. You also estimate how many people will be actually completing the survey. When you press the Calculate button, the first two fields will be filled with the corresponding values. ViewsFlash will use the every Nth visitor number as its criteria for deciding who should be presented with a survey and who should not.

2 **Displaying the survey in a pop-up window** defines the actual JavaScript code that will run on the browser for those visitors who are chosen to be presented with the survey. There is no need to modify it. If you do, examine it carefully first. Note that very creative applications of this technology are possible by modifying this script.

3 Press Submit to set the options you select.

4 **Very Important!** Add a snippet of JavaScript to the page or pages where the survey may pop up. The JavaScript to copy is presented on a color background at the Define Sample page, from where it can be cut and pasted into the <head> section of the page. The snippet typically looks like this:

```
<script language="javascript" src="http://yourdomain.com/ViewsFlash/servlet/viewsflash?vwfsample=yoursurvey">
</script>
```

This snippet calls the ViewsFlash server and asks for a further bit of JavaScript to execute when the page is loaded. ViewsFlash determines whether this visitor should be presented with a survey or not, and then return either a single space, which means there's nothing to do and no survey will pop up, or the JavaScript specified in the Define Sample page, which typically looks like this after ViewsFlash has removed all the [/...] tags and replaced them with the appropriate values:

```
window.open ('http://yourdomain.com/ViewsFlash/servlet/viewsflash?cmd=getsurvey&pollid=pollingplace!poll&userid=1',
'vwfsurvey',
'width=400,height=300,x=100,y=100,resizable=1,scrollbars=1,menubar=0,toolbar=0')
```

This JavaScript opens a window named "vwfsurvey", with dimensions and properties as given, and in that window the survey will be displayed by the http reference to viewsflash with the "getsurvey" command. Simultaneously, ViewsFlash will toss a cookie to the visitor's browser to prevent her from being eligible to participate in the survey twice.

5 Finally, go to the Publish page and set up the starting and ending dates and times for the survey. The start time should be about the same time when you insert the JavaScript into the web pages where the survey will pop up. It does not matter if it is slightly before or after. No surveys will pop up until the JavaScript is installed in the page where the survey can pop up AND the Publication opening date arrives. Similarly, once the JavaScript is removed or the Publication end date arrives, no more surveys will pop up, nor will any more surveys pop up once the requested number of *completed* surveys (responses) has been gathered.

Schedule the survey for a period of time that is representative of the life cycle of your site's content. For example, a stock market site could run a survey for one 24-hour week day, as visitors each day of the week should be the same as any other weekday. A news site could be treated as a daily site or a weekly site, if its content changes significantly for weekend programming. Other sites may be steady-state sites, where there is no reason to believe that visitors change from one day to another. The period of time should also be long enough to make sure that there will be enough visits to the hosting page so

as to provide the desired number of surveys; analyzing web logs for that page is one way to ascertain this information.

Monitoring the survey

Once the survey is up and running, visitors will be presented with the survey and some of them will be responding. You can get a good idea of how the survey is progressing by using the View Log command on the toolbar at the left of the screen. This log is updated periodically as voting progresses, and is updated with a final report when the Publication ending date is reached, or the requested number of surveys are completed, whichever happens first.

Changing settings after going live

You can hasten the end of a survey by changing its Publication ending date. Sampling begins anew whenever you change Publication dates.

You can also change the Sampling parameters while the survey is in progress. For example, if you are not gathering surveys fast enough, you can change the sampling rate (every Nth visitor) to accelerate the pace at which they are gathered. Some of the statistics in the log will be reset, but sampling will not be affected.

Technical Considerations

When used for sampling, the ViewsFlash server is required to respond to every page where a survey may pop up. This response is very fast, but the amount of work required can be huge. For example, installing this on the front page of a portal site could result in millions of hits per day and incapacitate even a powerful ViewsFlash server. On a 300 MHZ Intel machine, ViewsFlash completes these requests in under 10 milliseconds CPU time. Use this as a guideline for where to put these surveys. In a very large site, ViewsFlash should be run in a distributed configuration; see **Architecture**.

Next: Printing

Questionnaire URLs

All functionality of the ViewsFlash application is accessed using a set of stable and predictable URLs, described in this section.

Every URL includes protocol, domain, port, context, servlet and query string parameters.
Example: http://yourhost.com/ViewsFlash/servlet/viewsflash?cmd=page&pollid=Examples!car_purchase

Protocol, domain, and port

Protocol. Usually, questionnaires use http:// (not secure) protocol. The Security page can require the use of https:// instead. Some organizations deploy ViewsFlash, usually with a proxy server, so that all requests to the application must use https.

Domain. Always use the fully qualified domain name of the server where ViewsFlash is deployed. Avoid using IP addresses, localhost, and intranet names that omit the fully qualified domain name. The reason for this is that under those conditions it is often the case that sessions and cookies (which can be used with ViewsFlash for authentication by requesting their use explicitly) do not work properly.

Port. The application server usually uses the default ports (80 for http:// and 443 for https://) for access to the server and in those cases it is not necessary to specify the port. It is possible to deploy ViewsFlash on any port, and when doing so, include the port in the URL, such as .../yourcompany.com:8080
Web Services API commands are usually configured to use a port other than 80 and therefore must include the port.

Context

The ViewsFlash application is normally deployed as /ViewsFlash, although this is not a requirement and any name can be used. In older versions of WebSphere Portal, the VIewsFlash application is deployed at an unpredictable context, such as /wps/PA_134ukf04.

The servlet

Following the Context, there are three servlets used in VIewsFlash:

/servlet/vfadmin is used for the ViewsFlash administrative user interface when **application security** is enabled.It requires that the user sign on to the application server.

/servlet/viewsflash is used for questionnaires and for the Web Sevices API.

/servlet/vfauth is used for questionnaires that choose "Basic" authentication in their Security settings. See also **application security**.

In summary, the following are the most common URLs used by ViewsFlash with standard out of the box settings:
http://www.yourcompany.com/ViewsFlash/servlet/viewsflash
http://www.yourcompany.com/ViewsFlash/servlet/vfadmin
http://www.yourcompany.com/ViewsFlash/servlet/vfauth

Displaying a questionnaire

To display a questionnaire in the browser, use these parameters:
?cmd=page&pollid=[/pollid]
This same URL is used to resume entering data into a questionnaire that is in progress after the user presses the Save button.

Specifying the questionnaire's ID

When displaying a new questionnaire for the first time, depending on the authentication method used, the standard URL automatically creates the questionnaire ID.
When resuming a questionnaire after a Save, the ID must be preset using the authentication method used by the questionnaire questionnaire, such as setting a session attribute.
When a questionnaire's authentication method is Question with a generic field, add this parameter to the URL, replacing XXXXX with the questionnaire's ID:
&vwfqueryauthenticateduserid=XXXXX
When the method is Question with a specific question, such as "formid", use the question name instead:
&formid=XXXXX

Predefined tags for URLs

When constructing URLs in a questionnaire, the following predefined tags are used to refer to information about the questionnaire where they are used:

[/webappname] usually becomes /ViewsFlash, the application Context. In WebSphere Portal's older versions, the name is not predictable and this tag is the only way to get the context reliably.

[/pollid] becomes place!questionnaire, the standard way to reference to a questionnaire that combines its Place and its Name.

[/authenticateduserid] can be used with &vwfqueryauthenticateduserid= (see above) to refer to the ID of the questionnaire that is being filled out currently.

Printing a questionnaire

To print a questionnare in progress, use this URI:
[/webappname]/servlet/viewsflash?cmd=page&pollid=[/pollid]&showallquestions=yes
This will display all data entered into the questionnaire as a single page in the browser. When it is printed, each page will be printed on its own piece of paper.
In a portal using Portlet authentication, use this URL instead:
[/webappname]/Router.jsp?cmd=page&pollid=[/pollid]&showallquestions=yes

It is possible to print and view the data in a response after it has been submitted. The **Data Review and Modification API** must be enabled, the ID of the questionnaire must be set, and the Security page must allow reviewing data by the Visitor. When all these conditions are met, use a URL like this:
[/webappname]?cmd=reviewdata&pollid=[/pollid]

Next: Live Tallying

Live tallying responses

1 This page has check boxes for every question asked. When checked, that particular field will be tabulated and will be available for display in reports as needed.

Text questions and hidden questions CAN be tallied by checking their box on this page. This is fine for user-entered data such as age, year of birth, or country of origin. Note that the "values between and" option on the **Question** page can be used to validate the entered data.

A hidden question is used often to store a calculated score. The score can be tallied by checking its box on this page.

For multiple line freeform fields, such as a suggestion box, it is best to leave tallying off.

Because tallying open-ended questions can be very expensive when used by mistake (such as when tallying a userid!), a limit to the maximum values to be tallied is imposed, which is set on this page. The default value is 250; override it if necessary.

2 There are also check boxes to turn on checking for inappropriate language. In freeform fields, this may be desirable. The dictionary used can be changed by the Administrator by changing the appropriate file on the ViewsFlash server.

Press Submit to Save your changes.

Next: **Saving data**

Saving data

While a participant is filling out the questionnaire, all data entered is stored in a temporary location. Only when the participant presses the Submit button is the data written to permanent storage. The Save page specifies where to save the data.

The first option is to save the data on the server, as a .txt file. This option is only recommended when ViewsFlash is not hosted on a database.

The second option is to write the data to a table in the database. Each response is one row in the table, and each question is one column in the table; question names are used as column names. This option is recommended as the most natural way to store questionnaire data. This option requires that the database administrator grant the ViewsFlash application the right to create tables, which may not be possible, in which case the third option is recommended.

The third option writes the data from all questionnaires where this option is used to a "Normalized database", which is a single table (VWFDATA), in the format of one row per question per response. For example, a questionnaire with 10 questions will write 10 rows for each response; the response includes the name of the question. Because this table is created only once, the ViewsFlash application does not need the right to create database tables. ViewsFlash can be configured to disable any of these options, in which case they will not be available on this page. For in depth information, see **Using a database.**

The analysis and data retrieval functions of ViewsFlash work equally well with either database format.

Some questionnaires require saving very large text fields. These fields can only be saved in the Noramalized database and text formats. If a very large text field is stored in the questionnaire's own table, the response will be truncated at the maximum possible field width, which can range from 4000 to 667 characters, depending on the database and the use of Unicode text such as Japanese, Chinese, Arabic, etc.. In most instances, there is no practical limit on the amount of text that can be saved in the Normalized database. For complete information, see **Using a Database**.

1 Click on Show Questions to see which questions are saved. Only those that are checked will be saved. Use the All button to select all the questions. Certain question types don't save data, such as HTML, Actions, and Comments.

2 The Date Stamp and Time Stamp check boxes, when checked, will save the response's date and time of day.

The Browser Value check box (Named Cookie before release 3.4), when checked, will use the specified value from the user's browser and HTTP request and store its value in the data file. This can be useful for correlating data gathered by ViewsFlash with other data previously gathered, using for example an identification cookie tossed by other data gathering applications, or the IP address, or the value of an HTTP header. What is saved is determined by the contents of the Browser Value field, as follows:

cookie=cookiename	Saves the content of a cookie by that name
header=headername	Saves the content of an HTTP header by that name
value=valuename	Saves the content of a servlet session value by that name
attribute=attributename	Saves the content of a servlet session attribute by that name
cookiename	Saves the content of a cookie by that name
ipaddress	Saves the IP address of the originating request

For example, cookie=MYCOOKIE will save the value of a cookie by that name. header=X-FORWARDED-FOR will save the value of that HTTP header.

The authenticated ID, determined by Authentication in the Settings / Security page, can be saved by checking the Authenticated User ID specified in Security box.

3 When "Save Data as Text File" is checked, ViewsFlash will record the data from each response as one line in an ASCII file. The data can be viewed using **Data Retrieval** in a variety of formats, including Excel.

Each response is recorded as fields, separated by either Tab characters or in quoted, comma-delimited format ("a","b","1 2 3"). It is possible, by checking the appropriate box, to save the question names as the first record of the file. Lines can be terminated with a carriage return and linefeed or by a linefeed only.

When using quoted, comma delimited format, any line breaks entered by visitors in multi-line text entry forms are converted to spaces, and double quotes (") are replaced by single quotes ('), so that they can conform to this format and import successfully into other software.

4 When "Save data in its own database table" is checked, all responses for this questionnaire are stored in their own database table, one row per response, one column per question. ViewsFlash can create the database table itself, or can use an already created table. It can optionally include the names of the place and the questionnaire. It can also write a unique sequence number to a field (these sequence numbers may not be contiguous).

If the table will be used by other applications, it may be useful to create table indices. Enter one or more question names in the space provided, and an index will be created for each question. If the index should contain unique values, prefix the question name with an asterisk, such as *formnumber.

Once the questionnaire is open and respondents are filling out questionnaires, the data can be viewed by using **Data Retrieval** in a variety of formats, including Excel.

This format is a questionnaire's natural format for storing data, and other applications such as report generators find this format very easy to work with.

5 When "Save in Normalized Database" is checked, ViewsFlash will store responses in a common database table named VWFDATA, which is shared by all questionnaires that use this option. Data is stored as one row per question; a questionnaire with 50 questions will create up to 50 rows if all questions are answered. Data can be retrieved using **Data Retrieval** in a variety of formats, including Excel.

This format is difficult for other applications to work with. It is recommended only when Database Administrators do not allow storing questionnaire data in their own tables.

Next: **E-mailing responses**

E-mailing responses

1 When "Send Data as E-mail" is checked, ViewsFlash will e-mail the data from each response as soon as the response is recorded.

The e-mail's body can contain a personalized message, which should use **question piping** tags, like this:

A sweepstakes response has been entered by [/name], of [/city], [/state].
which will fill out to
A sweepstakes response has been entered by John Young, of Kalamazoo, Michigan.

To use a file in the server's file system for the body, enter its complete path in the "use this file for the email body" field.

The e-mail will be sent to the indicated recipient. This allows the gathered data to be exported to customer-service e-mail applications, for example.

2 When the e-mail body is left empty, a body will be generated which includes the questions and their answer values, like this:
name=Tom Jones
color=red
age=25

Click on Show Questions to see the list of questions and to select which should be included in the e-mail..

The Date Stamp and Time Stamp checkboxes, when checked, will include fields with the date and time of day, respectively, in the ViewsFlash server time zone.

Browser Information, when checked, will include the data item specified in the **Save** page under Browser information.

3 Specify the e-mail addresses of the persons to receive the e-mail, separated by commas. They must be valid addresses. A from-address and Subject are required so that the recipient can identify the e-mail's sender and content.

E-mails are only sent the first time a questionnaire is entered. If revising data has been enabled for a questionnaire, e-mails are not sent when the data is revised, unless the "Send e-mail when questionnaire is updated" box is checked.

A test email can be sent by checking that box. Note that question piping will not be available in the test-email.

Next: **Personalized e-mail**

Personalized e-mail

Creating the email body
In the **E-mail** page, you can specify the body of an email. This body can contain any of the tags used for **question piping**, and those normally used in a response page.
The most useful tags for an e-mail are:

[/QuestionName] will be replaced with how the visitor answered this question. For example, if a question "name" asks the visitor for their name, then:
Dear [/name]:
will produce a personalized greeting.

[/ifresponded,QuestionName,Value] ... [/endifresponded] allows you to tailor the e-mail content depending on how visitors answer a particular question. For example, if visitors tell you how much TV they watch by using values 1,2,3,4,5 in question TV, you could give a warning to heavy TV users with:
[/ifresponded,TV,5] You may be watching too much TV! [/endifresponded]
Text between these tags can include other tags; this can also be used for quiz responses, for example.

Sending the mail to visitors and to you
If you ask for visitor's names and e-mail addresses, use the following in the To: field to e-mail them the personalized response, as well as to yourself:
[/name]<[/email]>,yourself@yourcompany.com

You can use these [/QuestionName] tags in the To, From, and Subject entries; you can send a copy to multiple recipients by separating their names in the To field with commas.

Pitfalls to avoid
Be sure that your Administrator has hooked up ViewsFlash to an SMTP server that allows e-mail relaying, or all your e-mail responses will go nowhere.

A from-address and Subject are required so that the recipient can identify the e-mail's sender and content.

If you expect that a good part of your audience cannot read HTML mail, then create your template without HTML tags. Otherwise, use simple, lowest-common denominator HTML in your message.

Next: Data summary

Summary and Live Tally

This page presents an instant analysis of responses received and question tallies:

1350 responses started
1297 responses complete
0 rejected as duplicates

When the questionnaire is deployed as a stand-alone application (not in a portal), "responses started" shows how many respondents have seen the first or subsequent pages of the questionnaire. "Responses completed" indicates the number of questionnaires that have been completed and whose responses have been written to the database. "rejected as duplicates" counts attempts at responding more than once.

When the questionnaire is deployed in a portal, "responses completed" and "rejected as duplicates" have the same meaning. "Responses started" depends on how the questionnaire is deployed in the portal. If the questionnaire is embedded so that it is visible as soon as the respondent visits the page where the questionnaire is embedded, this number simply counts the number of visitors who have seen the page. If the questionnaire requires the user to click on a "fill out questionnaire" link, or if the portlet is configured to show a list of all eligible questionnaires, this number counts the number of visitors who have actually clicked on the link and opened up the first or subsequent pages of the questionnaire.

Live Tally presents a quick, no-options, real-time analysis of responses. The questions listed are selected in the **Settings / Tally** page. Here's an example:

Service	Value	N	%
5	5	5	50 %
1	1	2	20 %
2	2	1	10 %
3	3	1	10 %
4	4	1	10 %

Data provides a way to examine data from completed responses as well as responses that are only partially complete and have not been committed to the database yet.

Next: Data retrieval query

Installing ViewsFlash

For architecture and in-depth discussion of all installation and deployment issues, see **System Administration**.
For detailed database information, see **Using a database**.
For detailed information on application security, see **Application Security**.
For a simple initial development deployment this section is usually sufficient.

For how to deploy the self-contained setup.exe Windows evaluation download, see these **instructions**.

For how to deploy the Cogix Survey portlet in WebSphere, Weblogic, Sun, Vignette or Liferay portals, read these instructions as a general guideline, and then follow the specific instructions provided at **www.cogix.com** for that portal in the JSR 168 section.

Deployment complexity depends on whether a database will be used now and whether application secutiry will be used now. Both can be added after initial deployment. Without a database, email invitations and staging are disabled. Without application security, weak password protection is used.

Installation summary:

- Download ViewsFlash.war
- If using a database, configure a Data Source in the application server and optionally create the ViewsFlash tables
- If using application security, implement the security constraint roles defined in WEB-INF/web.xml
- Provide ViewsFlash with servlet parameters in a viewsflash.properties file or in WEB-INF/web.xml
- Deploy ViewsFlash.war application in the application server
- Check for successful installation
- Perform additional setup tasks
- Troubleshoot

Before deployment, print this page and gather the information needed using the **worksheet** below. Use this information to prepare ViewsFlash servlet parameters, and then deploy the application.

Step by step procedure

1. If you haven't already done so, fill out the download request form at **www.cogix.com**, and download the ViewsFlash.war at the link provided in the return e-mail. Use the java JAR utility to expland ViewsFlash.war to a temporary directory for examination or open the file with a utility such as WinZip.
2. Identify and create a data directory on the application server file system for the ViewsFlash application. The application server must have read/write/create rights to the directory. and write its full path in the data parameter of the worksheet. In a Windows server, start with the drive letter, like this: C:/cogix/data, using the exact capitalization of the directories.
3. If you have received a license key, note it. If you don't have one, it can be added later; without a license key, the application will stop running in 30 days, and pages will contain an "Evaluation Copy" message. To get an evaluation key, contact Cogix.
4. (If **using a database**) Enter in the worksheet the name that corresponds to your database. For additional information for your database and your application server, see **Database specific information**.
5. (If **using a database**) The database must be available as a named datasource in the application server. The data source should use Connection Pooling and auto-commit. It does not need to use XA Transactions. The database user associated with the datasource should, if possible, have the right to create, modify and delete tables. If this is the case, there is no need to create the ViewsFlash tables in advance. If this is not permitted, usually for security reasons, then the tables must be created before deploying ViewsFlash, using the SQL scripts provided in the appropriate directory in WEB-INF/sql in the expanded ViewsFlash.war file. Write the name of the JDBC Data Source in the datasource parameter of the worksheet. If you plan to deploy without a database now and to add the database later, see **Using a Database**.
   Different application servers use different naming conventions. The following table shows a sample JNDI data source name and its corresponding servlet parameter:

Application server	JNDI data source name	datasource= parameter
WebSphere	MyDB	MyDB
WebLogic	MyDB	MyDB
JRun	MyDB	MyDB
Tomcat 5.5, 6	jdbc/MyDB	java:comp/env/jdbc/MyDB
JBoss	MyDB	java:/MyDB

6. (If using **application security**) Review and modify the security constraint elements in WEB-INF/web.xml, especially the

security roles. These roles may need to be mapped to other roles or groups, depending on your application server. In WebSphere, ViewsFlashRespondent is usually mapped to the All Authenticated group. In Weblogic, review and modify the WEB-INF/weblogic.xml file.
Use the appsecurity=user parameter, and also enter an administrators= parameter that lists the names and/or roles of users who are the designated ViewsFlash application administrators; these users grant and delegate access rights.
(If not using **application security**) Enter an adminpw= parameter, which will be used by the ViewsFlash administrator to enter the application. Individual users will be able to create their own passwords to protect their own application areas.

7. The **Servlet Parameter Reference** section explains less commonly used parameters.
8. Having compiled the worksheet with servlet parameters, provide them to the ViewsFlash application in one of these three ways:
   a) Create a viewsflash.properties file at /etc/cogix, like **this one**, and fill in the values.
   Each line has a property name and a property value, separated by an equals sign. Lines with a # are comments.
   In a Windows server, this file must be on the same drive as the application server, at \etc\cogix\viewsflash.properties

   b) If it is not possible to use /etc/cogix, create the viewsflash.properties file in another directory and enter its location in the propertiesfile init-parameter in WEB-INF/web.xml.

   c) Do not use a viewsflash.properties file at all, and enter each servlet parameters as an init-parameter in WEB-INF/web.xml.
   Remove the propertiesfile init-parameter, and enter each parameter as an init-param element in the viewsflash servlet element.
   Example:

```
<servlet>
  <servlet-name>viewsflash</servlet-name>
  <servlet-class>com.cogix.vwf.viewsflash</servlet-class>
     <init-param>
       <param-name>data</param-name>
       <param-value>/var/cogix</param-value>
     </init-param>
     <init-param>
       <param-name>license</param-name>
       <param-value>type=unlimited;exp=...rest of the license key</param-value>
     </init-param>
... more init-param elements ...
<load-on-startup>1</load-on-startup>
</servlet>
```

9. If any changes have been made to web.xml, weblogic.xml, or portlet.xml (in portals only), make an archival copy of the changed files and repack the ViewsFlash.war file using the Java JAR utility. The archived files will be needed when upgrading ViewsFlash to new versions. For detailed instructions on how to unpack and repack the ViewsFlash.war file, see **Modifying the ViewsFlash.war file**.
10. Weblogic users only: review and modify, if appropriate, the WEB-INF\weblogic.xml file.
    Note that If servlet parameters specify that ViewsFlash is running on a database, ViewsFlash can be deployed as a war file. If they do not, ViewsFlash must be deployed as an exploded war file.
11. Deploy the ViewsFlash.war file in the application server as a Web Application. Once installation is complete, start the application in your application server.

Print this page and use the following worksheet to gather all needed information in one place. You can also use this sample **viewsflash.properties** file.

ViewsFlash servlet parameters for a typical production configuration, using a database and application security:

Parameter	Fill with	Examples	Value to use
data=	Directory for ViewsFlash data. This parameter is required.	/etc/cogix/data /var/cogix C:/etc/cogix/data	
license=	Get a license key from Cogix. Can be added after initial deployment. ViewsFlash 5 requires license keys in a new format.	license=....	
If installing with a database:			
database=	Name of the database	Oracle9 MSSQL	
datasource=	JNDI name of data source	mysource	

		java:comp/env/jdbc/mysource	
If using **application security**:			
appsecurity=	user	user	
administrators=	User ID's and roles of ViewsFlash application administrators	me,@thisrole	
If not using application security:			
adminpw=	A password to identify the application administrator.	p@ssw0rd	

Post deployment configuration and troubleshooting

1. It is very important to verify correct installation before proceeding. Check all of the following:
   The application server's administrative console should show that the application started successfully.
   Look for "ViewsFlash" in the application server logs. There should be no error messages.
   Look for a file "viewsflash.log" in the directory specified in the data parameter. There should be no error messages in this log.
   Typically, error messages in any of the above will be caused, most often, by database connectivity or permissions.

   If any errors are found, stop the ViewsFlash application, modify parameters or adjust application server settings as required, and start the ViewsFlash application again.
2. Once no errors are found in the logs, go to one of these two URLs:

   If application security is enabled with appsecurity=user, use this URL and log in:
   http://yourserver.com/ViewsFlash/servlet/vfadmin?Diagnose=1

   If application security is not enabled, use this URL, and log in with the password provided in adminpw:
   http://yourserver.com/ViewsFlash/servlet/viewsflash?Diagnose=1

   If successful, the bottom of he diagnostic display should say:

   Installation successful.
   Click here to continue with Administration and Setup.

   If there are error messages instead, correct them, restart the ViewsFlash application in the application server, and try the ?Diagnose=1 command again.
   Once no errors are shown by Diagnose=1, click on the Administration and Setup link, and fill out the entries in the Administration page that follows.

   PORTAL USERS: If you are using a portal, specially WebSphere, the ViewsFlash application URL may different. Go into the portal, install the Survey portlet, log in as a survey portlet administrator, and go to the Configure mode, and click on the Survey Setup link. This will open up a window where you can enter the Diagnose command. Bookmark that page for easy access.
3. The application is now installed.
4. Review all pages in the Default place and in the Default questionnaire and modify them if needed. For example, to make the default language direction right to left, unlock the Default questionnaire, modify the Language page, publish the Default questionnaire and lock it again.
5. If using User Security (appsecurity=user), the Administrator needs to become familiar with creating places and managing their **Access Rights**.

Next: Modifying the ViewsFlash.war file

Data retrieval query

Data Retrieval retrieves the content of completed questionnaire forms to the browser and to Excel.

A Data menu at the top of the page specifies what data should be displayed in the report:
Data: Table EXAMPLES_STORE

Choosing "Responses in progress" allows reviewing responses that have not been submitted and saved yet. Responses are not saved until they are submitted. After they are submitted, they can be retrieved using one of the other options, depending on how responses are saved. See **Save** for how to change how the data is stored.

Otherwise, a simple message acts as a reminder of where the data resides:
Data: from database table

A **Filter** menu appears next, which allows limiting the records that will be retrieved. For example, selecting a comment field and choosing must be "answered" will display all comments submitted.
Filter: recommend must be equal to this value:

"Select questions to display" lists the names of all questions saved. Clicking the All check box allows selecting and de-selecting all questions. Clicking the Show question text check box displays the complete text of each question.

For each question, choose whether to display just the question name at the top of the report or the question text, or both.
Also choose whether to display each answer's text rather than its value.

The first two check boxes indicate what to display at the top of the report: the name of each question, and the full text of the question.
For each question, display:
☑ question name ☑ question text
☐ answer text

The answer text checkbox indicates that, rather than showing the values stored in the database for each response, the text that corresponds to that response should be used in the report. For example, if a radio button has values 1,2 and 3 for Low, Medium, and High, checking this box will show the words Low, Medium or High instead of the values 1, 2 or 3.

The Report Format drop down allows three formats: Table, CSV, and XML. The CSV format displays the data in comma delimited format. The XML format displays the data in that format.

View in Excel can be used with the Table format display; the View Report link will show the data in Excel.

The header and footer regions allow defining report headers and footers. HTML can be used in these fields.

After pressing Submit, click on the View data report link to display the report in a pop-up window.

The look and feel of the Data retrieval report can be modified by creating a custom Data Style template based on the DataReport style. In this case, an additional drop-down menu will appear that will allow selecting that style.

Using the Table format report

When View in Excel is not selected, the data is displayed in the browser in tabular form. If the question name has been selected, the report can be sorted on any column by clicking on the labels in the top row:

Training session report

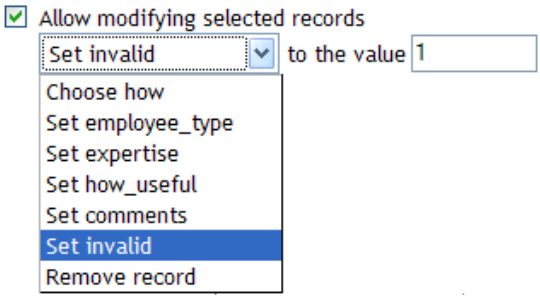
Date	Sequence	employee_type	expertise	how_useful	comments
Date entered	Record number	Employee Type	What is your level of in-depth expertise?	How useful was the training to you?	Please share any comments you have:
6/29/06	1	tech	1	5	The room was too cold.
6/29/06	16	mgr	1	3	Thanks for keeping the intro short and getting right to the material.
6/29/06	20	mgr	2	5	I have nothing special to say right now.
6/29/06	26	tech	2	5	I could have used a break after a couple of hours.
6/29/06	27	tech	2	3	Thanks for keeping the intro short and getting right to the material.

Modifying records
These features are useful for identifying outliers or malicious entries and flagging them accordingly. A Filter can then be used to remove flagged

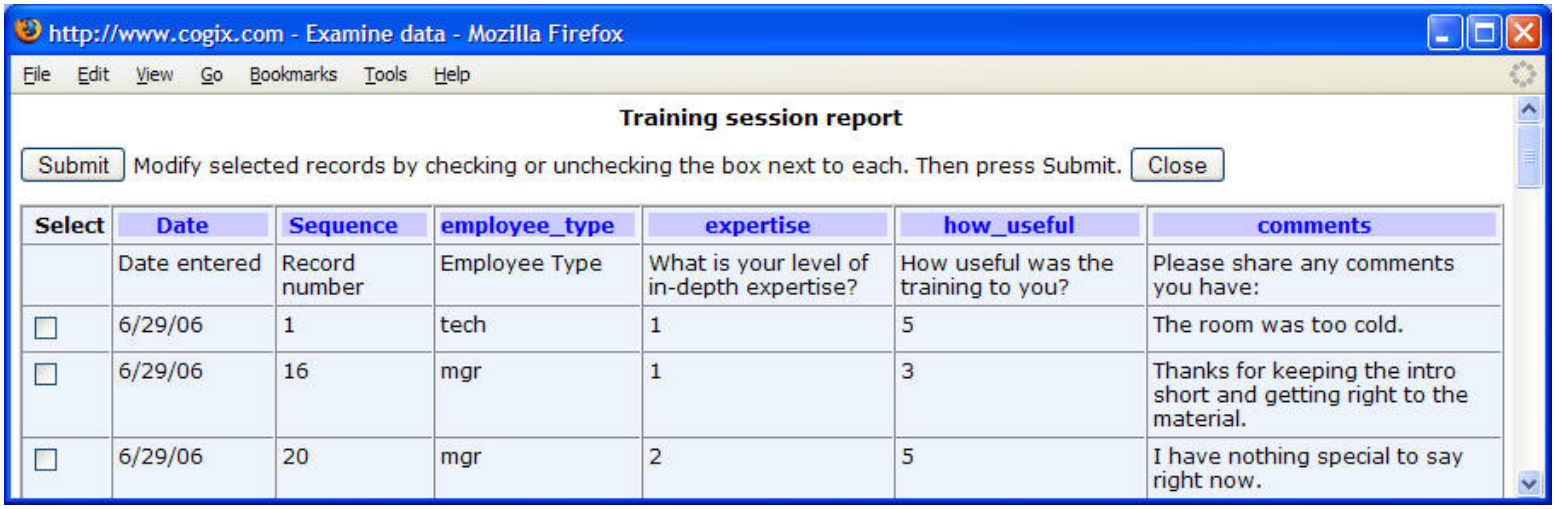
records from analysis.

When the questionnaire saves data in its own table (see **Options / Save**), and the report uses the Table format, the Allow modifying selected records option is available.

When checked, it offers the following options:



The report will include an extra column, "Select", and Submit and Close buttons:



Select	Date	Sequence	employee_type	expertise	how_useful	comments
	Date entered	Record number	Employee Type	What is your level of in-depth expertise?	How useful was the training to you?	Please share any comments you have:
☐	6/29/06	1	tech	1	5	The room was too cold.
☐	6/29/06	16	mgr	1	3	Thanks for keeping the intro short and getting right to the material.
☐	6/29/06	20	mgr	2	5	I have nothing special to say right now.

The check box in the Select column will be checked if the value of the "invalid" field is set to 1, and will not be checked otherwise. To change a record's invalid setting, check or uncheck the box and press Submit.

The "Remove record" option will delete selected records from the database altogether when Submit is pressed. It is not possible to restore them later.

Saving the report
The report can be saved permanently for later use by giving it a name and pressing Create:

To access the report, click on the **Reports** link.

Troubleshooting filters

After selecting a filter, press the Submit button at the bottom of the page. After the page is submitted, the "Show filter SQL" link, which is displayed below the Filter menu, will display the portion of the SQL WHERE clause that will be used to enforce the filter. This can be useful when a filter is not producing the desired result.

If a data analysis report produces unreasonable results, it's a good idea to retrieve the actual data using the filter and without using a filter.

Next: Using filters

Using Filters

Data, Univariate and Cross tabulation reports can be filtered when their data is stored in a database.

When data is stored in more than one format, a Data menu appears at the top of these pages:
Data: Table EXAMPLES_STORE
Change the choice to analyze data from different storage formats. See **Save** page for how to change where the data is stored.

Otherwise, a simple message acts as a reminder of where the data resides:
Data: from database table

The Filter menu appears next. Use the pull down menus below to see more of the options.
Filter: recommend must be equal to this value:

The "Filter" dropdown lists all the fields in the form, as well as the date when the questionnaire was submitted, the user ID if it was saved, and the sequence number of the record. The "must be" dropdown specifies the relationship to the value entered in the "this value" field. Note the following recommendations:

The "answered" option selects only those records where the respondent answered the selected question

The "not answered" option selects only those records where the respondent didn't answer the question

The "answered" and "not answered options" can be used with any field, and they are specially useful with text and comment fields.

When Date is chosen, an explanation is shown: Date format: M/d/yy which means that the date must be entered in that format, eg 3/31/06. Pressing on the Change link switches to **My ViewsFlash**, where the format of entered dates can be changed.

When a checkbox question is chosen, the options in the must be menu become "any of", "answered" and "not answered". Selecting "answered" will include records where any of the checkboxes was checked. Selecting "not answered" will include records where all the check boxes were left unchecked. Selecting "any of" will include records where any of the values listed in "this value" is checked, separated by commas. For example, if the values stored are "vanilla","strawberry","chocolate" and "mint", entering "mint,vanilla" (without the quotes) will select only those records.

When selecting "must be between", an extra field opens up to select a range of records. This is useful for selecting records between a range of dates, for example.

Troubleshooting filters

After selecting a filter, press the Submit button at the bottom of the page. After the page is submitted, the "Show filter SQL" link, which is displayed below the Filter menu, will display the portion of the SQL WHERE clause that will be used to enforce the filter. This can be useful when a filter is not producing the desired result.

If a report produces unreasonable results, it's a good idea to retrieve the actual data using **Data** retrieval, both with the same filter and with no filter.

Next: Univariate Analysis

Univariate Analysis

Univariate Analysis provides a convenient way to produce the most useful statistics about some or all saved questions.

For numeric fields, which contain only numeric values or missing data, ViewsFlash can calculate measures such as mean, standard deviation, median, mode, high and low. Simple frequency distributions are available as well.

The Univariate Analysis query page

When data is stored in more than one format, a Data menu appears at the top of these pages:

Data: Table EXAMPLES_STORE

Change the choice to analyze data from different storage formats. See **Save** page for how to change where the data is stored.

Otherwise, a simple message acts as a reminder of where the data resides:
Data: from database table

A **Filter** menu appears next, which allows limiting the records that will be retrieved. For example, selecting a comment field and choosing must be "answered" will display all comments submitted.

Filter: recommend must be equal to this value:

Select questions to display. Check the names of all the questions you want to tabulate. To view the full text of the questions, check the Show Question Text box.

Questions created with the ReportText style are displayed in this list. These questions can only be edited using the List Editor, as they are invisible in the Visual editor and in respondent questionnaires. The matrix header questions in Likert and Semantic Differential Scales, when opened, include a field:

☐ Use as report text

This field needs to be checked for the question to be visible in this list. By selecting it, the explanatory text that normally appears above the matrix is shown in the report.

To see other questions that are not listed, check the Show all questions box; questions not shown are usually unsuitable for analysis, such as text fields not selected for tallying in the Options / Tally page.

Select the options in the "For each question, show" section as needed. The graph option enables **charts**.

The View report in Excel option will retrieve the data and display it in Excel.

The header and footer regions allow defining report headers and footers. HTML can be used in these fields.

After pressing Submit, click on the View Univariate report link under the Submit button. The report will display in a pop-up window or in Excel.

From the pop-up results window, use the browser's File Print or File Save As commands to print or save the report.

The Edit/Select All and Edit/Copy commands can be used in the results window to paste the resulting tables into programs such as Word and Excel.

The look and feel of the Univariate Analysis report can be modified by creating a custom Univariate Style template based on the UnivariateReport style. In this case, an additional drop-down menu will appear that will allow selecting that style.

Saving the report
The report can be saved permanently for later use by giving it a name and pressing Create:

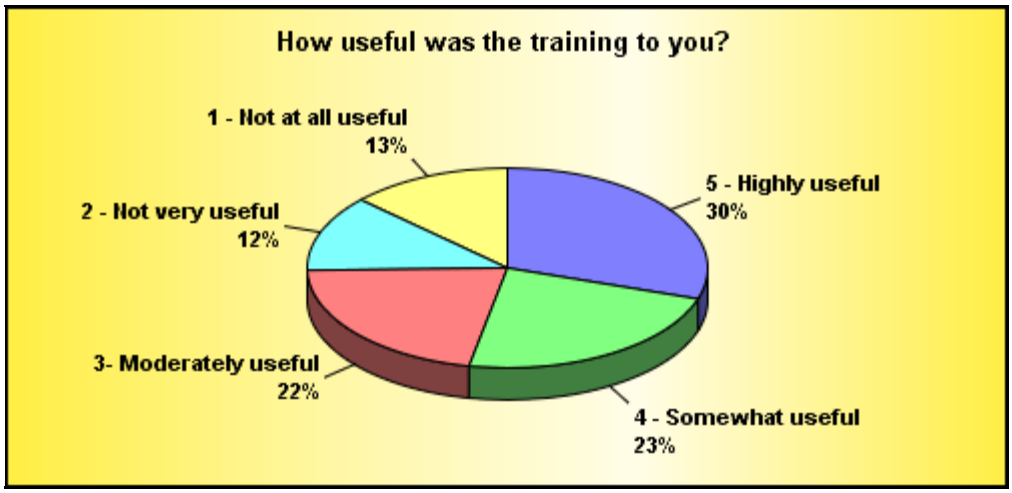
To create a permanent report with these settings,
enter a report name: and press Create

To access the report, click on the **Reports** link.

Next: Charts

Using Charts

Univariate and Cross tabulation reports can now include charts. To produce the charts, simply click the "graph" option on that page.
When the report is viewed, it will include a graph. For example:



The chart can be edited and downloaded to the browser, Excel and PowerPoint. Clicking on the graph produces a page with the following options.

The top of the page includes four links:

- Printable chart - displays the chart by itself in the browser, where it can be printed using File / Print.
- View in PowerPoint - downloads the chart into PowerPoint, where it can be resized and used in a presentation.
- View in Excel - downloads the chart into Excel, where it can be used in a spreadsheet.
- Return to report - closes the page and redisplays the original report

Below the graph, four links open up sections that allow changing the graph appearance:

- Style - allows changing the graph type from a pie chart to a bar chart (for univariate reports) and between a side by side bar chart and a 3D stacked cubes chart (for cross tabulations)
- Text - allows changing the chart legends and titles
- Font - allows changing the chart font, size, and color
- Color - allows changing the chart background, whether the background is flat or a color gradient, the colors of the bars or pie elements, and whether the bars are solid colors or transparent

At the bottom of the page, three buttons determine what to do next:

- View - redraws the chart after style, text, font or color settings are changed
- Save - saves the chart settings for the next time the chart is drawn
- Set default - saves the chart settings so that they become the default for other univariate or cross tabulation charts in this survey

System Administrator note: About fonts in charts

The fonts that are available depend on the fonts that are installed in the system, in the application server's Java VM.
When using international character sets such as Chinese, an appropriate font must be installed. Some fonts to consider are: Microsoft Arial Unicode MS and Cyberbit.

Default chart options can be set by adding the following to viewsflash.properties:
chart.fontstyle=Arial Unicode MS
chart.fontsize=10
chart.fontcolor=0
chart.colors=8080FF,80FF80,FF8080,80FFFF,FFFF80,FF80FF
chart.bgcolor=0x00FFEE44

chart.3d=true

Next: Cross Tabulation

Cross Tabulation query

ViewsFlash provides the data analyst with an interactive querying tool for discovering correlations between variables in the collected data. To get the best results, please read this section first.

Preparation

Create a survey in the usual manner. On the Save page, make sure that you are Saving all the questions you may want to tabulate later.

There's no need to set up special values (such as the numbers 1,2,3) for specific answers. You can specify the descriptive values 'male' and 'female' as answers to the question Sex, rather than 1 and 2. There are, however, two circumstances that bear special consideration.

Numeric values and Numerical Measurements. For a question such as "Rate your satisfaction with customer service", it is best to assign numerical values such as 1,2,3,4 to the visible text of bad, ok, good and great, for example. This allows calculating both cross tabulations by each value as well as Means, Standard Deviations and Medians.

Two-valued answers and t-test scores. If a question has two answers, a t-test can be calculated. It's easiest to set up these questions as two radio buttons.

Multiple responses. For a question such as "What are your favorite sports", where multiple answers (baseball, football, basketball, etc) are possible, use one question with multiple checkbox answers. ViewsFlash will analyze this data properly. You could also use a different question with a single checkbox answer or a yes/no radio button answer. This will work too, but it will be more difficult to visualize the effect of one question on all the choices at once.

Missing data. You have the option of counting responses in a crosstabulation when one or more of the questions have not been answered, in the "Tabulate No Answer in a dependent variable" option. You may want to use the option in the Question page to require people to answer particularly important questions.

Others. Tabulations can include values that are not explicitly defined as question values. These "Others" include questions that explicitly allow "Others", text questions, hidden questions, and calculated scores stored in a hidden option. All these can be tabulated, **provided** that the Tally page is set to calculate a Live Tally on the chosen field when the survey begins. When tabulating Others, you can choose to tabulate them in a single category or separately as individual values. Note again that count Others as individual values, the question must be selected on the Tally page.

Data Editing. You have the option to save response data in a .txt file or a relational database. In either case, you can edit the data by hand and run your crosstabulations again. This allows for complex offline response editing, for example.

Using the Cross Tabulation Query page

When data is stored in more than one format, a Data menu appears at the top of these pages:

Data: Table EXAMPLES_STORE

Change the choice to analyze data from different storage formats. See **Save** page for how to change where the data is stored.

Otherwise, a simple message acts as a reminder of where the data resides:
Data: from database table

A **Filter** menu appears next, which allows limiting the records that will be retrieved. For example, selecting a comment field and choosing must be "answered" will display all comments submitted.

Filter: recommend must be equal to this value:

Select the questions to analyze by choosing the independent and dependent variables. To add a third (control) variable, simply choose it from the third column. To view the full text of the questions, check the Show Question Text box. To see questions that are not listed, check the Show all questions box; questions not shown are usually unsuitable for analysis, such as text or hidden questions not selected for tallying in the Options / Tally page.

Select the options in the "Show" section as needed. The option "in separate tables" separates the display of counts and percentages into separate tables. This can be very handy for tables that are copied to reports.

The "graph" option displays the relationships with a bar chart. The "shade cells by percentage" colors the cells in the percentage chart with a varying degree of color that allows easy visual spotting of relationships.

Checking the Chi-square box calculates and displays this significance statistic and the probability P that the results are due to

random variation. A value of p < .10 means that there's a 10% probability that the relationship observed could be due to chance.

Checking the t-test box calculates and displays this significance statistic at the desired level of confidence. Note that the independent variable must have exactly two values and the dependent variable must have only numeric values.

The View report in Excel option will retrieve the data and display it in Excel.

The header and footer regions allow defining report headers and footers. The HTML editor can be used in these fields.

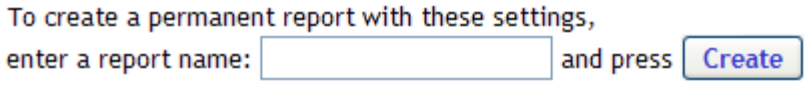
After pressing Submit, click on the View Cross tabulation report link under the Submit button. The report will display in a pop-up window or in Excel.

Cross tabulation results are displayed in a pop-up window. From that window, you can use the browser File Print or File Save As to print or save the report.

The Edit/Select All and Edit/Copy commands can be used in the results window to paste the resulting tables into programs such as Word and Excel.

The look and feel of the Cross Tabulation report can be modified by creating a custom Xtab style based on the XtabReport style. In this case, an additional drop-down menu will appear that will allow selecting that style.

Saving the report
The report can be saved permanently for later use by giving it a name and pressing Create:

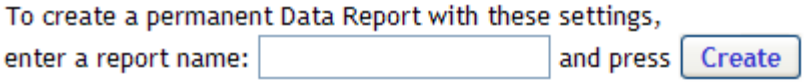


To access the report, click on the **Reports** link.

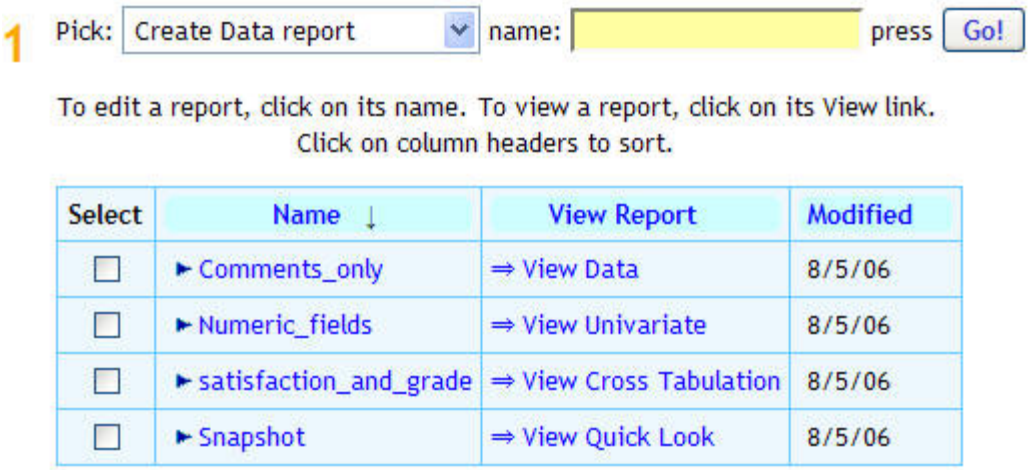
Next: Reports

Reports

The Data, Univariate, and Cross tabulation pages can view and analyze data in a variety of ways. Each of those reports includes at the bottom a section for saving the report settings:



These reports are then listed in the Reports page:



Clicking on the report Name allows editing the report.
Clicking on the View Report link opens that report in a pop-up window.

A report can be created from this page as well by picking the report to create, giving the report a name and pressing Go.

In addition, Create a Quick Look report allows creating a report that tallies questions and emails the report periodically.

The Quick Look report

Sections 1 through 3 provide options for a report header, footer, and selecting the questions to be included in the report, and a Response style to use.

Section 4 specifies when to run the report:

Check "Now" to do so at this time. Check "Every" to run the report at the frequency (every day, specific day of week, hour, and minute) and time (in the ViewsFlash server time zone) indicated by the fields that follow. At the appointed time, ViewsFlash will create a web page with the latest poll results.

Check "Keep numbered copies" to create a differently numbered HTML file each time the report is created. The web pages will have names ending in 1.html, 2.html, etc..

Check "E-mail the report to", and enter one or more e-mail addresses, separated by commas, to have the report e-mailed to the recipients as an HTML attachment when it is generated. For example, you can receive a daily survey report in your mail box.

After pressing Submit, use the links under the Submit button to preview and view the report live.

Next: Security

Security and Authentication

Introduction

Questionnaires, when more than one page long, require an ID for identification and authentication as the person filling it out moves from page to page. In some cases, such as a retirement plan survey, it is necessary to know the identity of the person who is filling out the questionnaire. In other cases, the questionnaire is not about the person who is filling it out, but rather about some event, such as a Help Desk request form. identified by a form number. Finally, there are times when no ID is required at all, such as anonymous popularity polls for entertainment.

Polls, also known as single page forms as designated on the General page, are the only kind of questionnaires that do not require authentication. All others require selecting one of the methods presented in this section.

Regardless of the authentication method used, anonymity can be preserved by choosing, in the Save page, to not record the ID of the person completing a questionnaire in the data set.

1 Questionnaire Authentication

Part A, Specify what method to use to identify each questionnaire. The methods are categorized by where the questionnaire ID comes from.

Identifying a questionnaire with the respondent's Login or user ID.

When a questionnaire should be identified by the user's ID, choose one of these methods. All of them require the user to be logged in to the corporate network with their usual credentials, and all of them will user their User ID to identify the survey. Note, however, that anonymity can be preserved by choosing to NOT save the questionnaire ID in the questionnaire's **Save** page.

Basic. When this option is chosen, the questionnaire is available at a special URL, namely http://..../servlet/vfauth?... This special URL is protected by the application server, which requires the respondent to authenticate.
(see **Application Security**). This option is used most often in intranets, allowing for single sign on.

Technically, this option retrieves the user ID using the getRemoteUser() method and is compatible with any authentication method that supports this standard J2EE protocol. See also **Extensible Authentication** for other authentication techniques supported.

Portal. This method is used in questionnaires that are presented in a Portal by the Cogix Survey portlet. The user ID of the currently logged is used.
If the user is not logged in, a unique random user ID is used and the respondent is anonymous because on subsequent visits the respondent has a new user ID, so preventing duplicate responses requires that the portlet be in a page that requires logging in.

IIS Windows Integrated Authentication. The user ID provided by IIS is used. This method expects an AUTH_TYPE header of NTLM, Negotiate, or Kerberos, and the user id in REMOTE_USER. This is how IIS normally operates.

Role. This method is identical to Basic, above, and in addition to being authenticated, the visitor must be in the designated J2EE role.

Identifying a questionnaire with a predetermined value in the request's environment

ViewsFlash is often deployed as part of a larger application, which is responsible for managing the IDs of its questionnaires. In a complex form application, the larger application may generate and manage Form IDs; in a heavily trafficked web site, the web server may create tokens that enforce third-party authentication and make them available to ViewsFlash. The common

element is that ViewsFlash must look in its environment for information that identifies the questionnaire.

Cookie. This method extracts the ID from a cookie. Many custom authentication systems and some single sign-on systems record an authenticated ID in a cookie. A prefix and suffix that surround the ID can be specified to parse the cookie.

If the cookie name is left blank, and the Place where the questionnaire is located specifies concurrent surveys, a cookie named ViewsFlash will be used, and if that cookie is not present, a cookie will be sent to the browser with a unique random number that will identify that user's computer for all surveys. If the cookie name is left blank and the place specifies one survey at a time, a cookie with the name of the place, expiring when the current poll closes, will be used instead.

Session Attribute. When choosing this option, if the name is blank, the Session ID will identify each visitor while they complete a multi-page questionnaire. The Session ID is a random number created at random that identifies the visitor while they are sitting at their computer; it does not contain any traceable personal information, so this option allows visitors to participate anonymously. Note that questionnaires with too many items on a page that take a long time to complete are at risk of experiencing session timeouts and loss of data.

If an attribute name is used, the ID found there. Technically, this value must be set by another application using HttpSession.setAttribute (Name,Value) to store a user ID.

Entering the value CAPTCHA allows using this method with the ViewsFlash/start.jsp "Captcha" authentication form. Instead of directing respondents to /ViewsFlash/servlet/viewsflash?cmd=page&pollid=SSS!PPP, direct them to /ViewsFlash/start.jsp?cmd=page&pollid=SSS!PPP. A captcha login form will be displayed, asking them to enter the letters in the graphic. This is an excellent method of allowing anonymous surveys while preventing against robot attacks. The start.jsp page can be copied and customized as desired, and includes the source code for adapting the graphics if necessary.

Header. This method extracts the UserID from the designated HTTP header. The web server is responsible for setting the header.

Identifying a questionnaire with a value passed as a query string or a login form from an invite list

Question. The ID is passed as a field in the questionnaire's form. The default option, "Use a generic field", assigns random, encrypted ID numbers using to the questionnaires using a hidden field in the form.

If a specific question is selected, the questionnaire must pre-populate that question using the ViewsFlash questionnaire's query string. For example, if a question named "formid" is used, the URL is:
/ViewsFlash/servlet/viewsflash?cmd=page&pollid=xx!xx&formid=334433
The question is usually created as a ViewsFlash question of type hidden.

When a specific question is selected, ViewsFlash can assign numbers to the questionnaires by setting the prepopulated value to $ or $*. For example:
formid=$ assigns a sequential number, starting with 1.
formid=$* assigns random encrypted IDs.

Form. This method requires using an Invite List. The visitor is presented with a login form to enter an ID and an optional password. These are validated against entries in the **Invite List** currently in use to **invite** people to a survey.
To customize the default login form, make a copy of ViewsFlash/surveylogin.html in the ViewsFlash directory, modify it, and enter then name of the modified file in the "Use login form" field, such as mysurveylogin.html.

None. This method can only be used with polls and single page questionnaires where, because they are only one page long, an ID is not needed to move from one page to the next.

What method is best?

If none of the authentication methods that refer to a specific ID are appropriate, use "Question" with a generic field. This method generates, automatically, sequential ID numbers for each questionnaire, and passes the IDs from page to page in a hidden form field. It is the easiest and most reliable method for authenticating questionnaires when no better choice is available. The Question method, with a specific ID field, gives a little bit more control over the IDs, especially for creating URLs, because the ID is stored in a field that can be referred to like any other field.

For applications where the questionnaire is essentially a form, rather than a set of questions about the respondent, question authentication is also recommended.

For applications where ViewsFlash is being used as a component in a larger system, use a Session Attribute and have the host system set that attribute before displaying the VIewsFlash questionnaires.

Other authentication tests

These options are available always regardless of the authentication method selected above.

Questionnaire must use SSL. If this box is checked, the survey URL will begin with https://, and all data entered in the survey will be protected using Secure Sockets Layer technology. If a visitor attempts to take this survey at the same URL without https://, they will be considered not authenticated. To use a non-standard port for https, use the **securesurveyport** servlet parameter.

Referer header must not be blank. If this box is checked, the "referer" HTTP header must be non-blank, or else the questionnaire will be considered not authenticated. If this option is not checked, a blank HTTP header will always be authenticated regardless of the setting in the "referer must contain" option below.

Referer header must contain. If this box is checked, the "referer" HTTP header will be checked to make sure that it contains the exact string provided. If it doesn't, the visitor will be considered not authenticated.

When using invite lists, allow IDs that are not on the list. When using an e-mail invite list, questionnaires are authenticated usign the method selected above. If a questionnaire's ID is not on the invitee list, it is not rejected as not authenticated. Checking this box allows questionnaires that pass authentication when their IDs are not on the invitee list.

Part B, When an attempt to participate is made without proper authentication. These radio buttons specify what to do when authentication fails for any reason. The respondent can be redirected to a specific page, or a standard response page can be constructed. If neither one is specified, the respondent will receive an empty page, usually displayed as an error by the browser. A default response page is provided: viewsflash/unauthorized.html, which will explain why authentication failed. If you create your own, start with a copy of this page to report more clearly what caused the authentication failure. This situation will most likely arise only while testing, since the web server and/or application server, when properly configured, will prevent an unauthenticated visitor from even seeing the questionnaire.

2 How to handle multiple entries for the same questionnaire

Part A, Choose what is allowed.

For multiple page questionnaires, the following two options are shown:

- ☐ Allow one response only, which cannot be edited after completing it.
- ☐ Allow one response only, and allow editing it after completing it.

Leave both boxes unchecked to allow multiple responses with the same ID.

When "Allow one response only, which cannot be edited after completing it." is checked, ViewsFlash will authenticate the user according to the selected method, and use that unique User ID to detect attempts at a duplicate responses.

In some applications, such as entering a Help request form, it is appropriate to have the same user submit multiple instances of the form. Leaving both of these selections unchecked is appropriate for that case.

In others, such as surveys, it is best to have the respondent complete the form only once. Checking the first of these selections accomplishes that.

In others, such as submitting a request that can be revised after the form data is reviewed, check the second of these selections. When the respondent returns to the questionnaire with the same ID, it will be presented with all data already entered filled in, and the respondent can make changes to the data previously entered.

In voting and survey applications, it is usually to ascertain that only authorized users can participate. Using a single sign-on system and an Invite List together provides this certainty.

For polls and single page questionnaires, as selected in the General page, the following additional options are shown:

- ☐ Treat repeated submissions from the same IP address as duplicates, with the following exceptions:
  - After 300000 milliseconds, accept submissions from the same IP address.
  - ☐ Treat different Browser ID strings as different submissions
  - ☐ Throw a cookie

When "Treat repeated voting from the same IP address as duplicates" is checked, ViewsFlash will check the IP address of the visitor's computer against the IP addresses of all visitors who have submitted entries in this poll in the indicated time, and reject the response as above if the IP address is already recorded. However, because many ISP's reuse IP addresses, particularly for dial-up connections, it's important to try to detect when different people are using the same IP address. This is done by checking the "Treat different Browser ID strings as different submissions" box, and by entering a time limitation on the check. Changing these values will result in more strict checking. For example, on an intranet, there is no reuse of IP

addresses usually, and it would be best to leave the time limit blank and remove the check from the different browsers checkbox.

When "Throw a cookie" is checked, the ID selected by the authentication method, if any, is ignored, and instead, a cookie is maintained in the participant's browser, with a history of the questionnaires the respondent has filled out, and is used to detect an attempt to fill out the form repeatedly. This method is not as good as using the single sign on authentication methods and is recommended only as a last resort.

The combination of both of the above methods, used simultaneously with the default values, assures the best protection possible against accidental or deliberate duplicate responses when authentication is not available.

Revising Entries
When "Allow one response only, and allow editing it after completing it" is checked, ViewsFlash allows a respondent to fill out the questionnaire normally the first time. If a respondent returns to the questionnaire again with the same ID, the respondent will be allowed to revise the entry. This is useful when the data entered is a user preferences profile or a services request form, for example. This feature requires running on a database and using a multiple-page questionnaire. If these conditions are not met, this option will not be displayed. Note that live tallies count only the first entry, but analytical tools such as Univariate and Crosstabulation analysis count the modified entries.

Part B, acting against duplicate attempts

When an attempt at a duplicate response is detected, you can choose to respond as if the response was not a duplicate, displaying poll results just like always. This method has the advantage that accidental, innocent duplicate responses will not disturb the visitor experience, while at the same time someone who is intentionally trying to influence results by responding repeatedly will probably think that they are being successful and will give up reasonably quickly. You can also choose to display a message, such as "You have responded already," or even take visitors to an entirely different page.

The four radio buttons specify what to do when a duplicate submission attempt is detected. You can choose to respond normally, giving no indication that the attempt was detected as a duplicate and was ignored.

If you provide text to insert, such as "You have responded already," it will take the place of the [/statusmg] tag in the response template.

If you specify a page to display, it can be a plain html page or a results template. The reference can be: a file name without a leading / and therefore relative to the ViewsFlash web server root, an absolute filename such as /dir/dir/filename, or an absolute URL (http://) reference to a page on another web server.

If you specify a page to redirect to, the visitor's browser will go there. If you enter *, the browser will return to the same page (the 'referer'). If you enter *?x=y, ?x=y will be appended to the URL of the referring page. This can be very useful with the Web Services API.

3 Saved Entries. In multi-page surveys, when the "Save" button is enabled, you can specify what will happen. You can redirect the visitor to a specific URL. When you redirect, if you specify a page to redirect to, the visitor's browser will go there. If you enter *, the browser will return to the same page (the 'referer'). If you enter *?x=y, ?x=y will be appended to the URL of the referring page. This can be very useful with the Web Services API. You can also use a specific response page by entering its full path in the file system, beginning with /. Additional information available in **Multi-Page Surveys**.

4 Incorrect entries. When a question requires an answer, as checked in the Define Question page, the submitted entry will be checked to make sure that all questions that must be answered have been answered by the visitor. If any of the questions have not, the three choices shown determine what to do with the entry. Note that a rejected entry will not count as a response for the purposes of preventing duplicate responses.

The first choice is to just ***tally any data that is available***. (Note that not all four choices are always available).

The second choice specifies a page to redirect to. If you enter *, the browser will return to the same page (the 'referer'). If you enter *?x=y, ?x=y will be appended to the URL of the referring page. This can be very useful with the Web Services API.

The third choice is to ***ignore the answers and to display the results***, using a Response Template page, into which the names of the unanswered questions are inserted, as in: "You have not answered the following questions: xxxx, xxxx, xxxx. Please answer them first." The viewsflash/vfincomplete.html template can be used or modified for this purpose. This page must be formatted in the same way as a Response Style Template, and must include the [/pendingerror] tag, which will be replaced by a listing of the names of the missing questions, separated by commas.

The fourth choice ***redisplays the poll or survey*** form, with the data entered by the visitor filling the form fields. An additional message is included to remind the visitor that certain fields are required, and a marker next to each unfilled

question is displayed. This allows the form to be held up until the visitor completes all the required fields. The visitor only has to answer the questions that remain to be answered. It requires a Poll Template page; leave the field blank to use the place's template. On that template, it uses the [/statusmsg] to display the error text, [/dataerror] to display the error pointer, and [/originanswer] to display visitor's original answers. (See **Style Template Tags**). The default values for these cause "Please answer the questions marked ***X***" to appear at the top of the survey form and ***X*** to appear to the left of each unanswered question.

5 **Late submissions.** It is possible that a visitor's survey form remains open in their browser beyond the survey closing time. This section specifies how to respond. You can choose to display the normal response page and add a message. You can also display results using a specific template.

You can also redirect the browser to a specific URL. If you specify a page to redirect to, the visitor's browser will go there. If you enter *, the browser will return to the same page (the 'referer'). If you enter *?x=y, ?x=y will be appended to the URL of the referring page. This can be very useful with the Web Services API.

6 **Data review and modification.** NOTE: this option requires that the **Data Review and Modification API** be enabled by the System Administrator. In multiple page surveys, these settings enable reviewing a survey's data after it has been completed, as well as modifying that data, using **API commands**. When the data is saved in its own table using the settings on the Save page, it can be modified using these options. When the data is saved only in the Normalized format, it cannot be modified and the Modify options are not available.

The options mean: "No one", the default, disables this capability. "The visitor, right after completing it" allows redirecting the visitor to a review page immediately after completing the survey (while their browser session is still active). "The visitor, any time after completing it" allows letting the visitor review or update his results any time after completing the survey. These options only allow the user to review or modify the survey he completed. However, when using **User Security**, the "Administrator with access rights" option allows ViewsFlash administrators with the Examine Data access right to review or modify anyone's data.

Next: Cookie Flavors

Cookie flavors

There are three ways to specify what cookies ViewsFlash should use for detecting duplicates. By simply selecting Use Cookies in the Security page, a different cookie will be thrown for each different poll. The cookie is named VFpollid, where pollid is the poll's id. While convenient and simple, this can lead to cookie proliferation very quickly.

All ViewsFlash cookies have the shortest meaningful expiration date, depending on the closing dates of the polls they cover. They normally take on the domain of the server they are issued from, unless you use the cookie domain parameter in the Administration page. This is only recommended as a last resort.

Using the history parameter in the poll page style template allows you to specify a single cookie for multiple polls. There are two flavors of the history parameter, one centered around a place and one centered on a simple voting history.

place centered cookies use a history parameter like this:
history=1,cookiename,[/spotname],redirectinfo

The "redirectinfo" is optional, and when used it instructs ViewsFlash to redirect to that page after a successful vote.

The cookie thrown has the form:
|spotname:date|spotname:date|spotname:date|

The dates in the cookie are the next date when a visitor can vote in a particular spot (shorthand for place). This cookie format is useful when many departments of a site want to share a cookie. The cookie is never allowed to get larger than 256 bytes; earlier polls are removed while new ones are added at the end.

It is important that every poll used with this type of cookie always have an ending date, because when a poll is voted on which doesn't have one, the ending date is set to six months into the future, which could prevent a visitor from voting again for a very long time.

You can use separate cookies for each poll with this option, by using history=1,PFX[/pollname],[/spotname],redirectinfo
This will throw cookies named PFXmypoll, PFXanotherpoll, etc.. PFX is an arbitrary string.

Poll centered cookies use a history parameter like this:
history=2,cookiename,redirectinfo

The cookie thrown has the form:
|pollid|pollid|pollid|

This is simply a list of the pollid's the visitor has voted in. When using this form, it would be a good idea to use a cookie for all the polls in a particular section of a site, to prevent cookie truncation. The cookie is never allowed to get larger than 256 bytes; earlier poll id's are removed while new ones are added at the end.

Next: Internationalization

Internationalization

See also **Localization** for how to configure ViewsFlash's default language to a non-English language, such as Chinese, French, Arabic, etc..

ViewsFlash allows questionnaires and polls to be created and filled out in any language in the world. Internally, ViewsFlash uses Unicode. In the browser, it uses UTF-8 by default, unless you specify otherwise in the Language page. If saving data as .txt files in the file system, it uses that character set. If saving to a database, the database converts Unicode to its internal representation. See **Technical Specification** for additional information.

Creating questionnaires in left-to-right languages
By default, questionnaires are created using the standard UTF-8 character set, displayed from left to right.

Right to left languages
- Create a questionnaire and use the Language page to set the right-to-left direction.
- The Language page also includes settings for setting the "lang=" tag and the "dir=" HTML tags, which specify the questionnaire's language and left to right or right to left direction.

Creating a questionnaire in a specific language
To create a questionnaire in a specific language other than the default ViewsFlash language, create it by adding "_ _ x x" to the original questionnaire's name.
For example, to create a questionnaire "Atitud" in Spanish only, name it "Atitud__es" (two underscorees, 'e','s'). The URL of the questionnaire will be the URL shown on the Publish page, as usual.
All the questionnaire validation messages, such as "answered required" will be shown in Spanish.

Translating a questionnaire into multiple languages
Create a multi-page questionnaire in its original language and name it as usual, with a name such as "Attitude". Publish and test the questionnaire. When satisfied, make a copy of the questionnaire for each language that the questionnaire will be translated to, using a special questionnaire name by adding "_ _ x x" to the original questionnaire's name. For example, if the original questionnaire is named Attitude, name the Spanish questionnaire Attitude__es (two underscores, 'e', 's'), and the French questionnaire Attitude__fr, etc. Then, translate the Attitude__es questionnaire using the Visual editor, and Publish the questionnaire. Each questionnaire will have its own URL; however, when the Submit button on the final page is pressed and the questionnaire is committed, the saved data and tallies will be stored in the underlying questionnaire, Attitude, rather than Attitude__es.

The codes used are ISO 639-1Language Name codes as used by Java Locale. Reference:
http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt

To analyze responses by language, create a question of type Hidden, and click on "Edit answers" at the bottom of the page to allow entering specific values. Enter the values with the language code of each language being used, and mark the language of each questionnaire as the default value by adding an asterisk, like this:
en
fr*
es
Questionnaires in the other languages modify this default value to indicate their language.

Validation messages and button legends such as "Please fill out this required field" and "Next", are internationalized using the lookupi18n tags, which are used throughout the ViewsFlash styles. These values, by default, are retrieved from a series of properties files described in **Localization**. These can also be translated using the **Messages** page without having to modify the properties files.

Directing respondents to the questionnaire in their language

Once the questionnaires in multiple languages are created, if there is a need to direct respondents to a single URL where they will choose the appropriate language for the questionnaire, follow this procedure.

Go to the Publish page for the original questionnaire, whose name does not end in __xx. Take the questionnaire URL and modify as described here. Example: starting with,
http://myserver.com/ViewsFlash/servlet/viewsflash?cmd=page&pollid=MyQuestionnaires!Attitude

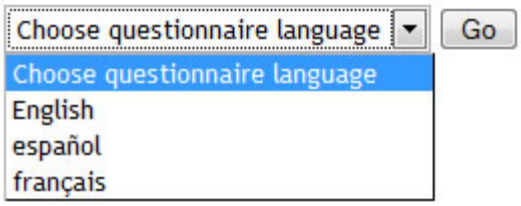
Insert the following between ViewsFlash/ and servlet :

i18n.jsp?i18n-url=

so that the result is

http://myserver.com/ViewsFlash/i18n.jsp?i18n-url=servlet/viewsflash?cmd=page&pollid=MyQuestionnaires!Attitude

When participants go to that URL, they will see:



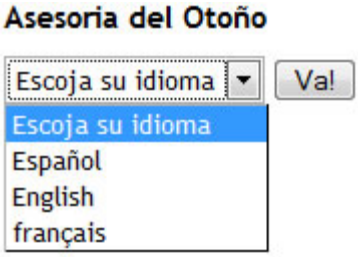
After the participant selects their language and presses Go, the browser redirects to the questionnaire in the chosen language. Only those languages for which a questionnaire is available are displayed in the menu, and the languages are displayed in their native name and character set. Chinese displays 中文, Arabic displays and so on.
If only one language is available, this page redirects to the questionnaire immediately, without presenting the language choices.
If any required parameters are incorrect, or the pollid of the questionnaires is incorrect, this page does not display anything.

Additional parameters can be inserted in the URL, like these:
i18n-submit=Va!&i18n-title=Asesoria del Otoño&i18n-choose=Escoja su idioma&i18n-language=Español&

so that the result is

http://myserver.com/ViewsFlash/i18n.jsp?i18n-submit=Va!&i18n-title=Asesoria del Otoño&i18n-choose=Escoja su idioma&i18n-language=Español&i18n-url=servlet/viewsflash?cmd=page&pollid=MyQuestionnaires!Attitude

This will display a page appropriate to a questionnaire where the most common language is Spanish, and looks like this:



The additional parameters are:
i18n-title is the title. Default is no title.
i18n-choose is the text for the dropdown menu showing the languages that the questionnaire is in. Default is "Choose questionnaire language".
i18n-language is the original questionnaire's language. Default is the language of the server's Locale, such as English. It is the first choice in the drop-down menu.
i18n-submit is the text to use in the button. Default is "Go".
i18n-url is everything to the right of the original ViewsFlash/ . This parameter is required.

All the URLs used in the Publish page are allowed. This example uses https, a port, a custom web application name instead of ViewsFlash,
and J2EE authentication (through servlet/vfauth):

https://myserver.com:88/MyWebAppName/i18n.jsp?url=servlet/vfauth?cmd=page&pollid=MyQuestionnaires!Attitude

Database and servlet parameters

An appropriate database encoding must be chosen. We recommend using UTF-8, which supports all languages.

Oracle setup: The default setting for Oracle 8.1.7 is "WE8ISO8859P1", which corresponds to ISO-8859-1. For other languages, create the database used by ViewsFlash with UTF8 encoding and add the following line to viewsflash.properties:
dbcharwidthmultiplier=3.
If a database has not been created with UTF8 encoding, you must export it and import it -- there is no way to convert one on the fly. For this reason, if there's any possibility of questionnaires being created in languages other than ISO-8859-1 in the future, create the database with UTF8 at the outset. Other applications that access questionnaire data stored by ViewsFlash will work fine if they are Unicode compatible. To find out what encoding your database uses in Oracle, use:
select parameter, value from nls_database_parameters where parameter=NLS_CHARACTERSET';

Technical Specification

Internally, ViewsFlash always processes all text as Unicode. On the browser, it uses UTF-8 as its default encoding. Very old application servers may require specifying a more specific encoding; in that case, specify it using the Language page.

Browsers
To conduct questionnaires in languages other than English use **IE5, Firefox 1, Opera 6, or Netscape 6, Safari 1.2 or higher.** The browser must have the appropriate fonts and language packs installed to be able to view a page in the desired encoding.

Each browser must also have a way to enter international characters, such as a language-specific keyboard, a "compose" key, or Windows Input Method Editors (IME) -- see Regional Options/Input Locales in the Windows Control Panel. In Unix workstations, logging in with the correct Locale will adjust the keyboard properly.

Saving questionnaire results as .txt files
Questionnaire data can be viewed as usual from the Live Data page. Questionnaire response data is stored in these files using the specified encoding. Other applications can use these files provided they take that into account.

The Language page determines the character set to use when displaying and receiving data from a questionnaire or defining that questionnaire. The default is UTF-8. When a questionnaire specifies a character set in the Language page, the following things happen:

- internally, the ViewsFlash application keeps all text as 2-byte, Unicode Strings.

- all ViewsFlash pages include an HTTP header that specifies that character set. From this, the receiving browser knows the intended encoding. Example of a Content-Type header:
text/html; charset=UTF-8

- all HTML pages created by ViewsFlash include a <meta> tag with the specified character set. For example:
<meta http-equiv="Content-type" content="text/html; charset=utf-8">

- all voting forms and application HTML pages include a hidden field 'charset'. ViewsFlash uses this field to decode the received form input to Unicode, and it is therefore essential. Example:
<input type="hidden" name="charset" value="Big5">

- questionnaire data stored in .txt files is encoded into the indicated character set.

- questionnaire data stored in a database is not encoded; the database encodes and decodes Unicode characters. In Oracle, when using character sets other than ISO-8859-1, this requires defining the database to use "utf8" as its encoding, and adding the dbcharwidthmultiplier=3 servlet parameter to the viewsflash servlet. Because of this, the values of answers are restricted to 10 characters instead of the usual 30, and the values of freeform text answers are restricted to 666 characters, unless the dbmaxvarchar=4000 servlet parameter is used to increase this number to 1333.

Earlier versions of ViewsFlash suggested using very specific Language codes, such as ISO-8859-7. It is much more reliable to use UTF-8, which is now the default, and to use a different character set in the Language page only if UTF-8 fails, which is very rare.

Next: **Installation**

Administration and Setup

From the menu bar, click on Administration to see the Administration and Setup screen. If you have configured the system with a password, you will first need to enter it in the box provided, and press the Submit button. Then complete this form with the required information, as follows.

1 Contact Information
ViewsFlash uses E-mail to communicate with the Administrator about potential and actual problems it detects, so it requires complete e-mail information. Provide the **Administrator's name**, his or her **E-mail address**, and a **from-address**, such as ViewsFlash@yourcompany.com, to identify the messages. (No mail is ever sent to this address.) Provide also the name of an **SMTP mail server** accessible from the ViewsFlash machine. When the software is installed, an e-mail will be sent to this address to verify successful installation.

2 The Public URL of the ViewsFlash application, should be set already. If it isn't, you must provide it here. It is usually best to leave this field unchanged.

3 Directory for writing static HTML. When used without a database, ViewsFlash writes HTML fragments to this directory. It is usually best to leave this field unchanged.

4 Other options and permissions.

Backup files on file system, when not running on a database, creates extra copies of all state files used by ViewsFlash; although this takes a certain amount of server resources, it's worth the extra security and is recommended. Leave unchecked when running on a database.

Allow saving data, when not running on a database, allows data from polls and surveys to be saved permanently

Allow sending e-mail allows sending the data from each submitted survey by e-mail, and allows sending e-mail invites.

Notify administrator of error conditions by E-mail sends the Administrator copies of entries in the Administrator Log that indicate potential trouble, such as unanticipated software exceptions, low-memory warnings, etc. This is highly recommended, especially during the first weeks of operation, or when running under increased volume.

Allow non-administrators to edit Styles, when not using User Security, allows users other than the Administrator to edit Styles.

Allow non-administrators to edit Invite Lists, when not using User Security, allows users other than the Administrator to edit Invite Lists.

Cookie domain is needed *only* if the ViewsFlash server URL is in a different subdomain than other systems it is exchanging cookies with. This field indicates the domain to use in the cookies exchanged for this purpose. For example, If ViewsFlash runs as polls.yourcompany.com, and your main domain is www.yourcompany.com, this parameter should be ".yourcompany.com", so that cookies tossed by ViewsFlash are visible to all parts of yourcompany.com. Please note the crucial leading "dot". If the ViewsFlash server URL is the same as your main domain, leave this field blank.

When you are done completing these entries, press Submit. ViewsFlash will check all parameters; if any errors are detected, a message will be displayed at the top of the screen.

At this point, the software is completely installed and is ready to be turned over to the design, production, and editorial people. The next step for them is to create a Test place and test questionnaires. Once the appropriate settings have been figured out, it is recommended to change the settings in the Default place and the Default Poll; their settings will be used by default when creating additional places and questionnaires.

As the Administrator, you may access all portions of the ViewsFlash application. If using **User Security**, you must assign users appropriate access rights, and users simply see only those places they have rights to. If not using User Security, users should be given the URL to their place and told to enter a password in the Settings page.

View Log

This option allows you to view the ViewsFlash application log, in which status messages are recorded.

Download Log

This option downloads the ViewsFlash application log to your browser as a zip file.

Trim Log

The Trim Log link, when the ViewsFlash log is saved on a database, lets you purge old entries from the log. When the log is not saved on a database, this link starts a new viewsflash.log file.

Suppressing repetitive log entries

Sometimes, an environment change such as the database becoming unavailable, or a user setup error, can cause a flood of identical messages to be written to the ViewsFlash log, to a poll's log, and to the Administrator's e-mail in-box. Beginning with release 3.5, duplicated error messages are automatically suppressed. The first message will be written to the log normally. The second message will add:
*** suppressing further repetitive messages

A message is suppressed when it is identical to a previous message for the same poll, and it is posted within a short interval. This interval, by default, is set to 10 seconds. It can be modified using the errorinterval servlet parameter.

Next: System Administration

System Administration

System Administration entails planning, installation and maintenance. This is a quick guide to what to read and what to do. We suggest printing this page for reference and reading it completely first.

Planning -- see **Architecture**, **Using a Database**, **Application Security**, and **Servlet Parameters**.

For a quick installation, read this page and then go to **Installation**.

The **Architecture** section describes different configurations options. Installing with a database is the preferred method. This enables a scalable, redundant **Peer Architecture** that deploys to a cluster and automatically scales to larger numbers of servers and instances and allows staging between servers. Without a relational database, only one instance of the application can run on a single machine.

The **Application Security** section explains how to secure the ViewsFlash application from unauthorized access. Simple password security only requires one parameter in viewsflash.properties. Container managed security uses Application Server Authentication and Authorization and requires determining what Users or Roles will be designated as ViewsFlash administrators. Note that if the viewsflash.properties entries for Container Security are added **after** the application is up and running successfully, deployment might be easier.

Additional servlet parameters may be useful in viewsflash.properties. The **Servlet Parameter** reference section summarizes them.

For detailed steps for Installing the ViewsFlash web application, go to **Installation**.

Where to find the ViewsFlash application

After installation, the ViewsFlash Administrator will usually find the software at http://www.yourcompany.com/ViewsFlash, except in some portals, who must access the application through a portlet's Configure mode. The Administrator will create places, where different users will create their surveys. The place URLs should be distributed to the individual users. If using User Security, the Administrator must grant appropriate access rights to the places.

Application Maintenance -- see **Maintenance**

ViewsFlash normally runs non-stop for long periods of time unattended. During an orderly shutdown, it will complete all its pending tasks before shutting down. It recovers automatically from shutdown and restarts, both orderly and unplanned.

If the database is scheduled for short periods of down time periodically, it is safe to leave ViewsFlash running; if the database is offline for long periods, it's safer to stop the ViewsFlash application before turning off the database. As a precautionary measure, it is recommended to cycle ViewsFlash periodically, like any other web application.

Other information

The **Specifications** section describes supported environments and file usage.

If you are upgrading from a previous version of ViewsFlash, please read the **Upgrading** section

Next: Architecture

Architecture

The ViewsFlash polling, survey, voting and ratings system

ViewsFlash helps web site personnel to manage surveys, forms, polls, quizzes, voting and ratings systems. It includes tools for designing, scheduling, formatting, publishing, archiving and searching, and a portal-strength form processing engine for capturing, validating, tabulating and displaying results in real time.

The Administrator creates different places for each group of users and grants them appropriate Access Rights. These users design and schedule questionnaires in their own place, protected from other users. Each place can use a different Style Template, if desired, to give its questionnaires its own look and feel.

Technical Overview

ViewsFlash is a web application hosted in an application server or a servlet container. This can be a shared application server, a server dedicated to running the ViewsFlash application, or a portal. All popular web, application servers, and portals are supported. See **Specifications**.

ViewsFlash is completely web-based; administration uses a standard web browser and requires no special client software. Questionnaire, security and scheduling information are entered using web forms.

Database Usage

ViewsFlash operates best when installed on a relational database, as some capabilities are not available without it. In a distributed configuration, database installation is mandatory. All questionnaire definitions and other settings are stored in the database. Data gathered is also stored there, in separate tables. See **Database Use**. When not installed in a database, settings and data are stored in the server's file system.

Topology, scalability, redundancy and fail-over

ViewsFlash employs an industry standard **Peer Architecture** consisting of two or more load-balanced active instances. Each instance stores state in the database, caches all necessary objects, and refreshes changed objects periodically. Results are tallied on the database and updated periodically. All instances can display live results directly to the visitor, or a publishing system can get current results by querying any instance or querying the database. Instances can be added live for dynamic scalability or can be taken offline at any time. Redundancy and fail-over are constant and automatic.

Staging

It is possible to set up a staging server, perform all questionnaire development and testing there, and then to lock and stage those questionnaires to a production server outside a firewall. The staging server exports staging files to its file system, those files are moved to the production system, and then they are imported into the production system. In such a configuration, it is possible, if necessary, to configure the production system so that application administration can be done only on the staging system. It is possible to set up a multi-tiered structure of development, test, staging, and production servers as needed; questionnaires are staged from each tier to the next when ready.

Capacity planning

For capacity planning it is necessary to take into account that design and scheduling operations comprise a negligible portion of the load on the ViewsFlash server, while composing dynamic survey pages, tabulating voting forms and composing live response pages make up most of the load.

Integrating with dynamic content

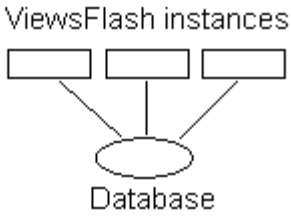
Respondents can fill out a questionnaire by going to its unique URL. Techniques are available for embedding questionnaires in **Portals**, **JSP pages**, and **iFrames**.
The **Web Services API** provides a standard methodology for other applications to manage questionnaires.

Next: Peer Architecture

Peer Architecture

ViewsFlash uses an industry standard, distributed, fault-tolerant, scalable, database-resident Peer Architecture. This section describes the considerations you need to keep in mind to use it effectively.

ViewsFlash connects to the database with one set of parameters and expects all its tables to be within the same schema or database.



All ViewsFlash instances, on multiple servers, store all their data in a relational database. All instances can perform all ViewsFlash functions equally, or some can be configured in front of a firewall for public access while others are configured behind the firewall for secure administration. In a load-balanced configuration, failover is completely automatic.

Polls, surveys, and their settings are defined by the administrators using the web-based administrative UI. These definitions are posted on the database and are available to all instances when needed. Each instance keeps cached copies of all needed objects. When an object is modified, a notice is written in a table which keeps a notification queue, which is read by all instances periodically; when an instance learns in this way that one of its cached objects is stale, it refreshes its cache from the database. This keeps all instances on all servers current while preserving the performance benefits of object caching.

Each instance is completely independent of all others. If a server crashes, upon restart the instance will refresh its objects from the database as needed and will automatically resume working with no intervention. Instances can be taken off and on line at any time.

For scalable live poll tallying, each ViewsFlash instance counts votes on its own and periodically updates the totals in the database. Meanwhile, in parallel, each instance periodically queries the database for totals to display when requested. All these intervals are set by servlet parameters; updating often gives virtually real-time results, while updating less often reduces overall system load and allows much larger voting volumes with no additional hardware.

Next: Using a database

Using a Database

This section includes additional servlet parameters to be used during Installation.

See Database specific information for additional information about using Oracle, MSSQL, MySQL and DB2 in various database versions.
Please read the appropriate section.

See also the **database** and **database error recovery** fine-tuning parameters in the Servlet Parameters section.

Planning
ViewsFlash can be deployed with or without a database. When used without one, some features are not available, such as inivitations and staging. Cluster deployment requires using a database.

To connect to the database, ViewsFlash uses a JDBC 2.0 compliant JNDI data source, managed by the application server. The data source should use Connection Pooling and auto-commit. It does not need to use XA Transactions.
The database user associated with the data source must have insert, update, and delete record rights. It is usually granted the right to create, alter, and drop tables as well; if this is not possible, see **Disabling table creation**.
ViewsFlash can run on its own database schema or on a shared schema. Allocate 10 Gigabytes to the ViewsFlash application, and provide generous auto-increments for expanding beyond that size. To use a schema other than the one normally associated with a data source, use the **servlet parameter** dbtablenameprefix.

The database is used to store **internal data** and data gathered from questionnaires.

Saving questionnaire data
While a questionnaire form is being completed, ViewsFlash writes the information to a temporary internal record. When the questionnaire is completed by clicking on its Submit button, the information is recorded permanently, before the response is shown to the respondent. If the database is temporarily out of service, storing will be retried up to three times; if the record cannot be written, the respondent is notified, the error is written to the ViewsFlash log, and the administrator is notified.

Data can be viewed from Analysis / Data page. Data can also be accessed directly off the database, with standard SQL statements, tools, and applications.

Each questionnaire specifies, on its **Save** page, how to store response data. These options include saving the questionnaire's data in its own table, in a common table, and as a .txt file. These three options can be turned on or off globally for all questionnaires by using the **Save Data** servlet parameters save.table, save.text, and save.normalized.

When data is saved in a questionnaire's own table, the table is created and updated dynamically when the Publish page is submitted, using CREATE and ALTER statements as necessary. This requires that the database user associated with the data source have the right to create, modify and drop tables. If this is not possible, see **Disabling table creation**.

In clustered configurations, data should not be saved on the file system, so disable this option with the **save.text=false** servlet parameter.

In addition to storing questionnaire data, ViewsFlash stores the live tallies of fields marked for tallying as they happen, in real time. This allows for interesting real-time applications driven by live results, which are stored in the VWFTALLIES table.

Installing for the first time with a database

If you are installing ViewsFlash for the first time and you want to use a database right away, read this section and provide the appropriate servlet parameters as described in the **Installation** section **before** deploying the ViewsFlash.war file.

If the database user associated with the data source has been granted table creation rights, all the necessary tables will be created by the ViewsFlash application. If this right has not been granted, the ViewsFlash tables must be created **before** deploying the application, using the .sql files provided in the appropriate **WEB-INF/sql** directory.

The ViewsFlash tables will be populated automatically the first time the ViewsFlash application starts.

After the application has started, examine the viewsflash.log file in the ViewsFlash data directory.

If ViewsFlash has already been deployed without a database, use the following procedure to migrate the internal ViewsFlash data to a database.

Migrating to a database
After ViewsFlash is up and running without a database, follow these steps to connect it to a database and migrate the internal data from the file system to the database.

1 Read this entire section and note all the appropriate servlet parameters needed, including database= and datasource=.
2 Create the JNDI data source in the application server and test the connection.
3 If the JNDI data source does not have create table rights, create the tables manually using the .sql files in the appropriate WEB-INF/sql directory.
4 Stop and restart the ViewsFlash application in the application server. All the data in the file system will be migrated automatically to the database.
5 Check the viewsflash log and the application server logs for any errors.

Disabling table creation

It may not be possible to grant table creation rights to the ViewsFlash data source. In this case, use the **save.table=false** servlet parameter to disable table creation.
The ViewsFlash tables must be created manually before deployment by a database administrator, as described next.

Manual table creation
To create the ViewsFlash tables manually, a database administrator should use the appropriate utility to run DDL scripts on the database. The appropriate scripts are found in the expanded ViewsFlash.war file in the WEB-INF/sql directory. There are two scripts: Create.sql and CreateNormalized.sql. The Create script creates the tables described in **Internal ViewsFlash data** below and the CreateNormalized script creates the VWFDATA Normalized database table. Some databases, like Oracle, include more than one version of the scripts; use the one that matches the database= parameter that is used.

Cluster Deployment

Deploying ViewsFlash in a cluster requires running ViewsFlash on a database. Deployment and configuration is the same as standard deployment except for the following:

1 Instead of putting servlet initialization parameters in a /etc/cogix/viewsflash.properties file, it may be easier to put them in ViewsFlash's WEB-INF/web.xml file and repack the war file. When deploying to a Weblogic cluster, it is necessary to use a database and to deploy from the war file, rather than an exploded directory.

2 If the ViewsFlash data directory, specified with the data= servlet parameter, can be on a shared file system, put it there. A single viewsflash.log file will be written to that directory by all ViewsFlash instances in all servers in the cluster.
If a shared directory is not available, then specify a local directory in data=. The directory can be shared by multiple instances in the same server. However, each server will have its own viewsflash.log file, which is not desirable, so write the viewsflash log to the database, and use the ViewsFlash application UI to read the log. Use the **dbstores=** servlet parameter to write the log to the database:

dbstores=data|polls|results|styles|votes|logs

3 Create the ViewsFlash tables manually before deployment using the .sql files in the appropriate WEB-INF\sql directory.

4 After deployment is complete, if staging will be used, perform the standard **Staging** procedures.

Moving data from a development system to a production system

To deploy ViewsFlash on a production system without any data, just follow the instructions in this section. If questionnaires have been created in the development system, use one of the procedures described below to migrate the data to the production system.

1 Deploy ViewsFlash on the production system as described here, as if there were no development system. Go to the Administration page and check all entries. Then, on the development system, use Staging to create and download a Zip file with all the data that needs to be moved , and use the Stage Import option in the production system to migrate all the data.

2 Alternatively, before deploying ViewsFlash on the production system, copy the data directory from the development system to production, and copy the database schema from development to production. Then deploy ViewsFlash on the production system as described in this section, and go to the Administration page and correct all entries which will have incorrect values left over from the development system.

Internal ViewsFlash data

In addition to the data gathered from questionnaires, ViewsFlash stores internal information, such as the questions in a questionnaire. When not using a database, the information is kept in the file system, in the ViewsFlash data directory. When using a database, it is normally kept in the database in the VWFOBJECTS table. However, this can be overriden by using the **dbstores** servlet parameter. If not specified, it defaults to:
dbstores=data|polls|results|styles|votes

This writes everything except the application log to the database; the application log is written to the data directory, or to a directory specified by the logpath= servlet parameter.

The dbstores parameter is any combination of these keywords, separated by | symbol:
dbstores=data|polls|results|styles|votes|logs

The table below indicates the different kinds of files stored in the database for each keyword.

Keyword	File	Directory where stored If not on database
data	Internal files with poll and survey content, for example	data
polls	HTML for static voting and survey forms	Web server document root
results	HTML fragments used to construct response pages.	Web server document root
styles	Style Templates	ViewsFlash/viewsflash
votes	Multi-page survey status for each voter	data/votedir
logs	viewsflash logs	data

Another consideration is writing HTML files for single page polls, which can be written on the server's file system when a poll is published. To enable this, you must do of the following:
- add the servlet parameter writehtmltodisk=1 to viewsflash.properties
- use the option "Save files to disk as well as on the database" in the place setup page.
- to avoid writing these fragments on the database altogether, specify a dbstores servlet parameter and remove polls from dbstores:
dbstores=data|results|styles|votes

Database Schema
You may want to create a Database User for ViewsFlash. Use the following as a guideline for Oracle:
CREATE USER COGIX IDENTIFIED BY PASSWORD;
GRANT CONNECT TO COGIX;
GRANT RESOURCE TO COGIX;

When you install ViewsFlash with database options, the viewsflash?Diagnose=1 command checks that the all necessary tables exist and reports this. If none of the tables are present, it automatically creates all the tables. If only some tables are present, it reports an error, and you will either have to create the missing tables by hand, or drop all tables, restart the servlet, and run Diagnose=1 again. You can create the tables beforehand with the DDL below. Substitute the indicated values for "timestamptype" and "longbinary", depending on your database vendor.

See the *.sql files in the .../ViewsFlash/WEB-INF/sql directory, which contains DDL statements for creating all needed tables in Oracle, MSSQL, DB2, and MySQL.

The following table explains what is stored in each table.

Table Name	Stores
VWFLOG	Message log
VWFOBJECTS	Application objects as serialized Java objects

VWFNOTIFICATIONS	Messages notifying instances of changes to application objects
VWFTALLIES	Real-time field tallies
VWFVOTED	Tracks who has taken a survey
VWFNOTIFICATIONSEQUENCE (Oracle only)	Used with VWFNOTIFICATIONS
VWFTRACKING	Tracks e-mail invite status
Other Tables	Each Survey that saves data has its own table. Use View Log in the survey to see the SQL used. See **Saving Survey Data**
Normalized Database Tables:	
VWFDATA	All data from all surveys. See **Extensible Database Storage**
VWFDATASEQUENCE (Oracle only)	Used with VWFDATA

Database Verification
These commands help diagnose database problems.

/servlet/viewsflash?dbcmd=dbdir
Gives a listing of all objects in the VWFOBJECTS table, which stores objects in the ViewsFlash data directory in a non-database configuration.

/servlet/viewsflash?dbcmd=dbdirs
Same as dbdir, but listing is sorted by the date when the object was last modified.

/servlet/viewsflash?dbcmd=dbdirn
Lists all currently active Notifications between different instances. These expire after a short while and are automatically removed.

Database Manipulation
The following commands are similar, but operate on a single file or object.

/servlet/viewsflash?dbcmd=pagedir&kind=KIND
where KIND is one of data, polls, results, styles
Gives a directory listing of all objects in the database of the indicated kind.

/servlet/viewsflash?dbcmd=pageget&kind=KIND&name=NNN
where KIND is one of data, polls, results, styles and
name is the name of an object, such as the name of a Style Template file (use pagedir first to make sure you have the right name).
This command retrieves the file and displays it in your browser. This can also be used as an API command.

/servlet/viewsflash?dbcmd=pageremove&kind=KIND&name=NNN
where KIND is one of data, polls, results, styles and
name is the name of an object (use pagedir first to make sure you have the right name).
This command removes the indicated file from the database.

/servlet/viewsflash?dbcmd=pageimport&kind=KIND&name=NNN
where KIND is one of data, polls, results, styles and
name is the name of an object. This can be used to add files from the data directory which were not there originally when dbcmd=dbmigrate was performed.

Next: Database specific information

Database specific information

For **Oracle**, **MSSQL**, **MySQL**, **DB2**.

Oracle 8, 9, 10, 11

Specifying the database driver

Depending on the version of Oracle that is in use, add the appropriate database= servlet parameter:

With Oracle 8, use database=Oracle
With this parameter, binary data is stored in VWFOBJECTS as LONG RAW. Text fields are stored as VARCHAR(2000).

With Oracle 9, 10, or 11, use database=Oracle9, database=Oracle10, or database=Oracle10. These are identical, but named differently for possible future changes.
With this parameter, binary data is stored in VWFOBJECTS as a BLOB. Text fields are stored as VARCHAR(2000).

With Oracle 9, 10, or 11, database=Oracle9Blob can be used if having trouble with one of the above.
With this parameter, binary data is also stored in VWFOBJECTS as a BLOB, but the driver uses Blob methods explicitly.
When using database=Oracle9Blob, the following parameter is required:
databaseextension=Oracle9DatabaseExtension

Storing very long text fields
Some questionnaires require saving very large text fields. Text fields are normally limited to 2000 or 4000 characters, depending on the Oracle version. With Oracle 9 and above, it is possible to store larger fields: select "Save in Normalized database" in the questionnaire's Save page. The data will be saved in the Normalized table VWFDATA using a CLOB field, and it can be retrieved using the Analysis / Data option. To use this option with Oracle 9 or higher, use the following servlet parameter:
databaseextension=Oracle9DatabaseExtension

Storing multi-byte Unicode data such as Japanese, Chinese, Arabico
Oracle must be configured to store data in UTF-8 encoding. Use this servlet parameter:
dbcharwidthmultiplier=3
The maximum width of fields will be reduced by a factor of 3. In Oracle 8 this is VARCHAR(666) and VARCHAR (1332) in higher versions. This does not affect data stored in CLOBs in the Normalized table VWFDATA.

Upgrading from Oracle 8 to 9,10,or 11
Change the database= parameter to the appropriate driver, as described above, and restart the ViewsFlash application.
If you use the databaseextension=Oracle9DatabaseExtension parameter described above, the VWFDATA table will be altered automatically when switching to this driver, and data is moved from the old VARCHAR field to the new CLOB field as it is accessed.
If the user associated with the datasource does not have table creation and alter rights, then stop the ViewsFlash application and manually alter the VWFDATA table with:
ALTER TABLE VWFDATA ADD (ANSWERCLOB CLOB)
and restart the ViewsFlash application.

Upgrading from Oracle 9 to 10 to 11
Change the database= parameter to the appropriate driver, as described above, and restart the ViewsFlash application.
Nothing will change in the tables themselves.

Application server notes

To configure **Tomcat 5.5**, see **JNDI Resources HOW-TO**.
To configure **Tomcat 6.0**, see **JNDI Resources HOW-TO**.
In both, use the following ViewsFlash servlet parameter, if the JNDI data source is configured as MyDB:
datasource=java:comp/env/jdbc/MyDB

If the application server does not include a JDBC driver for Oracle, **download the JDBC driver** that most closely matches the version of Oracle in use.

Using the Oracle JDBC driver directly
In Oracle only, it is possible to connect to the database without using a datasource.
This method can be used with Oracle 8, 9, or 10. Binary data is stored as LONG RAW and text fields as VARCHAR(2000). Normalized data is stored as a VARCHAR.

Use the following servlet parameters:

dbdatabaseconnectionclass=OraclePooledDatasource
dbstores=data|polls|results|styles|votes
dbdrivertype=thin
dbdatasource=yourdatabaseserver.yourcompany.com
dbdatabasename=yourdatabasename
dbprotocol=tcp
dbport=1521
dbusername=yourviewsflashusername
dbpassword=yourviewsflashuserpassword

MSSQL Server 7, MSSQL 2000, MSSQL 2005

Specifying the database driver

The same database= servlet parameter is used for all these versions of MSSQL:
database=MSSQL
With this parameter, binary data is stored in VWFOBJECTS as an IMAGE field. Text fields are stored as NVARCHAR(2000).

Storing very long text fields
Some questionnaires require saving very large text fields. Text fields are normally limited to 2000 characters. To store larger fields, select "Save in Normalized database" in the questionnaire's Save page. The data will be saved in the Normalized table VWFDATA, and it can be retrieved using the Analysis / Data option.

ViewsFlash releases prior to 5.8 used VARCHAR(2000) for the text field in the Normalized table. In release 5.8, this has been changed to NVARCHAR(2000) for MSSQL Server 7 and MSSQL Server 2000, and to NVARCHAR(max) in MSSQL 2005. Thus, in ViewsFlash 5.8 with MSSQL 2005, it is possible to save practically unlimited text in the Normalized database.

Storing multi-byte Unicode data such as Japanese, Chinese, Arabic
ViewsFlash releases prior to 5.8 used VARCHAR(2000) for text fields. 5.8 and above use NVARCHAR(2000), allowing storage of Unicode data such as Japanese, Chinese, Arabic. There is no need to modify any tables when upgrading unless storing Unicode data is contemplated in the future; in that case, use an ALTER TABLE to change the desired fields from VARCHAR to NVARCHAR. For the Normalized database table VWFDATA, stop the ViewsFlash application and manually alter the VWFDATA table with:
ALTER TABLE VWFDATA ALTER COLUMN ANSWER NVARCHAR(max)
and restart the ViewsFlash application.

Upgrading from MSSQL 7 or MSSQL 2000 to MSSQL 2005
In MSSQL 2005, it is possible to store practically unlimited large blocks of text in the Normalized database in VWFDATA.
If upgrading to MSSQL 2005 from an earlier version, stop the ViewsFlash application and manually alter the VWFDATA table with:
ALTER TABLE VWFDATA ALTER COLUMN ANSWER NVARCHAR(max)
and restart the ViewsFlash application.

Application server notes

When using Adobe **JRun,** go to the JRun Management Console and, in the default server, under Resources, create a JDBC Data Source linking to the MSSQL server. Then provide ViewsFlash with its JNDI name (usually MSSQL) in its servlet parameters.

To configure **Tomcat 5.5**, see **JNDI Resources HOW-TO**.
To configure **Tomcat 6.0**, see **JNDI Resources HOW-TO**.
In both, use the following ViewsFlash servlet parameter, if the JNDI data source is configured as MyDB:
datasource=java:comp/env/jdbc/MyDB

If the application server does not include a JDBC driver for MSSQL, **download the Microsoft JDBC driver** that most closely matches the version of MSSQL in use.

MySQL 4,5

Specifying the database driver

The same database= servlet parameter is used for all these versions of MySQL:
database=MySQL
With this parameter, binary data is stored in VWFOBJECTS as a MEDIUMBLOB field. Text fields are stored as VARCHAR(4000) in MySQL 5 and VARCHAR(255) in MySQL 4.

Storing very long text fields
Some questionnaires require saving very large text fields. Text fields are normally limited to 2000 characters. To store larger fields, select "Save in Normalized database" in the questionnaire's Save page. The data will be saved in the Normalized table VWFDATA as a TEXT field, of practically unlimited size, and it can be retrieved using the Analysis / Data option.

Storing multi-byte Unicode data such as Japanese, Chinese, Arabic
MySQL automatically stores Unicode in the server's default character set; we recommend using "utf8".

Upgrading from MySQL 3 to 4 to 5
Nothing needs to change.

Application server notes

When using Adobe **JRun,** go to the JRun Management Console and, in the default server, under Resources, create a JDBC Data Source linking to the MySQL server. Then provide ViewsFlash with the Data Source's JNDI name in its datasource= servlet parameter.

To configure **Tomcat 5.5**, see **JNDI Resources HOW-TO**.
To configure **Tomcat 6.0**, see **JNDI Resources HOW-TO**.
In both, use the following ViewsFlash servlet parameter, if the JNDI data source is configured as MyDB:
datasource=java:comp/env/jdbc/MyDB

If the application server does not include a JDBC driver for MySQL, **download the MySQL JDBC driver** that most closely matches the version of MySQL in use.

DB2 release 8 and 9

The DB2 database tablespace assigned to ViewsFlash should include a Table Space for Regular data set up with a large page size, such as 32K. This is needed to store tables from large surveys with many questions on their own tables, which often require a large page size.

Specifying the database driver

The same database= servlet parameter is used for all these versions of DB2:
database=DB2
With this parameter, binary data is stored in VWFOBJECTS as a BLOB field. Text fields are stored as VARCHAR(2000).

Storing very long text fields
Some questionnaires require saving very large text fields. Text fields are normally limited to 2000 characters. To store larger fields, select "Save in Normalized database" in the questionnaire's Save page. The data will be saved in the Normalized table VWFDATA as a CLOB field, of practically unlimited size, and it can be retrieved using the Analysis / Data option.

Storing multi-byte Unicode data such as Japanese, Chinese, Arabic
DB2 should be configured to use UTF-8. Use this servlet parameter:
dbcharwidthmultiplier=3
With this parameter, the maximum size of text fields will be 667. This does not affect data stored in the Normalized table VWFDATA.

Upgrading from DB2 release 8 to 9, or 9 to 10
No action required.

Application server notes
In WebSphere application server, if the JNDI data source is named MyDB, use datasource=MyDB as the ViewsFlash servlet parameter.

Using schemas and tablespaces
To use tables inside a specific schema, see the **dbtablenameprefix** servlet parameter, eg dbtablenameprefix=COGIX_SCHEMA. (with a trailing period). All tables will use two-part names, eg COGIX_SCHEMA.VWFOBJECTS.

See the **dbcreatetablespacestatement** parameter to specify a CREATE TABLESPACE statement.

See the **dbtablenamesuffix** parameter to allocate database tables associated with specific questionnaires to a named or random tablespace.

Next: Using other Data Sources

Using alternative data sources

Several Actions, namely **SQLSelect**, **SQLUpdate**, **SQLDisplay**, **DynamicDropDown** and **CascadingMenus** can operate with data in a different database.

Normally, ViewsFlash database operations use a single database schema referred to using a JNDI data source, defined by ViewsFlash servlet parameters, especially database and databasename.
These operations allow referring to other datasources that can refer to different schemas in the same database, to other databases, and even to tables in different brands of database (eg Oracle / DB2).

The Actions mentioned here accept an additional parameter:
datasource=JNDIDataSourcename

The SQL statements used by those queries will be executed against the database specified by this alternate JNDI data source. There is no need to specify the brand of database (Oracle, etc). The SQL statements need to conform to the syntax of the particular database, of course.
All of the database configuration parameters, such as user ID, password, schema, etc., have to be specified in the data source.

Note that all these Actions accept the [/dbtablenameprefix] syntax in the query. For example,
select * from [/dbtablenameprefix]tablename where x=1
will replace [/dbtablenameprefix] with the value used for that servlet parameter if one is present.

Note that the exact syntax of the datasource parameter depends on the application server being used.
For most application servers, it is just the name of the JNDI data source:
datasource=NameOfTheJNDIDataSource
Tomcat, however, requires:
datasource=java:comp/env/NameOfTheJNDIDataSource

Next: Application Security

Application Security

Overview
When ViewsFlash runs in J2EE compliant application server, it can use declarative and programmatic security to provide very fine-grained access control by users and roles to different functionality. When such security is not available, password security and port security can provide less fine-grained access control. The methods to be used are summarized in the table below. The most sophisticated methods are listed first.

Method	Benefit	Requirement	How it is implemented
User security	All ViewsFlash users must be authenticated by the application server. Optionally, they can also be required to belong to an established role or group. The ViewsFlash Administrator grants specific access rights to other users and groups or roles. Different users are assigned rights to their own places and specific functions. For example, in a health care organization, a Research place can authorize certain users to create surveys, while other users can analyze survey results and yet other users can examine the raw survey data.	A web or application server with User Authentication. Groups and roles can be defined by the application server or directly in ViewsFlash.	Two servlets (viewsflash and vfadmin) are set up in the application's web.xml file, with different access controls. Additional fine-grained access rights are administered internally by ViewsFlash.
Servlet security	All ViewsFlash users must be authenticated by the application server, and can be required to belong to an established role or group. User security is a superset of servlet security.	A web or application server with User Authentication.	Two servlets (viewsflash and vfadmin) are set up in the application's web.xml file, with different access controls.
Password security	The ViewsFlash administrator has a password that grants access to the entire application. Each place can set a password to restrict access to it	This option can be used with all other forms of security except User security.	The ViewsFlash application verifies the passwords.
Port security	The ViewsFlash application can be required to use a different TCP port for its administrative functions.	This option can be combined with all other forms of security.	The network topology makes sure that only computers at authorized IP addresses have access to the application's servlets.

User and Servlet security are described in **Application Server Security**. If one of these methods is not used, a WARNING will be present in the Administration page. Using these forms of application security is highly recommended.

Password Security
An administrative password is specified as a ViewsFlash servlet parameter:
adminpw=xxxxxx
This will protect the more sensitive areas of the application with an Administrator Password. Individual places can also be protected with their own passwords, and all polls or surveys in password protected places will require that password to access their user interface. A complementary servlet parameter, "admintimeoutminutes", makes the cookie expire after the indicated number of minutes, so that a user will need to enter the password again after that time. This guards against someone inadvertently leaving their ViewsFlash application unattended in a browser. This is the minimum amount of security that is recommended.

Port Security
ViewsFlash can be configured so that the application's administrative interface can be accessed only on a particular TCP port. To do so, use the following ViewsFlash servlet parameter:
adminport=8087

With this, the URL for the ViewsFlash application will then become:
http://www.yourcompany.com:8087/ViewsFlash

If someone attempts to access the application through another port, the response will be a 404 Not Found or other rejection code.

Similarly, the HTTP ViewsFlash commands used in the Web Services APIs can be protected by requiring another TCP port in their URLs. To do so, use a ViewsFlash servlet parameter such as "apiport=8089". By default, this parameter is set to a port that is normally unavailable, and it must be set explicitly to an available port in order to use these APIs. See **How to use the Web Services API** to enable it when using it from JSP pages. **Setting apiport=80 allows public use of these APIs and is not recommended**.

Both Apache and IIS allow receiving requests from more than one TCP port. If this is not possible and your are running on a database, a ViewsFlash instance can be installed in two instances of the web server instances that allow access on two TCP ports.

When using these alternate ports, consider configuring router ACLs so that only selected people or servers are allowed access. For example, only a certain set of users could be allowed access to the administrative user interface on port 81. A results server that requests ViewsFlash data in real time for analysis or display using the Web Services API can access it on port 1999, but no other servers would be granted that privilege.

Next: Application Server Security

Application Server Security

Overview
In a J2EE compliant application server, ViewsFlash can use declarative security to provide access control to the application; this is referred to here as "servlet security". ViewsFlash can enhance this with additional programmatic security to provide fine-grained access control to different functionality within the application to users and roles; we refer to this as "user security".

Windows note: In the J2EE specification, users belong to Roles. In Windows, users belong to Groups, and application servers implement J2EE roles as Windows groups. When we use the word Role we refer to Roles or Groups interchangeably.

To use Application Server security, the ViewsFlash application's WEB-INF/web.xml file must include a security-constraint element that requires users to be authenticated before accessing the vfadmin servlet, which is the ViewsFlash user interface. This element also requires that users belong to one or more Roles. For a complete explanation of security-constraint see the Servlet Specification 2.2 and your application server documentation. Some application servers that implement Servlet Specification 2.3 may allow using * for role-name; this allows anyone in any role to access the application, but roles still have to be defined and users must belong to one or more roles.

Difference between servlet security and user security

Both methods provide the ViewsFlash administrative interface at /servlet/vfadmin and survey and poll taking at /servlet/viewsflash, and require authentication by the application server.

Servlet security, which can be combined with password and port security, adds another layer of protection to the ViewsFlash application.

User security, in addition, provides fine-grained access control to resources, and is well suited to enterprise environments where access needs to be carefully doled out to specific users. The ViewsFlash administrator acquires the responsibility for managing access control. Different options allow for fully secured and trusted security environments. Password security is not available when using user security and is automatically disabled if requested.

Access Rights are granted to individual users by listing their login user IDs, or to roles by listing their names, prefixed by an @ sign. For example, tom.jones,@HospitalPersonnel refers to user ID tom.jones and to anyone belonging to role HospitalPersonnel.

Whether a user belongs to a Role or Groups is determined by the application server. Any role or group names described to ViewsFlash will be automatically mapped by the application server through the standard Role Mapping mechanism, if active. It is possible to bypass the application server's role mapping and provide your own mappings between users and roles, as described in **Extensible Authentication**.

Installation for both Servlet and User security

1. Install ViewsFlash normally, using the default web.xml file provided, and check that the application is installed properly. This installs ViewsFlash as a web application at
http://www.yourcompany.com/ViewsFlash

2. Review /ViewsFlash/WEB-INF/web.xml. The <web-resource-collection> element for /servlet/vfadmin requires that application users must be authenticated by the application server. The <auth-constraint> element specifies that those application users must also be in the role of "ViewsFlashUser", and the <security-role> element defines that role. Instead of ViewsFlashUser you can use any role you like.

The <web-resource-collection> element for /servlet/vfauth requires that respondents who fill out questionnaires where basic authentication is specified in the questionnaire's the **Settings / Security** page must be authenticated by the application server, at the role of ViewsFlashRespondent.

The <login-config> element specifies that authentication will use the login form located at /ViewsFlash/WEB-INF/vflogin.html. Other standard methods may be used, of course, such as BASIC, DIGEST, or CLIENT-CERT, with appropriate parameters.

3. Add to the viewsflash.properties file an "appsecurity" parameter. This enforces sending survey and poll form submissions to the /viewsflash servlet and administrative UI form submissions to the protected /vfadmin servlet. Add one of the following:
appsecurity=servlet
appsecurity=user

Also add:
administrators=<list>

This parameter designates the ViewsFlash administrator(s). Its format is a list of names and roles prefixed by an @ sign. For example:
administrators=Tom.Jones,Nancy.Drew,@ViewsFlashAdministrator,@LawFaculty
This example appoints logged in users Tom.Jones and Nancy.Drew as ViewsFlash administrators, as well as any logged in user who has the role ViewsFlashAdministrator or LawFaculty.

The adminpw servlet parameter is not needed when using the appsecurity parameter.

4. If configuring for servlet security, installation is complete. Restart the ViewsFlash application and use it.

5. If configuring for user security, continue installation by adding the following parameters to viewsflash.properties:

appsecurity=user
administrators=<list>

In <list>, use a series of names and roles (or groups) prefixed by an @ sign. For example:
administrators=Tom.Jones,Nancy.Drew,@ViewsFlashAdministrator
Users Tom.Jones and Nancy.Drew are now ViewsFlash administrators, as well as any user with the role ViewsFlashAdministrator.

Additional keywords can be used with appsecurity=user, by entering on the same line, separated by a comma. These options change significantly how security is enforced. For example:
appsecurity=user,closed,trustadmin

appsecurity=user,closed (the default) requires that Access Rights be assigned to places, styles, or invitation lists **before** they can be accessed.
appsecurity=user,open allows using, and changing settings in, any place, style or invitation list **until** their Access Rights are restricted explicitly.
So, in general, "closed" provides a system where Access Rights must be explicitly granted, and "open" provides a system where Access Rights must be explicitly restricted. Open is recommended for small groups of users who work on a single project, for example. Upgrade considerations are covered in **Using User Security**.

appsecurity=user,notrustadmin prevents the ViewsFlash Administrator from assigning himself Access Rights, either directly or through a Role he belongs to. The default is appsecurity=user,trustadmin, which allows him to assign himself Access Rights to places, styles, and invitation lists.

appsecurity=user,adminuionly allows application access only to ViewsFlash Administrators, and only to those functions that they can perform, namely changing Access Rights. This option can be used in combination with **Staging** to secure a production server.

appsecurity=user,noui prevents any access to the application at all. This option can be used in combination with **Staging** to secure a production server.

Password security is ignored when using User Security. If existing places use passwords, they will not be required any longer.

6. For additional security, you may want to protect the documentation and the upload servlet by adding these **XML security constraints** to web.xml.

7. Restart the ViewsFlash application.

Next: Using User Security

Using User Security

*Before activating User Security, use this section for planning who should be granted access rights to what. Go to the Places, Styles, and Invite List pages, print them, and use them as a checklist when compiling what rights to assign to whom. Refer to the **Tools** section for instructions.*

Access Rights

Access Right	Allows Users to	Page
Create places	Create new places	Access Rights
Create styles	Create new styles	Access Rights
Create invite lists	Create new invite lists	Access Rights
Create surveys	Set up, modify, and schedule surveys in a place.	Place
Analyze data	Analyze aggregate data in surveys in a place.	Place
Examine raw data	View individual response data in surveys in a place.	Place
Lock and stage	Lock and stage surveys in a place.	Place
Grant Rights	Delegate assigning the right to create surveys, analyze data, and examine raw data in a place.	Place
Use style	Use a style template	Styles
Modify style	Modify a style template	Styles
Use list	Use an invitation list	Invite Lists
Modify list	Modify an invitation list	Invite Lists

Only ViewsFlash Administrator(s) can grant Access Rights. They do so in the pages listed in the table above, using options that are available only to Administrators. The Access Rights page grants the right to create places, styles and invite lists, The places, Styles and Invite Lists pages have an additional selection that opens an Access Rights page where the Access Rights of the items selected can be changed.

places
If ViewsFlash is used in a highly trusted, small group, use appsecurity=user,open. This lets everyone have access to all places; if any require special protection, the administrator can use Access Rights to limit who can create surveys, analyze results, or examine survey data. Before installing this parameter, the Administrator should get an idea from users as to which places require additional security. After restarting the application server with this parameter, the Administrator should go through every place and add appropriate Access Rights.

Most of the time, user security is installed using appsecurity=user,closed (closed is the default). When the application server is restarted with this parameter, **nobody** will be able to access any places; the application will be essentially locked down. Therefore, go through existing places in advance, and plan ahead to see what users and/or groups should be granted what rights in what places.

When a place is created, the user who creates it is assigned all rights, and the Administrator should restrict those rights if appropriate. Deleting a place requires the right to Create Surveys in that place.

Typically, a group of places belongs to a set of individuals. Create a role for them, and assign that role appropriate access rights in every place they use. If some of those places require more restrictive rights, such as allowing only certain individuals to analyze data, grant those rights to those individuals or roles only.

Styles
When using appsecurity=user,closed, all users are granted the right to Use the core styles that come preinstalled with the product, and nobody is granted the right to modify any styles. The right to modify styles is granted by the Administrator from the Styles page.

When a style is created, the user who creates it is granted the right to Use it and Modify it; the Administrator should change these rights if appropriate. Deleting a style requires the right to Modify it.

A style that has been created for a particular purpose is typically used in very few places by very few users; assign those users the right to Use that style and assign one or more of them the right to Modify that style.

When upgrading to User Security after surveys have already been created, any existing survey that uses a style is automatically granted the right to use that style, since otherwise existing surveys would stop working!

Invite Lists
When using appsecurity=user,closed, no one is granted the right to modify any invite lists. However, any survey or place or survey that uses an invite list is granted the right to that list automatically.

When an invite list is created, the user who creates it is granted the right to Use it and Modify it; the Administrator should change these rights if appropriate. Deleting an invite list requires the right to Modify it.

An invite list that has been created for a particular purpose is typically used in very few places by very few users; assign those users the right to Use that list and assign one or more of them the right to Modify the list.

Visibility and Unauthorized entry
In the places, Styles, and Invite Lists pages, as well as any pull-down menus, the only entities listed are those that the logged in user has the right to Use or Modify. The Access Rights granted to each user, either explicitly or by membership in a role, provide that user with a private view of what is relevant to him in the ViewsFlash application, while other user's views are kept private. Even a malicious user who copies or sniffs an application URL will not be allowed to enter an unauthorized place or perform any operations he does not have appropriate Access Rights for. Such an attempt, depending on its severity, will be either denied with a Forbidden (403) HTTP return code or with a message denying the request.

Administrator Tools
The Access Rights button in the main menu opens the **User Access Rights** page which allows revoking user rights and granting the right to create places,styles and invite lists, and gives a complete listing of who has what access rights throughout the application.

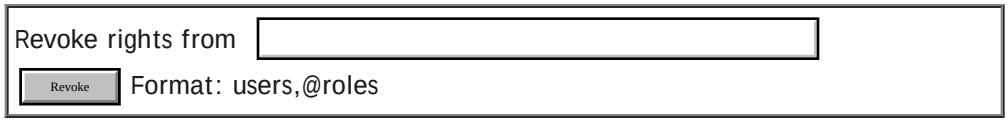
The places, Styles, and Invite Lists pages have an "Access Rights of selected" option, which allows selecting several of the listed entities and changing their Access Rights. A second option, "Who has Access Rights", shows next to each entity the users and roles that have rights to it. The Styles and Invitation Lists pages also have a Where Used option which shows what places use which styles or lists.

Delegated Security
The Administrator has the right to set all Access Rights throughout the application. The Administrator can also delegate the right to assign Access Rights in a place to someone else by simply filling out the Access Rights box in the Assign Access Rights page. The designated users can then decide who should have the right to create surveys, analyze data, and examine raw data in that place, and appropriate menu items become visible to those users. Only Administrators can delegate Security.

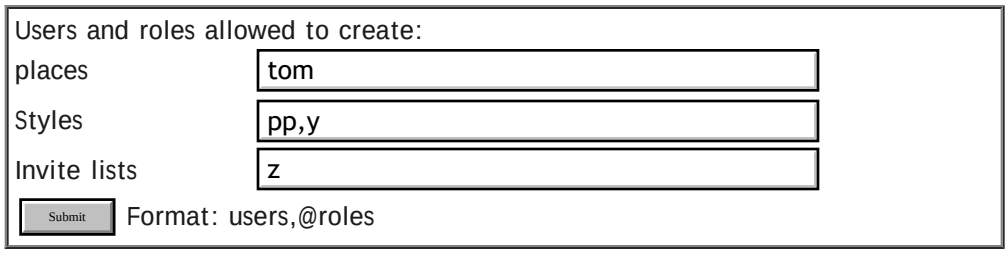
Next: User Access Rights

User Access Rights

The Access Rights button in the menu bar, visible to the Administrator only, leads to the User Access Rights page, which has useful functions for managing Access Rights.
Other useful **tools** are the Access Rights of Selected, Who has Access Rights, and Where Used options in the places, Styles, and Invite Lists pages. For complete information, see **Using User Security**.



The first form, above, is useful for quickly revoking all rights that a user or group has everywhere in the ViewsFlash application.
Simply enter a list of user IDs and roles and press Revoke.
For example, tom.jones,@ProjectX will revoke any rights that tom.jones and users in role ProjectX have. Note that the role name is prefixed with an @ sign.



The second form grants users and roles the Access Right to create places, styles, and invite lists. Next to each category, enter a list of user IDs and roles and press Submit.

For example, entering @SurveyCreators,susan.petersen in the places box will grant her and anyone belonging to the role SurveyCreators the right to create places. In the created places, they will also be granted the right to create surveys, analyze data, and view raw data. To change those rights, the Administrator must go to the places page and change the Access Rights to the created place.

Similarly, when a Style or an Invite List is created, the user who creates them is granted the right to Use and Modify it. To change those rights, The Administrator must go to the Styles or Invite Lists page and modify the Access Rights.

Next: Using LDAP

Using LDAP

In ViewsFlash, LDAP can be used in several areas.

1. User Authentication and Authorization. If the Application Server hosting the ViewsFlash application supports LDAP, then its Users and Roles are available to use with **User Security**. For example, users in the role of ViewsFlashSurveyCreator or LawFaculty could be granted the Access Right to create surveys. Selecting Basic or Role authentication in a questionnaire's Security page uses getRemoteUser() and isUserInRole(). When using these options, ViewsFlash does NOT communicate directly with the LDAP server.

If you need for ViewsFlash to communicate directly with the LDAP server, please contact Cogix. The production release does not include these components.

2. An Invite List can use an LDAP Group to define a set of users that should be invited to respond to a questionnaire, as well as to prevent users not in that group from participating, and tracking who has completed it and remind them to do so. See Using LDAP Invite Lists.

3. Using an LDAP directory instead of the Application Server to identify users and Roles. When an application server is set up to authenticate users only (using perhaps a method such as webauth) and the definition of what user is in what role lives in an independent LDAP server, then ViewsFlash can communicate with that LDAP server directly to determine if a user is in a particular role. This is also useful when using User Security to use LDAP Groups directly, without having to map users to groups in the application server. To activate this, use the lookuprole=LDAPLookupRole servlet parameter. This causes roles to be looked up directly in the LDAP server instead of using the Application server. See Role Lookup Extensibility in **Extensible Authentication**.

Uses an optional jndiroleproperties servlet parameter to point to an LDAP property file, described below. If this servlet parameter is not used, then the /etc/cogix/ldap.properties must contain appropriate parameters.

3. An Invite List can use an LDAP server to select who should be invited to take a survey, in addition to using a database table or a comma-delimited file.

Uses an optional jndilistproperties servlet parameter to point to an LDAP property file, described below. If this servlet parameter is not used, then the /etc/cogix/ldap.properties must contain appropriate parameters.

LDAP properties files

These files contain a series of parameters describing an LDAP server. It is possible to have different LDAP properties files for different purposes. If not, they default to /etc/cogix/ldap.properties.

For complete definition of other parameters available, see
http://java.sun.com/j2se/1.3/docs/guide/jndi/spec/jndi/properties.html

Linux LDAP server parameters

```
# define the JNDI context factory:
java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory

# define a Linux LDAP server:
java.naming.provider.url=ldap://yourcompany.com:389/dc=yourcompany,dc=com
# these may be required to present credentials -- consult with the LDAP Administrator
#java.naming.security.principal=cn=Manager,dc=yourcompany,dc=com
#java.naming.security.credentials=password
#java.naming.security.protocol=ssl

# Role Lookup related parameters. If not using roles, omit them.

# roleBase: Set to the element that is the base of the search for matching roles.
# If not specified, the entire directory context will be searched.
roleBase=ou=Sets

# roleSearch: Set to a filter expression used to search for role/group elements
# in the roleBase context. The search will find those roles/groups
# that contain a given username. Use {0} as a placeholder for the
# username.
roleSearch=(memberUid={0})
```

```
# roleSubtree: Determines whether the role/group search will be restricted to
# objects within the roleBase or will recurse through subtrees.
# The default is false, i.e., restrict the search to one level.
roleSubtree=false
# roleName: Set to the attribute name of the role/group
roleName=cn
```

Windows 2000 LDAP server parameters

```
# define the JNDI context factory:
java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory

# define a Windows 2000 LDAP server:
java.naming.provider.url=ldap://servername.domainname.yourcompany.cogix.com:389/dc=domainname,dc=yourcompany,dc=com

# these may be required to present credentials -- consult with the LDAP Administrator
#java.naming.security.principal=domainname\\UserName
#java.naming.security.credentials=password
#java.naming.security.protocol=ssl

# Role Lookup related parameters. If not using roles, omit them.

# roleBase: Set to the element that is the base of the search for matching roles.
# If not specified, the entire directory context will be searched.
roleBase=ou=Users

# roleSearch: Set to a filter expression used to search for role/group elements
# in the roleBase context. The search will find those roles/groups
# that contain a given username. Use {0} as a placeholder for the
# username.
roleSearch=(member={0})

# roleSubtree: Determines whether the role/group search will be restricted to
# objects within the roleBase or will recurse through subtrees.
# The default is false, i.e., restrict the search to one level.
roleSubtree=false

# roleName: Set to the attribute name of the role/group
roleName=cn
```

Next: Localization

Localization Guide

See also: **Internationalization** for how to create surveys in multiple languages.

ViewsFlash comes configured with English as its native language. This section describes how to configure it for a different default language.

Changing the default questionnaire language. In this example, we use GB2312 Chinese.

1. By default, the ViewsFlash application encodes all pages as UTF-8. If this needs to be changed, include in viewsflash.properties:
   appcharset=GB2312
   With this setting, all application pages will accept this encoding by default, and survey forms will accept this character set without having to use the Language page.
2. Open the directory ViewsFlash/WEB-INF/classes/com/cogix/vwflanguage. Copy Text.properties to Text_en.properties. This file contains messages such as "Please answer this question", which are shown to respondents when appropriate. Translate the values in Text.properties into the desired native language and save the file using Unicode Encoding. If you need to translate these into additional languages, name the files Text_de.properties for German, for example. These messages are used in the Styles to provide the text for default buttons and validation messages. However, the preferred way to translate messages is to use the **Messages** page for the survey in that language.
3. Open the directory ViewsFlash/WEB-INF/classes/com/cogix. Copy Portlettext.properties to Portlettext_en, if this file is present. If it is not, omit this step (this file is included only with the Cogix Survey Portlet for JSR168 compatible portals). Translate the values in Portlettext.properties into the desired native language and save the file using Unicode Encoding. If you need to translate these into additional languages, name the files Portlettext_de.properties for German, for example. See the Internationalization section of your portal user guide for more details.
4. Questionnaire publication dates are represented in a form that is appropriate to the application's Locale, as set in the application server. A user can change the format and time zone of these dates by using **My ViewsFlash**.

Style templates

Style templates use **[/lookupi18n]** tags that automatically translate the Next, Submit, and other buttons and language-specific messages. These tags use the Text.properties files described above.

Translating the entire ViewsFlash application.

It is possible to create a completely localized version of the ViewsFlash application. For further information, please contact Cogix.

Next: Upgrading

Upgrading ViewsFlash

Upgrading to ViewsFlash 7 from versions 4 and higher

ViewsFlash does not coexist simultaneously with earlier releases. If you have two or more instances of ViewsFlash sharing the same database, shut down all instances before upgrading, and upgrade ViewsFlash on all instances. When you upgrade, all data is upgraded automatically. Backup the database before upgrading.

The upgrade process consists of reviewing the servlet parameters, shutting down the ViewsFlash application, upgrading the ViewsFlash application with the new ViewsFlash.war file, and bringing the application back up.

Changes to be aware of before upgrading to ViewsFlash 7:

Uploaded attachments are now stored in the database by default, rather than the file system, and attachments are copied from the file system to the database's VWFOBJECTS table automatically upon startup. To prevent this and continue with the legacy behavior of storing attachments in the file system, see **Attachments** for the exact servlet parameters to use.

When using Oracle as the ViewsFlash database, note the following change.

If you were using database=Oracle, Oracle9,Oracle10, or Oracle11, and you did not specify a databaseextension parameter, a databaseextension parameter of "OracleDatabaseExtension" was used by default. In ViewsFlash 7, a new default is used, namely: OracleDatabaseExtension, Oracle9DatabaseExtension,Oracle10DatabaseExtension, or Oracle11DatabaseExtension. The effect of this change is that a new column, ANSWERCLOB, of type CLOB, is added to the VWFDATA table. This column is used to store data from text fields, allowing storage of very large text fields automatically. Data that has already been stored in the VWFDATA table is automatically moved to the new column when necessary. To prevent this change and to use the legacy behavior, provide explicitly the servlet parameter:
databaseextension=OracleDatabaseExtension.

After downloading the ViewsFlash.war file, examine its WEB-INF/web.xml file and compare it to your deployed web.xml. If they are identical, you can simply redeploy the new war file. If you have made changes to your web.xml file, the easiest way to upgrade is to unpack the war file, replace the web.xml file with yours, repack the war file, and redeploy it. Alternatively, you can stop the application, save web.xml, redeploy the new war file, restore web.xml, and restart the application.

All out of the box Styles are upgraded automatically when you upgrade to ViewsFlash 7. Copies of styles that have been created and modified will continue to work as before.

Upgrading from ViewsFlash 3.5 or higher

ViewsFlash 5 requires new license keys. Contact Cogix to request an upgraded license key. The key must be installed in the servlet parameters, usually stored in viewsflash.properties or in web.xml.

Read the **release notes** and if any ViewsFlash **servlet parameters** need to be changed, do so now. Read also **Upgrading Style Templates**.

Test your installation as described in the **Installation** section and **troubleshoot** if necessary.

If you have not made changes to the templates that need to be upgraded as listed by Diagnose=1 command, use this command to upgrade them: ?cmd=upgradestyles.

Upgrading from versions older than ViewsFlash 3.5

Next: Upgrading Style Templates

Upgrading Style Templates from earlier versions

When you upgrade ViewsFlash to a new version, styles that ship with ViewsFlash are updated automatically. Therefore, it is very important not to change the styles that ship with ViewsFlash, and to create new styles, by making copies of them, if modifications are needed. In the Standard_ style in ViewsFlash 5, most of the Javascript functionality is in a separate file, making it easier to construct alternate styles.

Next: Maintenance and Tuning

Maintenance and Tuning

System Administration entails planning, installation, setup and maintenance.

Maintenance consists of making sure that there is sufficient disk space on the ViewsFlash disks and that its data directories are backed up periodically. ViewsFlash will restart automatically and restore its state from its files as needed if the server is recycled.

ViewsFlash backups are kept, by default in a "backup" subdirectory of the main "data" directory. The servlet initialization parameter "backupdir" can be used to specify a different directory. If the backup directory is on a different volume, however, performance can degrade seriously.

When an error occurs, ViewsFlash sends an e-mail alert to the person identified in the Administration page and writes them to the ViewsFlash log. Both of these should be monitored constantly.

Tuning.

Monitor CPU utilization of each server to keep average under 75%. In a non-database configuration, the *flushtime, flushvotes, flushsize* **servlet parameters** should be fine-tuned until results under load are nearly real-time. In a database configuration, the storeflushtime, refreshresultstime and notificationinterval servlet parameters should be fine-tuned.

Monitoring API
Large installations use software monitoring and diagnostic tools to ascertain application health. The ViewsFlash Monitoring API consists of a series of HTTP commands that a monitoring application can send ViewsFlash and can then ascertain application health by examining the response, as follows. Make sure ViewsFlash is up and running on each machine before using this API! Please test this API with your systems monitoring software carefully before putting into production.

cmd=selftest
This will check that ViewsFlash has access to its data directory in either read or read/write mode, depending on whether the ViewsFlash instance is running as a satellite or a master.

cmd=selftest&masterul=1
In a satellite instance, this command will also check to see if the satellite can reach the master servlet at the address indicated by the masterurl parameter.

The selftest command will return "OK" as the first two letters if the tests are successful. Otherwise, the first two letters will be "NG", and the rest of the line will indicate the cause of failure, as follows. The 4th and 5th character describe the error state, which can be interpreted accordingly. Note that NOT all error returns are fatal.

Return Message	Meaning
NG,01, missing 'data' parameter on Master	Servlet unusable. Check the servlet parameters.
NG,02, cannot write data on Master	Servlet unusable. The 'data' directory cannot be written to.
NG,03, missing 'data' parameter on Satellite	Servlet unusable. Check the servlet parameters.
NG,04, cannot read data on Satellite	Servlet unusable. The 'data' directory cannot be read from.
NG,05, failed to connect to Master	Danger condition. The satellite cannot communicate to the master. The condition should be remedied as quickly as possible. If it persists for a more than a few minutes, the satellite should be marked as temporarily unavailable, until the selftest command returns OK again. The satellite keeps trying to reach the master up to 100 times until the connection is restored. If possible, do not bring down the satellite while restoring the connection. This code requires that you include the masterurl=1 parameter in the command string.
NG,06, missing 'data' parameter	Servlet unusable. Check the servlet parameters.

NG,00, something's wrong	Unexpected condition. If an unexpected exception occurs during the self-test, it is reported here. It should be examined. The servlet should not be considered unusable just because this message appears.

Next: Specifications

Requirements

Requirements

ViewsFlash is a standard Java Web Application that runs on any platform that supports the **Servlet v2.2** specification. The ViewsFlash JSR168 compatible Survey Portlet requires a JSR168 certified portal.

Hardware	Intel X86, Sun Sparc, System X, System i, System p, HP/UX, RS/6000, Apple
Operating Systems	Redhat Linux, SUSE Linux, Windows XP, Windows Server 2003/2008, Windows Vista, Solaris 8/9/10, AIX, HP/UX, OS/X
Java	1.4 or higher, including Sun, BEA JRockit, IBM
Application Servers	Weblogic 8,9,10. WebSphere 4,5,6, Oracle 9iAS/10iAS, Sun Java System, Tomcat 5/6, Jetty, Jrun, ServletExec
Databases	Oracle 8,9,10,11, DB/2 Universal, MySQL 4 and 5, MSSQL 2000, 2005
Portals	WebSphere Portal 5.1, 6, Weblogic Portal 9,10, Sun Portal 6.2, Liferay Portal 4, Vignette Portal 7

Hardware Requirements
A server with a contemporary CPU, 1G RAM and 10GB of disk storage will provide sufficient capacity for many installations. CPU and database power are the limiting components. Cluster configurations are recommended for scalability and availability.

Performance Considerations

ViewsFlash is designed from the ground up for maximum reliability and performance. It is written as a thread-safe, multithreaded servlet. For performance, computations are carried out entirely in main memory. All HTML is composed in main memory as well, and the connection to the visitor's browser maintains a keep-alive connection.

One main factor affecting performance is CPU speed and availability. On a single-processor 333 MHz Pentium-II PC, using Tomcat and Oracle 8, and allocating 256M of RAM to the VM running web applications, ViewsFlash runs consistently, without memory leaks, at 30 page views per second, or 33 ms. turnaround time, including the time required to send all information back to the client browser, on a 10 megabit local area network. CPU utilization averaged 80%.

The other main factor affecting performance is database latency and availability. ViewsFlash caches data in RAM whenever possible, minimizing database access. Slow networks and overworked databases can slow ViewsFlash performance significantly.

For heavy usage, the questionnaires in the intended application should be profiled using a stress testing tool to make sure that all components in the cluster scale to the expected volume.

For complete information on scalability, refer to **Architecture**.

Servlet Session Usage

ViewsFlash uses servlet sessions in these circumstances:
- to guarantee that during survey and poll design using the administrative UI, multiple page transactions (such as adding a question) are routed to the same servlet instance in a distributed environment.
- when requested explicitly in the Security page to identify respondents in multiple-page surveys.
- in JSR-168 portlets for caching.

Files Used

When not using a database, ViewsFlash writes all files to the "data" directory specified at installation time in the servlet parameters. Backing up this directory periodically will ensure against data loss. If the ViewsFlash data is being stored in a database, only those files marked * are kept in this directory.

System Files (one each)	File function	Backup created
viewsflash.cnf	setup configuration *	every change
viewsflash.log	system log *	
vwfpublish.cnf	publication scheduling *	every change

vwfspots.cnf	place configuration	every change
vwftemplates.cnf	style template configuration	every change
vwf2.0	2.X signature *	
viewsflash.browsers	browser versions *	
place Files		
spotname.spot	place configuration file	every change
Poll Files		
pollid.poll.cnf	poll configuration and tallies	every change, unless "backup all files" chosen in Setup
pollid.txt	poll data	was pollid.dat in 1.X
pollid.definition	poll definition	every change
Directories		
backup	file backups	
votedir	temporary files of multi page surveys	

Next: Staging

Staging, Import, Export

Overview

It is common to have a staging server where everything is tested before being installed on a production system. ViewsFlash Staging allows survey creation and testing on a staging system and controlled promotion of proven surveys to the production system. The staging system can also be used to promote surveys from a development system to a staging system.

When staging is enabled, the Publish page allows an administrator to lock and unlock a survey (preventing any change to the survey while locked), and to stage the survey and the styles and invite lists it uses which have changed at the same time. If using User Security, these capabilities are available to users who have Staging access right in that place. See **Publish**. Administrators also have a Staging page that allows them to manually stage all components.

How it works

When a survey, style or invite list is staged, one or more files are written to the export directory in the staging server's file system (data/export by default). From this directory, a system administrator or a content management system moves them to the destination server's import directory (data/import by default). Moving is preferable to copying, to avoid the possibility of staging something twice. The entire contents of the /export directory should be moved all at once. When the destination ViewsFlash detects these files, it uses them to update its configuration. The ViewsFlash administrator and the place e-mail recipient are notified by email whenever components are written to the export directory.

If a survey, place, style or invite list is removed in the staging server, an entry appears in the Staging page with Status set to "Removed", which allows the Administrator to remove those components on the destination system.

Staging requires that the staging and production servers use a database.

Staging from one server to a clustered server

The destination servers can either share a single import directory, or each server can have its own. If each server has its own, the staging procedure should move the files to be staged from the export directory to every server's import directory; however, everything works if only one of the clustered server's import directories receives the staging files (it is best to put it on all of them for redundancy).

Staging from clustered staging servers to clustered production servers

The staging procedure must move the files from each staging server's export directory to a corresponding import directory on the production server. When something is staged, the staged files will appear in only one of the server's export directories, so the staging procedure must be alert to both. The staging procedure can also move the files to every production server's import directory for redundancy.

Synchronization

In all the above configurations, the software synchronizes among different servers in the cluster using a database table named VWFSTAGING. In the staging server, this ensures that a component is only staged once, and components already staged are not staged again unnecessarily. On the destination server, this table ensures that components are not imported more than once. When a component is imported into one of the servers in a cluster, it notifies the other servers that the component has been modified, and the other servers upgrade to the latest version automatically.

Import and Export

Staging and Importing can be used to transfer surveys, styles and invite lists between different servers. Just export the survey, transfer the files from the export directory to the destination server's import directory, and import them. The two installations can even use different databases.

Configuration

Staging is configured using **servlet parameters**. If these parameters are omitted, only ViewsFlash administrators can export and import.

Servlet Parameters to use in the staging (originating) server.

- exportby controls who can export. Its value is either none, admin, or user, as follows:
  exportby=admin (the default) allows staging only by ViewsFlash administrators
  exportby=none disables exporting
  exportby=user allows staging by ViewsFlash administrators and users with appropriate rights. If using **User Security**,

only users who have the Staging access right can stage surveys.

- exportusersecurity=yes allows staging User Access Rights to the destination server. Without this parameter, User Access Rights are administered separately on the staging and the destination server.
- exportdir=full path to an existing directory. If omitted defaults to data/export.

Servlet Parameters to use in the production (destination) server:

- importby controls who can import. Its value is either none,admin, background, or a number, as follows:
  importby=admin (the default) allows the ViewsFlash Administrator to manually import from the Staging menu
  importby=none disables importing
  importby=background imports automatically every 5 minutes and allows manual import by the Administrator
  importby=NNNN imports automatically every NNNN seconds and allows manual import by the Administrator
- importdir=full path to an existing directory. If omitted defaults to data/import.

Staging without using the file system

There are situations where it is preferable to write and read the staging files from a database. To us this method, add the servlet parameter to both the source and the destination ViewsFlash instances:
stagingextension=DatabaseStaging

With this parameter, staging will read and write records to the database instead of the file system. But, of course, since the two databases are different, this is not sufficient.
An additional parameter is needed to tell the two instances to look for the records in the database of the other instance, not its own. Use the following parameters:

datasource.export=JNDIDataSourceName
or
datasource.import=JNDIDataSourceName

The first of these tells ViewsFlash to write staging records on a different database, specified in a different JNDI data source. The second, conversely, tells ViewsFlash to read staging records from a different database, specified in a different JNDI data source.

Note that all references to staging tables, on all data sources, are always prefixed by the dbtablenameprefix servlet parameter, if used.

Note that the exact syntax of the datasource parameter depends on the application server being used.
For most application servers, it is just the name of the JNDI data source:
datasource=NameOfTheJNDIDataSource
Tomcat, however, requires:
datasource=java:comp/env/NameOfTheJNDIDataSource

Next: Servlet Parameters

Servlet Parameter Reference

ViewsFlash is a web application. Some of its functionality is controlled by servlet parameters. These can be provided as init-param entries in its web.xml file, or in a viewsflash.properties file where they take the form name=value. The viewsflash.properties file is located at the location specified in web.xml, which is by default set to /etc/cogix/viewsflash.properties:

```
<init-param>
  <param-name>propertiesfile</param-name>
  <param-value>/etc/cogix/viewsflash.properties</param-value>
</init-param>
```

Servlet parameters can also be specified in the web.xml file, like this: <init-param>
  <param-name>name</param-name>
  <param-value>value</param-value>
</init-param>
When deploying ViewsFlash in a distributed configuration, it is preferable to put all parameters in web.xml.

To modify web.xml, unpack the ViewsFlash.war file, modify web.xml, and repack the ViewsFlash.war file. The "jar" Java utility can be used for this purpose.

This section is divided into Commonly Used and Rarely Used sections. It's safe to avoid the latter section unless there's some unusual trouble that cannot be fixed by modifying the commonly used parameters.

Commonly used parameters

Required	See **Installation**
data	Directory where to store ViewsFlash files
license	Licensing Key. Without one, ViewsFlash will expire after 30 days, and each screen will include an "Evaluation Copy" message. Note that ViewsFlash 5 requires new license keys; older license keys will not work. Contact Cogix for a new key.
When running with a database using a data source	See **Installation** and **Using a database**
database	One of these: Oracle, Oracle9, Oracle10, Oracle11, DB2, MSSQL, MySQL
datasource	The name of a JNDI data source.
When using application security	See **Application security**
appsecurity	Usually appsecurity=user
administrators	A comma-separated list of user IDs and roles; the roles are prefixed by an @ sign. If the logged in user ID matches any of those on the list, or if the logged in user is in any of the roles, they are granted the rights to access all portions of the ViewsFlash application. Example: peter.ustinov,rhonda.fleming,@viewsflashusers

Rarely used parameters

When running without a database	
backupdir	Directory where ViewsFlash stores *.bak files. For best performance, put it on the same drive as ViewsFlash, or turn this option off in the Administration page. Default is the data directory.
logpath	A directory for recording ViewsFlash logs. Default is the data directory.
txtpath	A directory for recording .txt files that capture votes or surveys, one entry per line. Default is the data directory.
votedir	A directory for storing temporary information during multi-page voting. Highly recommended under heavy loads. Default is the data directory.

logrequests	Set to 1 to log every http request to the ViewsFlash log. Set to 2 to also log every database access. Useful for debugging, but log can get very large. Default is 0.
strictinit	If set to 1, all servlet parameters are checked for correct spelling. If any are misspelled, the servlet will refuse to run with an UnavailableException. Default is 0. When using this, if you want to disable other servlet parameters, prefix them with a ~ and strictinit will regard them as valid, but will ignore them.
confirmwrite	If set to 1, whenever an .html file is written to the file system (such as a poll fragment), ViewsFlash will make sure the write succeeded.
Other application security techniques	
admintimeoutminutes	Passwords will need to be reentered after the given minutes. This guards against people who walk away from their computers while using ViewsFlash. Default is 60.
adminport	Port for administrative functions; should be behind a firewall for maximum protection. Default is no protection - any administration request can come on any port.
apiport	Port for inter-application communication. When used, the Web Services API commands are only available on this port, which should be behind a firewall for maximum protection. By default, this port is set to a non-existant port.
apiattribute	Attribute for inter-application communication. Web Services API commands issued in JSP pages with a RequestDispatcher must set this attribute in the HttpRequest to a Long value with the current time. Requests that are missing this attribute will be denied. JSP pages provided by Cogix, such as i18n.jsp, expect this parameter to be set to: apiatttribute=com.cogix.vwf.api By default, this parameter is not set, and using the Web Services API from JSP pages is disabled. See **How to use the Web Services API**.
securesurveyport	For questionnaires that specify use SSL, if a port other than the default 443 is desired, use this parameter
normalsurveyport	To use a port other than 80 for questionnaire URLs, use this parameter
adminpw	When not using **Container Security**, this parameter provides password to restrict access to the administrative functions. Each place can also have its own password.
extravulnerabilitychecks	When set to true, application page forms include an additional security token to validate that each form submitted is unique. Note that when using this parameter users cannot use the technique of opening up two application windows to work on different parts of the application at the same time.
Web Application parameters	
webappname	When installed as a web application, normally ViewsFlash can guess its web application name automatically. If it does not, use this parameter to set it. /ViewsFlash is the normal name, except in WebSphere Portal.
appwebroot	Normally, ViewsFlash will automatically find the "document root" by discovering the location that corresponds to the URI "/". If this fails, set this parameter to that location.
usedefaultcharacterencoding	A few application servers (like Dynamo 7) interpret charset when receiving requests. If a questionnaire's Language page does not work, set this parameter to true to prevent ViewsFlash from decoding request parameters and form submissions.
Setup page parameters can be enterd as servlet parameters rather than entering them through the **Setup** page.	
admin.name	ViewsFlash Administrator name
admn.emailto	His or her email address for receiving ViewsFlash emails
admin.emailfrom	His or her from email address to identify the messages
admin.emailreplyto	His or her reply-to address for receiving replies
admin.smtp.host	Host name for the smtp server that handles outgoing email
admin.smtp.username	If the smtp server requires authentication, provide the username
admin.smtp.password	and the password

mail.	In addition, any parameters that start with "mail." will be used to initialize the JavaMail session for the smtp server. Example: mail.smpt.auth can be used when the mail server requires authentication.
admin.viewsflashapplicationurl	The URL for the ViewsFlash application. Specify only if the default value is incorrect. Example: http://www.yourcomany.com/ViewsFlash.
admin.viewsflashservleturl	The URL for the ViewsFlash servlet. Specify only if the default value is incorrect. This can be useful when the ViewsFlash application runs behind a proxy server that translates URLs to internal URLs and the public servlet URL needs to be different. This option requires the showadminservleturl=true servlet parameter.
admin.viewsflashsecureservleturl	The URL for the secure ViewsFlash servlet. Specify only if the default value is incorrect. This option requires the showadminservleturl=true servlet parameter.
showadminservleturl	Set this parameter to true to make additional fields in the Administration visible and editable. Use this option when the ViewsFlash URLs are incorrect. Normally, quesionnaire URLs are the same as the application URL. When this parameter is used, questionnaire URLs are generated from the settings in the Administration page instead. When using a reverse proxy server, this option allows for the ViewsFlash application to work at a different URL and port.
usingsslproxy	Setting this parameter to true prevents the generation of certain HTTP headers that interfere with downloading files and reports in IE6.
admin.pollroot	When not using a database, the directory where HTML files should be written. Specify only if the default value is incorrect. Example: /opt/virtual/yourcompany/ViewsFlash
admin.usersendemail	Allows sending email. Default true. Set to false to disable.
admin.emailerroralerts	Notify administrator of error conditions by email. Default true. Set to false to disable.
admin.cookiedomain	Cookie domain for ViewsFlash cookies.
Environmental parameters	
servletserver	Allows ViewsFlash to be named something other than /servlet/viewsflash. For example, if set to /cgi-bin/Polling, then you can use http://mycompany.com/cgi-bin/Polling/viewsflash.
servletpath	If the application server does not support getServletPath(), or if you are using 'servletserver' above, set this parameter to "viewsflash". (Very rare).
servletrequestinputstream	ViewsFlash versions before 3.0 parsed the servlet input stream, rather than using getParameterNames(). Set this parameter to "1" to revert to parsing the servlet input stream instead.
realpath	If the application server does not support ServletContext.getRealPath("/") in the HttpServlet.init() method, set this parameter to the equivalent of the hosting web server's root directory. Omit the trailing / or \.
referer	If the application server does not set the usual REFERER header, use this parameter to specify the name of the header that serves this purpose. (Very rare).
hostipaddress	Use this parameter to tell ViewsFlash its IP address. ViewsFlash normally determines the IP address of its machine by using getLocalHost(). Sometimes this returns 127.0.0.1, which is not useful. This is also useful when the machine has more than one IP address.
Database parameters	
notificationinterval	In a distributed database configuration, each instance checks for new events at this interval, in milliseconds. Default is 5,000. If not running in a distributed database configuration, you can set this value quite large (1000000 - one million) to reduce the load on the database a little bit.
instanceid	In a distributed database configuration, this parameter is used to identify each instance. If omitted, a random number is assigned. These instance ID's are used in the application logs to identify the instance that is writing to the log. You can assign a number on your own to make the log easier to read; note that if you do, it **must be unique** for each instance. **Failure to make instanceid unique can result in inconsistent instances**; for example, one instance could think that a poll is open, while another one doesn't.
storeflushtime	Number of milliseconds before saving a poll's tallied results to database. Default is 000 (no delay).

refreshresultstime	In a distributed database configuration, this parameter indicates how often to retrieve the current live results from the database. The default value is 3,000 milliseconds. A very small value will increase system load significantly.
dbmaxvarchar	The largest size of a VARCHAR type allowed. Default is 2000. If setting dbcharwidthmultiplier greater than 1, we recommend changing this to 4000.
dbcharwidthmultiplier	The maximum number of characters required to express one Unicode character. If the database will only store ISO-8859-1 characters, this value is 1 (the default). If the database must store other kinds of characters, we recommend setting this parameter to 3 to support the UTF8 character set and setting dbmaxvarchar (above) to 4000.
dbtablenameprefix	If present, this string will be prepended to the name of every table used by ViewsFlash. For example, COGIX. will use tables such as COGIX.VWFOBJECTS. Note that the trailing period must be entered explicitly.
dbcreatetablespacestatement	If present, this string must contain a CREATE TABLESPACE statement that is exectuted before executing a questionnaire's CREATE TABLE statement. It can contain the string %1, which is replaced by a 7 digit random number. Examples: dbcreatetablespacestatement=CREATE TABLESPACE A%1 in MYPARTITION USING...(and other parameters, all on one line) will produce CREATE TABLESPACE A12345 IN MYPARTITION USING ... Do not use this parameter if using the same partition for all data tables; precreate the table space manually. In either case, use dbtablenamesuffix as described below to assign the table to a tablespace.
dbtablenamesuffix	If present, this clause will be inserted after the CREATE TABLE statement used to create a questionnaire's own table. Examples: dbtablenamesuffix=IN VFD.A%1 will produce CREATE TABLE xxxx (...) IN VFD.A12345 dbtablenamesuffix=IN MYTABLESPACE will produce CREATE TABLE xxxx (...) IN MYTABLESPACE
database	Name of the Java class to use for database storage. Example: Oracle, Oracle9, Oracle10, MSSQL, MySQL, DB2. The actual class name is created by prepending "com.cogix.vwf" if not present, and appending "NamedDataSource" if the name does not include "DataSource". So, Oracle becomes com.cogix.vwf.OracleNamedDataSource.
databaseextension	Name of the Java class to use for writing responses to a single, normalized table. Example is OracleDatabaseExtension. By default, this parameter is assumed to be XXXDataBaseExtension, where XXX is Oracle, MSSQL, etc. Using databaseextension= (without an argument) will disable the **Normalized Database** functionality. Note that the actual class name is com.cogix.vwf.XXXDataBaseExtension, but com.cogix.vwf can safely be omitted.
normalizedtype	Some normalized database drivers use this parameter to specify what type to use for storing the data element, which is normally set to VARCHAR.
Saving data parameters	
save.text	Set to false to prohibit saving questionnaire data in the file system
save.table	Set to false to prohibit saving questionnaire data in its own database table. Use this when the database user should not be granted the right to create, alter and drop tables.
save.normalized	Set to false to prohibit saving questionnaire data in the common, normalized table VWFDATA.
upload.directory	Points to a directory where the ViewsFlash application server has been granted file creation, directory creation, and read, write and delete access rights. In a single-server deployment, any directory can be used. In a clustered deployment, this directory must reside on a shared volume, so that attachments gathered from all servers are stored in the same directory and can be accessed by all nodes in the cluster. If this parameter is omitted, a subdirectory of the data directory is used.
upload.maxsize	Specifies the maximum size of an uploaded document. The default value is only 1,000,000.

Database error recovery parameters	
dbretryinterval	After a database failure, wait this number of milliseconds before retrying. Default is 30,000 milliseconds.
dbretries	After a database failure, retry this number of times. Each retry generates an error log entry and an administrator e-mail. Defaults is 100.
dbignorevendorcodes	The dbignorevendorcodes parameter suppresses database retries for the specified database vendor error codes. Enter this parameter as a list of integers, separated by commas, **omitting** leading zeroes. For example, dbignorevendorcodes=1401 will disable retries for Oracle error code ORA-01401, caused by attempting to store a non-numeric value in a numeric field.
dbignoresqlstatecodes	The dbignoresqlstatecodes parameter suppresses database retries for the specified 5 character standard XOpen SQLState codes. Enter this parameter as a list of 5 character SQLState codes, separated by commas, including leading zeroes. For example, dbignoresqlstatecodes=23000 will disable retries for XOpen SQLState code 23000 (Integrity constraint violation), a more generic form of ORA-01401.
Java Extensibility more info	
publishnotifierextension	Name of the Java class to invoke after a publishing change. Example is com.cogix.vwf.PublishNotifierExtension
votenotifierextension	Name of the Java class to invoke after a valid vote is cast. Example is com.cogix.vwf.VoteNotifierExtension
requestnotifierextension	Name of the Java class to invoke after a request is scanned. Example is com.cogix.vwf.atgRequestNotifier This requires servlet API 2.2. If application fails to boot with a NoSuchMethod exception, try disabling this parameter to nothing with: requestnotifierextension=
webcasteventextension	Name of the Java class to invoke when a webcast poll or survey event happens. Example is com.cogix.vwf.WebCastEventExtension.
Miscellaneous	
errorinterval	Controls the interval, in seconds, used to calculate whether a repetitive log entry **should be suppressed**.
writehtmltodisk	When using a database, setting this option to 1 enables options in the place Settings page that allow writing HTML to disk.
webcast	Setting this option to 1 enables the **Webcast** features.
datareviewapi=true	Enables the **data review and modification API**.
appcharset=UTF-8	Defines the application's default character set. Useful when the native locale is not English. For example, GB2312 is useful in mainland China. appcharset=UTF-8 is the default
dbdatabaseconnectionclass	Deprecated: use database instead
dbservername	Deprecated: use datasource instead
polladjustment	Set to true to enable the legacy Options/Bulk Vote, Remove Entries, and Set Rank polling functions which are disabled by default as of ViewsFlash 6.
Staging parameters	
exportby,exportusersecurity, exportdir,importby,importdir	See **Staging**

Next: Production

Production

After a questionnaire is created, the URL on its publish page should be used to invite respondents to it in one of these ways:

- As a link, such as **Tell us what you think**, with the questionnaire's URL as the link, eg
http://yourcompany.com/ViewsFlash/servlet/viewsflash?cmd=page&pollid=Examples!Service
The exact URL is given on the Setup / Publish page.

- As a link in an e-mail, using the **Invite** capabilities.

- By displaying the questionnaire in an iFrame, using the questionnaire's URL from the Setup / Publish page as the frame's URL.

- By displaying the questionnaire as a pop-up window or using dynamic HTML, and fetching the questionnaire using the URL on the Setup / Publish page.

- By embedding the questionnaire in a JSP page, using the techniques described in **Embedding questionnaires in JSP pages**.

- By embedding the questionnaire as a portlet in a portal, using the techniques described in the appropriate portal guide at **www.cogix.com** under JSR 168 portals.

The rest of this section describes the Publishing System API, which describes how to use ViewsFlash as a Web Service for publishing and rotating single page polls automatically. This functionality has been superseded by JSP pages and portals but is included here for older systems.

Production Cycle explains the steps required from the production team for creating a new place and for embedding polls in web pages. **Web Services API** details how to tailor such a system to dynamically compose web pages incorporating ViewsFlash poll and result displays.

Next: Production Cycle

The Production Cycle

After the ViewsFlash software is installed, here's what you will need to do to produce questionnaires.

Standard uestionnaires with default settings are dynamically composed by ViewsFlash, and are simply referred to by their URL, which is shown in a very visible box at the bottom of the the Publish page.

All you need to do is to use the URL in the Publish page to direct visitors to the questionnaire:

If you use Invitation lists, emails use this this URL.

Standard questionnaires can easily be **embedded in a JSP page**.

In Portals, there is no production cycle. See the User Guide for your portal at **www.cogix.com** under JSR 168 portlet.

An **embedded** poll is one where the voting form and the results display are embedded inside a containing web page, thus requiring additional setup. If the site is using a content management system, each page containing a place is modified to include the appropriate publishing system tags in the spot reserved for the place. These tags ask ViewsFlash for the HTML fragments to use.

If you don't need to embed polls, skip the rest of this section.

In addition to **using a JSP page** (the easiest method) or an IFrame, two methods for embedding polls and results into a web site are described here. IFrames are found in the Support section of **www.cogix.com**.

If the site uses a Content Management System for dynamically constructing web pages, the ViewsFlash Web Services API provides the appropriate HTML fragments as needed. If a site does not use a Content Management System and uses static HTML, ViewsFlash can write fragments or even modify the pages where polls are embedded if necessary. Here are additional considerations.

Feature	Content Management System	Shared Files
ViewsFlash provides Poll voting form HTML fragments	via an HTTP request	by writing them to shared file system
Containing pages must have	tag to invoke WPS	<vwf=embed> and <vwf=endembed> tags
File systems on two servers	completely separate	must be shared
Display poll or results, depending on prior voting	easy	requires two complete pages
Results can be live	yes	yes
Results can be cached	yes	no
Settings to use in place Setup	Use "a complete web page"	Use "An HTML fragment"
See also	**Web Services API**	

Producing whole-page polls and surveys

1. The Designer should create a set of Style Templates to give the questionnaires the look and feel of your site. Start by copying the Standard_ template, and embellish it as appropriate, adding headers, banners, etc.. Similarly, use a copy of the Results template as a starting point for your own Response Template. Likewise, start with the MenuArchive or the ListArchive templates and add to them accordingly.

Once these templates are created, you'll be able to reuse them over and over at many places.

2. The Editor, for each poll or survey, creates a place and chooses the appropriate templates. Settings should specify "whole

page." After going through the normal design process, the URL is found in the Publish screen. The Designer uses that URL to link to the poll page or to publicize the survey by e-mail to bring people to the polling page. The page can also be e-mailed directly to people using HTML-capable e-mail software.

Producing embedded polls by hand

This is by far the easiest way to create embedded polls. The Editor creates a place, using an embedded style template, and publishes a poll. The Designer goes to the Publish page, clicks on the link below the Submit button with the Poll URL, and copies the entire contents of the window that pops up to the proper spot in the page where the poll should appear. If the template included JavaScript for **showing the results in a pop-up window**, you're done.

Producing embedded polls and surveys with a content management system.

See the **Automated Poll Rotation.**

Producing embedded polls and surveys without a content management system.

In the absence of a content management system, ViewsFlash writes HTML fragments on the file system. There are two ways to embed them in the containing pages: server-side includes and modifying the containing pages. Both methods require that the main web server and the ViewsFlash web server either be the same web server, or that they share a common file system.

1. The Administrator configures the web servers so that they have a shared file system. In the Administration screen, the URL and the location of the main web server's document root are provided, as seen from the ViewsFlash machine.

2. The Designer develops a set Style Templates, using with the HTML Fragment Poll and Result templates and the List and Menu Archive templates. All templates should omit <head> and <body> tags, as they will be used to generate HTML that will be dropped into the middle of a web page by the publishing systems. These templates can be reused over and over.

3. The Editor, for each poll or survey, creates a place and chooses the appropriate templates. The Settings to use depend on whether server side includes are being used or not.

In Step 2, choose "An HTML fragment", and enter the name of the page on the main web server, such as "News/Domestic.html", or "Politics.html". Choose the appropriate radio button depending on whether you are using server-side includes or not.

In Step 4, choose "An HTML fragment". ViewsFlash will use the settings in Step 2 to retrieve a copy of the containing page, embed the results between the <vwf=embed> and <vwf=endembed> tags, and display the resulting page in response after recording the visitor's vote.

The Designer includes the following HTML in the place on the containing web page where the poll will appear:

<vwf=embed>
<vwf=endembed>

If using server-side includes to incorporate the HTML fragment into the containing page, it appears in between these tags:

<vwf=embed>
<--#include virtual="/viewsflashserver/htdocs/pollhtml/pollingplace.html" -->
<vwf=endembed>

Note that in the server-side include, the HTML fragments are being included from a directory that the ViewsFlash web server writes to. If you are not using server-side includes, then the ViewsFlash server will modify the containing web page whenever the scheduled poll changes, but not in between.

After this, the Editor can design additional polls in the same web page, without modifying either the containing web page or the Style templates.

If you plan on using a Content Management System, read about the **Web Services API**. Otherwise, you can also skip right to **Style Templates**.

Next: How to use the Web Services API

How to use the Web Services API

The Web Services API provides applications the ability to interact with ViewsFlash. They can be other Java web applications in the same Java Virtual Machine, or any application on any server using the HTTP query protocol. The API calls require composing a query string and invoking the ViewsFlash application using a RequestDispatcher or an HTTP query. All Web Services API requests must pass appropriate security checks. The sections below specify in detail how to construct and send these queries.

Using the RequestDispatcher include() method

The information in this section is appropriate not only for the Web Services API, but for any application that wants to include the ViewsFlash application. These include: wrapping the viewsflash servlet inside another servlet to provide custom security and **embedding questionnaires in JSP pages**.

This method is available when the ViewsFlash application is deployed in the same Java Virtual Machine as the Java program that is using the Web Services API. If this is a JSP page, and the page is not in the ViewsFlash context, it may be ncessary to enable cross-context access in the Application Server. In Tomcat, in particular, this is done with a Context element; see the ViewsFlash/META-INF/ViewsFlash.xml file.

As of release 5.6, access to the Web Services API from JSP pages must be enabled explicitly with the "apiattribute" **servlet parameter**:
apiattribute=com.cogix.vwf.api

The JSP page must set this request attribute to the current time in milliseconds to authenticate itself.

The following example uses the API to request a list of all questionnaires in a Place. It uses a query string cmd=getpollnames&status=open&spotname=NAME

```
String place = "Examples"; // the name of the Place
String apiparams = "cmd=getpollnames&status=open&spotname=" + place ; //
cmd=getpollnames&spotname=Examples

String resultsattribute = "com.cogix.vwf.response"; // the name of a request attribute
String directtheresponse = "&outputtorequestattribute="+ resultsattribute; // instructs the API
to deliver its response in a request attribute

// Get the Context in a JSP page:
ServletContext otc = pageContext.getServletContext(); // use this when the JSP page lives
inside the ViewsFlash application
// ServletContext otc = application.getContext("/ViewsFlash"); // use a cross-context reference
only when this JSP page lives in a context outside /ViewsFlash

// If not in a JSP page, but rather in a web application,
use  getContext()  or  getServletContext().getContext("/ViewsFlash")  instead.

// Construct the query string
String query = apiparams + directtheresponse; // do not use a leading ?
request.setAttribute("com.cogix.vwf.onlyquery", query); // Place the query string in this
attribute to ignore all request parameters available using request.getParameter()
request.setAttribute("com.cogix.vwf.api", new Long(System.currentTimeMillis())); // set this
attribute to the current time to establish security credentials

// Find the request dispatcher
String servletpath = "/servlet/viewsflash"; // servlet that responds to Web Services API
requests
RequestDispatcher rd = otc.getRequestDispatcher (servletpath);

// include
rd.include(request, response); // pass the request to the servlet

// work with the response
String respond = (String) request.getAttribute(resultsattribute); // retrieve the Web Services
API response to the request
if ( respond != null ) // if response were null, there has been a malfunction. Most likely
cause: /servlet/viewsflash not found in the Context, or authentication failure
    out.print ( respond ); // this example would display a list of all the questionnaire ID's
in the Place
```

Frequently asked questions

How do you pass parameters to the ViewsFlash servlets?
Many application servers allow adding "?name=value" parameters to the servlet path passed to the RequestDispatcher.

However, when processing the include(), ViewsFlash will process parameters from both this query string and request parameters that were available to the JSP page. This can lead to paramter conflict or duplication. Some application servers ignore this query string. To avoid this problems, it is better to place the query string in one of these HttpRequest attributes:

Use "com.cogix.vwf.onlyquery" to ignore the original request query and all request parameters and use the parameters in this query string
Use "com.cogix.vwf.onlyqueryandparams" to use the available request parameters as well as the parameters in this query string

How do you find the /servlet/viewsflash servlet?
Use getContext ("/ViewsFlash") and getDispatcher ("/servlet/viewsflash"), as shown in the **example**. It may be necessary to enable "cross-context" processing, as in Tomcat's ViewsFlash.xml (or server.xml).
If working inside a web application in Java code (instead of a JSP), then find the context and servlet using getServletContext() or getServletContext().getContext("/ViewsFlash") instead.

Where does the response go?
Normally, the viewsflash servlet writes to the response output stream. This can easily interfere with the application server's output stream, however. To solve this problem, the following parameters can be used with any query string. The most useful one is outputtorequestattribute. See **Query String Reference** below.

Can I use RequestDispatcher.forward()?
Yes. Use the code provided above under **RequestDispatcher** and use forward instead of include. Instead of outputtorequestattribute, use the appropriate options from **Query String Reference**. Of course, after the forward, no other code will be executed.

Query String Reference

Command	Purpose
nooutputstream=1	Uses an already opened output stream, such as the JSPPrintWriter from a JSP page that includes output from the ViewsFlash application.
nooutputclose=1	Do not close the response output stream.
nooutputheaders=1	Do not set any headers in the output stream.
nooutputclose-1 and nooutputheaders=1 are often used together; they help make sure that the output stream is not closed prematurely and that response headers are not sent too soon.	
outputtorequestattribute=attributename	Do not write to the output stream at all; write the content to the named request attribute using HttpServletRequest.setAttribute (). See also **how to pass parameters**.
outputtosessionattribute=attributename	Do not write to the output stream at all; write the content to the named session attribute using HttpSession.setAttribute ()
rediraftersetattribute=url	Redirect to an absolute URL afterwards
rediraftersetattribute=*?p=u	Redirect to the referring URL, set command tail parameter p to u. (See the Security page for the syntax).
outputtosessionattribute and rediraftersetattribute are often used together. The redirect goes to a dynamic page which retrieves the servlet's output from the specified session attribute.	
stream=0 and &stream=1	If used, these must be at the very beginning of the query string. stream=0 forces reading the request parameters using getParameterNames(); stream=1 forces reading request parameters directly off the requests' input stream. The servlet parameter "servletrequestinputstream" controls this behavior; this provides a way to override it if necessary.

Using HTTP queries

An HTTP query can be constructed by any application in any language, including Java, ASP, PHP, etc. The query is sent to the ViewsFlash application over the network using the standard HTTP protocol. ViewsFlash receives the query and sends its response in the appropriate content type, usually text/html. The application then interprets the query and uses the information as appropriate. Web Services API calls must use the TCP port specified in **Application Security**.

The remainder of this section provides sample code for popular languages. Port 8888 is used arbitrarily; there is no default port for the Web Services API. The example is the same as the one used in **Using RequestDispatcher** above.

```
String place = "Examples"; // the name of the Place
String apiparams = "cmd=getpollnames&status=open&spotname=" + place ; //
cmd=getpollnames&spotname=Examples
```

Using AJAX

```
// Use your AJAX toolkit to set xmlHttp to the XMLHttpRequest object
var url = "http://yourserver.com:8888/ViewsFlash/servlet/viewsflash?
cmd=getpollnames&status=open&spotname=Examples";
xmlHttp.open("GET",url,true);
xmlHttp.send(null);
// Response handler will find response in xmlHttp.responseText
```

Using VBScript in an ASP page

```
Set xmlhttp = Server.CreateObject("Microsoft.XMLHTTP")
xmlhttp.Open "GET", "http://yourserver.com:8888/ViewsFlash/servlet/viewsflash", false
xmlhttp.Send "cmd=getpollnames&status=open&spotname=Examples"
// Response is in xmlhttp.responseText
```

Using Java in a different VM or server

```
String response = httpget ("http://yourserver.com:8888/ViewsFlash/servlet/viewsflash?
cmd=getpollnames&status=open&spotname=Examples");

String httpget ( String fn ) {
try {
 StringBuffer sb = new StringBuffer (500);
 URL href = new URL (fn);
 HttpURLConnection hc = (HttpURLConnection) href.openConnection();
 hc.setRequestMethod ("GET");
 hc.connect();

 InputStream ins = hc.getInputStream();
 int kar; char ch;
 while ( (kar = ins.read () ) != -1 )
   ch = (char) kar; sb.append(ch);
 ins.close();

 hc.disconnect();
 return new String (sb);
 }
catch (Exception e) {
 return "\r\n<!-- Oops=> " + e.toString() + "-->";
 }
}
```

Next: Web Services API Reference

Web Services API Reference

You can also use ViewsFlash as a component in a web application by using its API commands. This section provides a reference to each command and what it does.

How to issue commands
ViewsFlash receives commands through its service() method as a URL. Any method that results in this protocol is compatible with ViewsFlash. These include any way to issue an http get command from another application. For example, you can use Java's HttpURLConnection class in JSP pages, or the Microsoft.XMLHTTP class in Windows ASP pages.

You can also, from a Java environment that provides appropriate support, invoke the service() method of ViewsFlash directly, provided the appropriate conventions are followed. See **How to use the Web Services API**.

In all cases, you send ViewsFlash a query string, just like you would enter into your web browser, such as:
cmd=getpoll&spotname=Default

Responses and errors
When ViewsFlash receives a request, it returns a text/html string with the requested information. If the request is in error, ViewsFlash will return a string of the form:
<viewsflash [/NG] ... > with additional error information inside. You should parse for "<viewsflash" and log the message and take whatever action is appropriate for a failed request. Because the error message is enclosed in <>, if you let it pass through to a browser, it will be invisible but can be examined with a View Source. You must test for this string and take appropriate action.

Command Reference

Command, parameters	Operation
cmd=getpoll &spotname=SSS (required)	Fetches the voting form HTML of the currently scheduled poll in place SSS.
&pollid=pollingplace!poll	If present, fetch poll pollingplace!poll rather than the currently scheduled poll. Useful for previewing.
&style=TTT	If present, use this Poll Style template instead of the default.
&stylesfx=SSS	If present, append this to the name of the style template to use.
&pollidsfx=PPP	If present, present the voting form using the question and answer text from a poll named 'pollid'+'pollidsfx', but refer to the underlying 'pollid'.

Command, parameters	Operation
cmd=getresults &spotname=SSS (required)	Fetches the current (live) results for the currently scheduled poll in place SSS.
&style=TTT	If present, use this Response Style template instead of the default.
&results=r.html	If present, use this page, relative to the web root directory, to compose results.
&latest=1	If present, compose results of the current poll or, if closed, the most recent poll.
&pollid=pollingplace!poll	If present, compose results for this specific poll. You must also specify spotname.
&stylesfx=SSS	If present, append this to the name of the style template to use.
&pollidsfx=PPP	If present, present vote results using question and answer text from a poll named 'pollid'+'pollidsfx', but get the results from underlying poll 'pollid'.

Command, parameters	Operation
cmd=geteither	If the visitor has not voted yet in the current poll, fetches the voting form HTML of the currently

&spotname=SSS (required)	scheduled poll in place SSS. If there is no current poll, returns the default value specified in the place settings page. If the visitor has already voted in the current poll, fetches the current (live) results for it.

Command, parameters	Operation
cmd=ifvoted &spotname=SSS (required)	If the visitor has not voted yet in the current poll, returns "0". If the visitor has already voted, returns "1". If there is no current poll, returns "2". If the visitor is not authorized to vote, returns "3".
&verbose=1	Besides the single digit described above, returns a single '|' character followed by the appropriate error text from the "Display the following response page" fields in the Security page, which are usually set to begin with an equal sign when using this option, to indicate a literal.

Command, parameters	Operation
cmd=getpollnames &spotname=SSS	Fetches a list of poll IDs in place SSS, separated by a | character. Returns "-1" if there is nothing to return.
&status=archived or current or scheduled or possible or latent or all (optional, choose one; default is current)	current lists the single poll that is active, if any archived lists questionnaires in the Archived Polls page scheduled lists questionnaires with explicit start and end dates possible lists everything on the place page (i.e., everything not archived) latent lists questionnaires that are open now or are scheduled to be open in the future
&match=mask	If present, this mask is the same as a file system wildcard, e.g., P?21?,* where a ? matches one character and * matches one or more characters. Only polls that match the mask will be listed.
&sort=alpha	Without this argument, polls are listed in the same order as the Place page; when this argument is present, polls are sorted alphabetically, with case sensitivity.
&title=1	If this argument is present, each poll name returned is followed by 'title= and the Title of the poll
&smp=1	If this argument is present, each poll name returned is followed by 'smp=s or 'smp=m depending on whether the survey is a single page or multiple pages, respectively.
&isscheduled=1	If this argument is present, and the place schedules only one poll at a time, a * precedes either the pollname of the currently open poll, eg., *Go!now, or *-1 if there is no currently open poll.

Command, parameters	Operation
cmd=checkpoll &pollname=PPPPPP &spotname=SSSSS	Checks place SSSSS to check the status of poll PPPPPP. Returns the following: 0 - Poll doesn't exist. You **must** create it before using it! 1 - Poll open for voting 2 - Poll defined, scheduled to open in the future 3 - Poll closed and archived, voting no longer allowed 4 - Poll defined, but not scheduled Illegal opening/closing times will return the standard error return <viewsflash [/NG] ... > This code can be used in cmd=getresults by using the [/pollstatus,code,message] tag.

Command, parameters	Operation
cmd=getpubdate &spotname=SSS	Fetches the publication date of polls in place SSS, if available, or an invisible error message otherwise. If no spotname is used, returns the current time.
&pollid=pollingplace!poll	If not specified, looks up the currently scheduled poll. If specified, looks up pollingplace!poll .
&date=start or end	

(optional; default is end)	Looks up the start or end date
&format=a or b (optional; default is b)	If a, returns in format "8/9/00 11:00 AM". If b, returns a long integer representing Unix time in milliseconds since 1/1/1970, such as 965844000000.

Command, parameters	Operation
cmd=gettime	Returns the number of 10-second ticks since approximately 6/30/99, with leading zeroes. This is the same time convention used for the cookies thrown when you use the History parameter.

Command, parameters	Operation
cmd=getspotnames	Fetches a comma-delimited list of place names.

Command, parameters	Operation
cmd=multisample	Retrieves a javascript fragment for opening a window and conducting a survey in it. The survey shown is picked from the list of surveys identified by id= parameters. Each of the surveys must be set up for **Random Sampling** and must be currently active to be included; if not, it will be ignored.
id=pollid	The id of a survey, e.g., pollingplacename!pollname
id=pollid	repeat one pair of these for every survey
history=cookiename	The name of a cookie that will track the sampling history for all these surveys. Do not use the history parameter when using this command.
multiple=yes	If yes, the visitor may be presented with more than one of the surveys on the list. If no, the visitor will be presented with at most one survey.

Command, parameters	Operation
cmd=provision	Creates a Place and assigns a user access rights to that place. The operation returns "OK" if successful or an error message otherwise.
addname=SSSS	The name of the place to be created
users=UUUU	The user IDs of the users who will be granted all rights to the Place
email=xxxx@xxx.xxx	(Optional) The email address where a message will be sent including the URL of the newly created Place.

Command, parameters	Operation
cmd=provision2	Gets or sets access rights in a Place.The operation returns "OK" if successful or an error message otherwise.
spotname=SSSS	The name of the place
op=getrights	Use op=getrights to receive the rights in a place. The operation returns a list of the users and their rights, like this: createusers=*,tom|analysisusers=peter|viewdatausers=|stagingusers=@publishers|
op=setrights	Use op=setrights to set the rights in a place. Add the following parameters. If a parameter is omitted, that right will not be granted to anyone. createusers=*,tom&analysisusers=peter&viewdatausers=&stagingusers=@publishers
email=xxxx@xxx.xxx	(Optional) The email address to send a message alerting that Place rights have been set.

Command, parameters	Operation
cmd=provision3	Creates a questionnaire by copying from another

newpollid=SSSS!PPPP	SSSS is the name of the place, PPPP is the name of the questionnaire to be created. It must not exist. Required.
pollid=SSSS!PPPP	SSSS is the name of the place, PPPP is the name of the questionnaire to be copied. It must exist already. Required.
publish=1	If this parameter is used, the questionnaire will be published immediately, with a starting date of Now and no ending date.

Next: Embedding questionnaires in JSP pages

Embedding questionnaires in JSP pages

To embed a questionnaire in a JSP page, first go to this URL, on your ViewsFlash installation:
.../ViewsFlash/embedexample.jsp
or /ViewsFlash/embedexample.jsp?questionnaire=XXX
where XXX is of the form Examples!Service for a questionnaire named Service in place Examples

Retrieve the file embedexample.jsp from the ViewsFlash.war file or the file system and follow the detailed instructions there.

The technique used is described in **Using the Web Services API in a JSP page**.

Next: Data Review and Modification

Data Review and Modification API

To enable this API, the system administrator must add datareviewapi=true to the **servlet parameters** and restart the ViewsFlash application.

See also **Revising Entries** for an easier way to allow a visitor to enter data and revise it later, as in a Profile, for example.

In certain applications, it may be desirable to review or modify the contents of a submitted survey after it has been completed. The following queries allow this. These operations require that the data be stored in its own table. They do not work with data saved in a text file or in the Normalized database format in the VWFDATA table. The queries do not work on partially completed surveys, since they are not written to the database until the survey is completed or the survey is closed.

In the table below, "admin only" means that this option is only available when using application security, and the logged in user is an administrator with data review access rights, and that option is selected in the Settings / Security page.

Data Review

Command, parameters	Operation
cmd=reviewdata &pollid=pollingplace!poll	Retrieves the survey form and fills it with data on a single page. The previous, next, save and submit buttons are not displayed. The form cannot be submitted.
&review_fieldname=NNNN	(adminonly) The name of the database field to use to select a record. If not provided, AUTHENTICATEDUSERID is used.
&review_uniqueid=UUU	(adminonly) The value to search for in review_fieldname. If multiple records meet the search, the most recent one will be displayed.
&style=TTT	If present, use this Poll Style template instead of the default. This is particularly useful to change the presentation to include a Print button and additional text on why this is a review and not an editable form.

Data Modification

Command, parameters	Operation
cmd=showupdata &pollid=pollingplace!poll	Retrieves the survey form and fills it with data on a single page. Only the submit buttons is displayed.
&review_fieldname	((adminonly)) The name of the database field to use to select a record. If not provided, AUTHENTICATEDUSERID is used.
&review_uniqueid	(adminonly)The value to search for in review_fieldname. If multiple records meet the search, the most recent one will be displayed.
&review_redirect	(optional) If present, redirects to this URL after the form is updated. The url must be complete, including the leading http://. If not present, the updated form is shown in the same format as reviewdata.
&style=TTT	(optional) If present, use this Poll Style template instead of the default. This is particularly useful to change the presentation to include a Cancel button or other navigation.

If this page is submitted, the unique record identified by review_fieldname and review_uniqueid will be modified. Any values that have changed will be updated in the database. For security, it is not possible to issue a query that modifies data directly, but rather it is necessary to go through this URL.

How to use the data review and modification URLs

A common usage is to use the Response page's redirect field to redirect to these URLs. For example, redirect to
/ViewsFlash/servlet/viewsflash?cmd=reviedata&pollid=XXX&style=reviewstyle
In the reviewstyle poll template, replace the Submit button with a Print button and add a Continue button to redirect to another page.

These URLs can also be used in a JSP page that offers the opportunity to review or update a record.

Security

These commands are sent to the URL specified in the URL column in the table below. The options in the **Security** page determine who can review or modify data. A visitor can review or modify only his survey submission. When using User Security, only a user with Review Data access rights to the survey's place can review or modify anyone else's data; if not using User Security, only the visitor who created his survey can review or modify his own data. Depending on the option in the Security, page, here's the review_uniqueID used:

	Review	Modify	URL
No one	N/A	N/A	
The visitor , right after completing it	The user ID from the survey's Authentication method; review_uniqueid is ignored	same	servlet/viewsflash
The visitor, anytime after completing it	The user ID from the survey's Authentication method; review_uniqueid is ignored	same	servlet/viewsflash

Aministrator with access rights	Uses review_uniqueid	same	servlet/vfadmin

Data updating looks for a unique ID and fieldname saved by the showupdata command, rather than the command line or form fields, so that data cannot be changed by simply entering a URL.

Next: Participants API

The Participants API

Related: **Invite**, **Invite Lists** and **Authentication**.

The Participants API allows other systems to tell ViewsFlash that an event has occurred, and that as a result, a participant has become eligible to participate in a questionnaire and an email may be sent to that effect. For example, after a Help Desk customer interaction is completed, a service is delivered, or an online transaction is placed, those systems can use this API to tell ViewsFlash about it. ViewsFlash can then send an email to the person involved inviting them to fill out a questionnaire about their experience. The e-mail can include a unique URL, which the questionnaire can use to look up the transaction and automatically import certain elements into the questionnaire, thus avoiding asking the customer again. For example, after ordering telephone service, the questionnaire could already know the type of service ordered, the employee ID of the person taking the order, and the duration of the call. This information could be used in the questionnaire to ask specific questions based on the type of service ordered, and the information gathered could be used to analyze employee performance.

For this purpose, a web service API is provided for applications to use. The API can be used from a JSP page with an include, or can be used as a REST API over HTTP, as described in **Web Services API.**. The application signals ViewsFlash that an event has occurred by sending a command to ViewsFlash/servlet/viewsflash with the following query string parameters:

&cmd=participants
&action= insert or delete
&pollid= Place!questionnaire
&userid=userid
&username=username
&email=email
&password=password

cmd=participants is required, and indicates that this query activates the Participants API

action=insert indicates that this participant should be added; action=delete indicates the participant should be removed

pollid= gives the name of the questionnaire's Place and its name. For example, questionnaire Service in place Examples is referred to as pollid=Examples!Service. This notation is also used in all URLs that refer to a questionnaire, including its URL

userid= gives the unique ID of the questionnaire, which is discussed in more detail below
username= gives the participant's name, e.g.,, John Smith
email= gives the participant's email address, e.g.,, john.smith@yahoo.com
password= gives the participant's password for his unique ID, when type of authentication is being used

The Participants API is meant to be used with a questionnaire that uses an Invite List. These lists work together with the authentication method used in the questionnaire. A list specifies whether to use userid, password, e-mail address and username, in any combination. The Participants API should use exactly the same parameters that are specified in the Invite List. The Security options used should be compatible with the Invite list. Before using the API, set up the questionnaire, configure its Security, set up an Invite list, and test the entire thing manually. The Invite List can be created using a few entries in CSV format. When using this API, there is no need to store the invite list on a database or an LDAP server.

For example, a questionnaire that uses question authentication and designates the email address field as the user ID would require a list with email addresses and user ID's, and the web service calls would include the email address twice, once in userid= and once in email= ; if username were used, it could be used in question piping to compose a personalized e-mail.

Another example: a questionnaire that used Basic (J2EE) authentication would require a userid in the invite list. e-mail and username would be optional. This setup is most useful when ViewsFlash runs in a portal, where the "Show all eligible" questionnaires presents portal visitors with a set of links to the questionnaires they are eligible to participate in. The Participants API, when it receives the query, will result in the new questionnaire being shown to the participant identified by the user ID.

Another example: a questionnaire uses question authentication and a Case Number to identify each questionnaire. This setup would be useful in the examples envisioned at the beginning of this section. The userid is the case number (transaction ID, purchase ID, etc). The email address belongs to the customer.

How this works
This Web Service writes records to the internal VWFTRACKING table. ViewsFlash, using a background thread, sends email to new entries on the list that have an email address. The ViewsFlash survey portlet, if used in a portal, also uses this table to populate the list of available questionnaires. Whether an email is sent or not is indicated in the Invite page of the questionnaire setup; the email address field in the Invite List must also be enabled. If a user ID is used in the invite list, respondents IDs must be on the invite list or they are rejected.

Extensibility
It may be desirable to include additional information about the questionnaire in the email invitation. For example, a building safety assessment could use a building ID code as the User ID, and it would be desirable to include the building name in the email. The VWFTRACKING table contains a CUSTOM column that can be used for this purpose. Additional columns can be added to VWFTRACKING as well; the columns can use any legitimate name that doesn't use special characters. To populate the CUSTOM column and the additional columns, use their names as additional parameters in the API query described above, prefixed with "extra." For example:
cmd=participants&...other parameters...&extra.custom=Harris+Building&extra.region=West
In this example, the CUSTOM column is being used to hold the building name, and a REGION column, created specifically for this purpose by ALTERing the VWFTRACKING table, contains the region name.

To use the CUSTOM and other columns in the body, subject, to, from, or reply to fields in an email, use piping syntax in the Invite page: [/custom] and [/region]. Use lower case only. For example:
Please take the survey at this URL ... The survey concernts building [/custom], located in region [/region].

See: **Invite**, **Invite Lists** and **Authentication**.

Next: Ratings API

The Ratings API

Refer to the **Support** section of the Cogix web site to download a set of files that can be imported into your ViewsFlash installation to deploy an actual working example, in a place named "ratings" and a questionnaire named "ratelive".

The Ratings API allows constructing ratings systems for content such as articles, videos, photos, and products. For each set of items to be rated, create a different ratings questionnaire. Ratings questionnaires are ViewsFlash questionnaires created using the Standard and Response styles or styles derived from it.

Each item to be rated includes a copy of the ratings questionnaire form, customized so that the item's identification code is inserted into a hidden field named item_id inside the form. This is usually done by a content management system. The form includes javascript that modifies the form's behavior so that instead of submitting the form, the content of the form is sent to ViewsFlash as an HTTP query using AJAX, and the response is used to replace the ratings form. The ratelive questionnaire includes all the correct settings and can be copied to any place of your choosing and can be further modified to improve look and feel and behavior as needed.

A Web Services API is also provided to retrieve an item's rating at any time. This API can be used by CMS systems to publish rankings either live or at certain intervals and can be cached. The API is one http query to ViewsFlash/servlet/viewsflash, with the following query string parameters:

&cmd=fetchrating
&pollid=SSSS!PPPP
&itemid=TTTTT
&count=1
&average=1
&decimals=N
&nodata=msg

Parameters fetchrating, pollid and itemid are required. Either count=1 or average=1, or both, are required. "decimals" defaults to none. "nodata" is what to return if there are no items to rate yet. SSSS!PPPP is the name of the Place and the Questionnaire of the ratings questionnaire. TTTTT is the id of the item; item id's must be unique within a ratings questionnaire, but can be duplicated across different rating questionnaies.

The query returns the following content:
COUNT space AVERAGE space
where COUNT is a number of the people who have rated the item, "space" means a delimiting space, and AVERAGE is the average rating for the item, formatted to the indicated number of decimal places. If something goes wrong during the query, a string is returned in the following format:
<viewsflash [/NG] - message>
where "message" indicates what went wrong.

The Ratings API uses a database table, VWFRATINGS, to store for each item:
- ITEMID - the pollid concatenated to the item id, must be unique
- SCORESUM and SCORECOUNT - the sum and count of the ratings; average is calculated from the two
- POLLID - the pollid
- DATEUPDATED - when the record was created or modified

Records are added to the VWFRATINGS table dynamically as needed; there is no need to create records in advance. Entries are removed automatically when the ratings questionnaire is deleted. The table is created automatically by ViewsFlash on startup provided the application has the appropriate database rights. The table can be accessed directly by other applications.

Next: XML data exchange

Using XML for data exchange

XML has become a lingua franca for information exchange between applications. ViewsFlash now supports this important protocol in the following ways. You can create Style Templates in XML. With these, and the appropriate commands, you can retrieve live poll results and capture the questions and answers being asked.

Creating XML Style templates.
1. Construct a template, but write it in XML instead of HTML. If the first characters of such a template are <?XML and a space, ViewsFlash will automatically recognize it as an XML template and it will return a page of MIME type text/xml instead of the usual text/html.

2. Within the template, you can use all the tags available to a Form, Response, or Archive template, as needed.

Using the Web Services API
If you have licensed the API, you can issue the following servlet commands:

/servlet/viewsflash?cmd=getpoll&pollid=pollingplace!poll&spotname=SSSS&style=XMLStructure
/servlet/viewsflash?cmd=getresults&pollid=pollingplace!poll&spotname=SSSS&style=XMLResults

SSSS is a pollingplace name, PPPP is a poll id in pollingplacename!pollname format,
style is a Poll Style template or a Results style template, as appropriate.

You can also use for a single questionnaire and for rotating polls, respectively:
/servlet/viewsflash?cmd=getxmlstructure&pollid=pollingplace!poll
/servlet/viewsflash?cmd=getxmlstructure&spotname=SSSS

Both of these commands can also specify:
style=TTTTT
If this is not specified, the simple built-in XMLStructure style will be used. You can create your own XML form style as well.
If style=* is specified, the XML returned will be exactly the same as generated by the **Export questionnaire as XML** option in the Place page.

Next: Automated poll rotation

Automated Poll Rotation using the Web Services API

You may want to see the helpful **visualization chart** in the next page before proceeding.

1. The Poll Designer creates a variation on the EmbeddedPoll template which adds a **history** parameter, as described below. This parameter includes the name of a Cookie that will track activity in all polls used with this API. The EmbeddedResults template can be used as is.

2. The Editor creates a place for the first dynamically created page. In the Settings page, select the templates created in Step 1 and in "Choose how to schedule polls and surveys", select "Only one at a time". The Editor also creates a test poll and schedules it so that it is open for a brief testing interval. In the Security page, set up options as you would normally, except that the settings for Cookies will be ignored. Suggestion: start with all Security settings turned off. See **Additional Security Information** below.

3. The Administrator modifies the script or program used to generate the dynamically created page to **invoke ViewsFlash**, as described below. It will refer, by name, to the place created in step 2 and to the Cookie named in step 1. At this point, bringing up the dynamically created page should display the page with the embedded poll; after voting, the page should refresh with the results display instead.

4. After fine-tuning the Security settings, the Editor can create additional places, copying the settings of the first one. When creating polls, it is important to always use Ending Dates; if one is not provided, ViewsFlash will let visitors vote again after 24 hours, since visitors cannot vote in another poll in the same place until the current poll ends.

Similarly, the Administrator can reuse the script or program in additional dynamically created pages. If you're using unusual security settings, it may be useful to set the Security settings of the Default poll in the Default place to these settings so that they will be used by default.

Additional detail on the above steps follows.

Invoking ViewsFlash script
The content management system must be capable of making an HTTP call to the ViewsFlash web server, and must be capable of retrieving cookies and performing string comparisons. Its performance will be increased immensely if it is capable of caching the results of HTTP calls by a given amount.

First, using pseudo-code with a Java syntax, here's what the script must do. A script that performs these functions must be inserted **(only once!)** in the containing page where the ViewsFlash place goes.

```
// define the ViewsFlash web server:
String viewsflash="http://...yourcompany.com/ViewsFlash/servlet/viewsflash";
// define the name of this place; for example, Politics:
String pollingplace = "Politics";

// ask for the current time; cache for 1 minute, return very large number if not available
String timenow = httpget (viewsflash + "?cmd=gettime", 1, "9999999999999");
// get a cookie with record of visited places
String cookie = getcookie ("VFPOLLS");

// scan the cookie for the character | followed by the name
// of this place followed by a colon, and return what
// follows until the next |character.
// If there was no cookie, or if this place is not found, return "0"
String poll_ends = cookie.extract ("|" + pollingplace + ":")

// if previous poll in this polling location expired,
// or if visitor's never been to this place,
// insert the poll voting form HTML (cmd=getpoll),
// else insert the poll results HTML (cmd=getresults)
if ( timenow > poll_ends ) {
  // cache poll and results for 1 minute, return space if not available
  insert ( httpget (viewsflash + "?cmd=getpoll&spotname=" + pollingplace , 1, " ");
} else {
  // cache poll and results for 1 minute, return space if not available;
  // use 0 instead of 1 to return results in real time.
  // optional: if command line contains parameter x=y, force real-time display
  insert ( httpget (viewsflash + "?cmd=getresults&spotname=" + pollingplace , 1, " ");
}
```

In brief, this script examines a cookie to detect a prior vote, and then calls ViewsFlash using cmd=getpoll or cmd=getresults to display the voting form or the results form, as appropriate.

In detail, this script examines a cookie named VFPOLLS (you can use any name you prefer), and determines if the visitor has voted in the current poll on this page by comparing the date now, as represented by ViewsFlash, with the date in the cookie. The cookie has the form:

|spotname:000000|spotname:00000|...|spotname:00000|

each "spotname" is the name of a place, and "000" is replaced with the date that the current poll in that location expires.

Checking the cookie is not mandatory, of course. This is just a way to present results to a visitor instead of the voting form. The script can also interrogate its query string to see if it should display the voting form or the results.

If you are using the apiport security feature, then all the above calls to ViewsFlash must be made on that port; if they are not, ViewsFlash will return an empty page. See **Application Security**.

History template parameter
When ViewsFlash receives a submitted vote, it updates this cookie with the expiration date of the current poll. To do this, it is necessary that the Poll Style Template which generate the voting forms include the following hidden field:

<input type="hidden" name="history" value="1,VFPOLLS,[/spotname],*">

The meaning of the hidden history field is as follows. The 1 specifies the cookie scanning methodology described here. VFPOLLS is the name of the cookie to examine and toss back. [/spotname] is a ViewsFlash template substitution tag which is replaced, when the poll voting form HTML is created, by the name of the place in which the Poll is hosted.

The final parameter, when present, means to redirect the visitor's browser after counting the vote. A * means to redirect to the URL of the referring page, i.e., the page where the voting form appeared. When the client browser receives this redirect, it will ask the web server for the page again. Because the cookie has been reset, however, this time the script described above will insert the poll results instead of the poll voting form, and the visitor will have the desired experience of seeing the vote results after submitting his vote. Instead of *, it is possible to use a standard URL, which is useful when you want to have voting results displayed in a unique page. If this field is omitted, then the standard action (specified in the html by the results hidden field) will be carried out, such as displaying a particular response page with or without current results.

It is also possible to code this field as "*?x=y", for example. This causes ?x=y to be appended to the referring URL. This mechanism can be used with systems such as ATG Dynamo, JSP and ASP to interrogate the URL and display results instead of a voting form, for example. If the URL already contains other parameters, such as ?x=z&p=3, the substitution is "smart" and translates into ?x=y&p=3; in other words, other parameters are preserved and if there's already a parameter by the indicated name, it is replaced.

Note that the "history" hidden field should only be used when you are using a Web Services API as detailed here.

An alternative form of the history parameter is: 2,COOKIENAME,*
The COOKIENAME is an arbitrary cookie name; the [/spotname] must be removed; and the last parameter has the same functionality as the original form. When used like this, the cookie simply contains the id's of the polls the visitor has voted on, like this:
|pollone|polltwo|...| This cookie can be parsed easily, if desired.

Additional Security Information
The settings in the poll Security page are all available, except for Tossing a cookie, which is superseded by the History parameter. Duplicate IP detection is strongly recommended, as it is easy for visitors to just press Back and return to the Voting Form. User authentication may be used.

In the When a duplicate vote is received section, respond normally is best. If you choose to insert a message, then the results shown will be a whole page, without redirection. If you specify to display a page, it will be shown by redirection, so it must be an absolute URL.

Incomplete entries may or may not work in a reasonable fashion. In these applications, it is typical to ask very few questions and to choose tally whatever is available. Other options may produce confusing behavior.

Error checking
The API commands log errors to the ViewsFlash administrative log. In addition, they may return HTML like this:
<viewsflash [/NG] error message >
This message is invisible in a browser, so that if an unexpected error condition occurs, the visitor's experience is not disturbed. When the same command is issued from a browser, the software appears to return a blank page. A view source will

reveal the nature of the problem.

The code that issues getpoll and getresults should scan the answer for a string containing [/NG], and if such a string exists, it should report the error condition in an appropriate manner.

Next: Automated Poll Rotation Control Flow

Extending ViewsFlash with Java

Because ViewsFlash is a Java application, it is naturally extensible. Start by compiling the source code provided in the Java source files in the viewsflash/src directory. To activate your class, use the indicated servlet parameter.

Compiling your class

1. You will need the Java 1.3 or later SDK from Sun. To compile the classes, use "javac" from the command line. You must have the appropriate library jar files in the -classpath argument. For example, if you have a file called MyClass.java, use:

/<JDK_HOME>/bin/javac -classpath .../ViewsFlash/WEB-INF/lib/viewsflash.jar myClass.java (replace ... with your application server's webapp directory, where you installed the ViewsFlash application, or simply provide the path to a copy of the viewsflash.jar file)

You also need to include servlet.jar in the classpath. In Unix, use: ...viewsflash.jar:.../servlet.jar. In Windows, use a semicolon instead of a colon. The servlet.jar file is usually in the application server's directories.

To see what libraries you need to include in the classpath, look at the first few lines of the source code we provide and examine the import statements. Below are the most common. Make sure that you use the complete path.

Always:
/ViewsFlash/WEB-INF/lib/viewsflash.jar
servlet.jar

For Oracle libraries:
oracle/ora81/jdbc/lib
oracle/ora81/jdbc/lib/classes12.zip

For ATG Dynamo:
ATG/Dynamo5.1/DAS/LIB/classes.jar
ATG/Dynamo5.1/DPS/LIB/classes.jar
ATG/Dynamo5.1/DSS/LIB/classes.jar

2. After successful compilation, you should see a file myClass.class in the same directory as your .java file. Move this file to the /ViewsFlash/WEB-INF/classes/com/cogix/vwf directory.

Use /servlet/viewsflash?Diagnose=1 to verify what your classpath is.

Activating your class

After compiling your class and moving it, you may need to add another servlet parameter, as specified in the table below. Add the parameter to viewsflash.properties, and restart the application server.

Sample source code for each extensible class is in ViewsFlash/viewsflash/src, under the sample class names listed below. See the class source code for detailed documentation on its use and how to modify it.

Base Class	Servlet parameter with sample Class	Purpose
PublishNotifier	publishnotifierextension= com.cogix.vwf.PublishNotifierExtension	Methods are called when a poll is scheduled, opened, and closed. Used to convey this information to web site publishing systems.
VoteNotifier	votenotifierextension= com.cogix.vwf.VoteNotifierExtension	Methods called after a legitimate vote is recorded. Used to record the vote and/or its content in containing applications.
RequestNotifier	requestnotifierextension= com.cogix.vwf.atgRequestNotifier	Methods called as soon as request is received. Used to extract authentication information from the environment into a useful place, such as a Request Attribute.
Validator	Activated with the Custom Action field in the Question page rather than a servlet parameter. testAction BeforePageComposedGoTo WhenPageReceivedGoTo WeightedScore WriteInRanking WeightedRanking SimpleQuiz	Methods called before vote or survey is tallied. Used to check fields for validity. Also used to calculate additional fields or modify the values of some fields based on others.

	NumericScore IssueHttpCommand	
DataBaseConnection	database=OraclePooledDataSource	All database drivers (Oracle, DB2, etc) extend this class.
DatabaseExtension	databaseextension= com.cogix.vwf.OracleDatabaseExtension	Provides an alternate way of storing responses to all surveys and polls in a single table in the database.

Next: Writing custom Actions

Writing Actions

Because ViewsFlash is a Java application, it is naturally extensible. There are two types of extensibility available: Action classes, and Notifications.

Actions (see built-in **Actions**)

These classes, which extend the base **Validator** class, allow you to add code to check a response's validity and compute a new field based on the values of other fields. To use these classes, you will need to write some Java code, compile it, and put it in the "classpath" where ViewsFlash can find it. To tie it all together, you add a question of type Action to your poll or survey. In the field right below it, you enter the name of the validation class, such as "Weighted Ranking", followed by a space, followed by the parameters required by the specific validation class.

1. Study the source code for **testAction.java** - a class designed to familiarize yourself with the common operations possible in an Action class.

You can also study source code provided for other classes in the viewsflash/src directory. Other examples are highlighted below.

2. Compile your code and move the .class file to the .../ViewsFlash/WEB-INF/com/cogix/vwf directory, and restart your application server. For complete instructions, **click here**.

3. In the questionnaire where you want to use the Action class, add a question of type Action. In the field provided, enter the classname, a space, and the parameters required by the Action class. Usually these include the name(s) of other questions, for example. If ViewsFlash cannot find your class, it will say so when you add this question. Once it finds it, if you change the code in the class, you will need to restart the web server to unload the class from the Java VM.

4. If the class can invalidate entries, be sure to review the options for Incomplete Entries on the Security page.

Methods inherited from the parent Validator class that can be overridden:

boolean **onPageComposition** (RequestParams reqx, String questionname, String spec)

This method is called before a survey page is composed and displayed to the person taking the survey. It can refer to any value entered using getParamValue. This method returns true to ask ViewsFlash to display a different survey page instead of the page it was about to display; the designated page is the one that contains the question set in the skipToQuestion method. All Actions on the destination page that have an onPageComposition method will be executed before the page is displayed, and any of them can redirect to another page.

If this method returns false, it indicates that it has nothing to say.

For an example, see **BeforePageComposedGoTo.java**

boolean **onPageReceived** (RequestParams reqx, String questionname, String spec)

This method is called after a survey page is received but before the values entered in it are stored. This method returns true to ask ViewsFlash to act on its conclusions regarding data validity, which are indicated by the methods below. It returns false to tell ViewsFlash that it has nothing to say about the validity of the data in the current page. In either case, this method can modify the data entered using setParameterValue.

For an example, see **WhenPageReceivedGoTo.java**

boolean **checkValidity** (RequestParams reqx, String questionname, String spec)

This method is deprecated. It has been replaced by the better-named onPageRecieved method, which has the same functionality. Classes that use this method continue to function as before.

String **getDefinitionDisplayText** (String classname, String args)

Use this method to override what the survey designer sees in the Design Questions page and Index page for an Action. By default, the Action name and its parameters are displayed.

String **checkArguments** (String classname, String args)

Use this method to examine the arguments provided to the class and report anything wrong with them in the returned String.

If nothing is wrong, return null. If not overridden, this method returns null.

Convenience methods in the Validator class:

void **setRedir** (String s)

Sets redirection to this URL. Use * for redirection to the referring URL.

void **setValid** (boolean b)

Indicates whether the entry should be accepted or rejected as invalid.

void **setMessage** (String m)

Sets a message that will be displayed in the [/statusmsg] field of a results page.

void **setErrorTemplate** (String t)

Sets this as the path to a file to use as a template to display the invalid condition. Note that this is NOT the name of a style template, but rather the name of a file.

void **setMissingEntries** (String m)

Sets this string with the names of fields that are flagged as missing, separated by leading and trailing spaces.

void **setAction** (int nAction)

Indicates what to do after returning true. It also calls setValid() as described below. Valid values for nAction are:

Validator.AcceptAndRedirect means accept the vote and redirect to the URL set by setRedir(). Calls setValid(true). Use with setRedir (String).

Validator.ActionRefill means to reject the vote and redisplay the questions with the corresponding settings from the Security page for incomplete entries. Calls setValid (false). Use setMessage (String), setMissingEntries (String). If you don't call setErrorTemplate (String), the standard poll style will be used to redisplay the question; if you do, that page will be used as a template.

Validator.ActionRedirect means reject the vote and redirect to the URL set by setRedirect(). Calls setValid (false). Use with setRedir (String).

Validator.ActionShowPage means reject the vote and show results using the results page template set by setErrorTemplate(). Calls setValid (false). Use with setMessage (String) also.

Validator.ActionAcceptWhatThereIs means accept any data that is present and ignore any missing data. Calls setValid (true);

Validator.ActionUseInvalidSettingsonSecurityPage means take the action specified in the Security page. Requires that you call setValid (boolean) explicitly.

String **getAnswerText** (String qname, String answer)

Returns the full text of the "answer" to question "qname". Returns null if it can't find it.

String **getDefaultAnswerValue** (String qname)

Returns the default value for question "qname". This is the value checked as Default, which when used in Quizzes indicates the "correct" answer.

definition **getDefinition** ()

Returns the internal definition class for the poll. Advanced use only. Returns null if not found.

Poll **getPoll** ()

Returns the internal Poll class for the poll. Advanced use only. Returns null if not found.

String **getParamValue** (RequestParams reqx, String qname)

Returns the value of the answer entered by the visitor in question "qname". Returns null if not found. Returns "" for an empty text field. Returns null for an unchecked check box. Returns "value1,value2,value3" for a multiple check box.

boolean **isValid** ()

Returns the value set earlier by setValid (boolean), or false if it has never been called.

void **setParameterValue** (String qname, String value)

Will set question "qname" to value "qvalue". The value is available as if it had been included in the query string, and is useful for setting the result of a calculation in a field where it can be stored, tabulated, or analyzed. Note that to analyze such a field, you must define all its possible values; for example, you can set up a hidden field with answers that match the expected values.

For an example, see **WeightedScore.java**

void **skipToQuestion** (String qname)

Use this method to indicate that the next page shown to the person taking a survey should be the page that contains question 'qname'. Be sure to return true.

boolean **valuesContain** (String valueslist, String value)

This method tests to see if 'value' is found in the comma-delimited values in 'valueslist'. This list is in the format returned by getParamValue. Returns true if value is found on the list. Returns false if either argument is null or empty.

String **substitutePipedParameters** (String m)

This method performs question piping on String m. If m contains expressions of the form [/questionname], the current value of that question will be substituted in its place. See **Question Piping**.

The returned String contains the original string with all [/questionname] values replaced by their respective substitutions.

void **WriteLog** (String msgid, String msg)

This method writes to the ViewsFlash log and to the survey or poll's log, a message with ID 'msgid' and text 'msg'.

'msgid' is usually a code of the form "NNNE" or "NNNI". the E or I indicates an Error or Information. The NNN is a number; use numbers above 1000 for your own Actions. These messages are visible with the View Log buttons.

Use this method liberally while debugging and sparsely when going to production.

Next: Event Notification

Writing Event Notification classes

Another way to extend ViewsFlash's capabilities is by adding code of your own which will be executed at key events.

1. Study the source code provided, and refer to its documentation in the section below.

Base Class	Purpose
PublishNotifier	Called on Publishing events
VoteNotifier	Called when a valid vote is cast
RequestNotifier	Called after an HttpServletRequest is received and parsed.

2. Compile your code and move the .class file to the .../ViewsFlash/WEB-INF/com/cogix/vwf directory, and restart your application server. For complete instructions, **click here**.

3. Add the indicated viewsflash servlet configuration parameter to the viewsflash.properties file. For example: publishnotifierextension=PublishNotifierExtension

4. Restart your application server.

5. Use logging statements to debug. Look for them in the ViewsFlash Administrator log in the Administrator page, or in viewsflash.log file in the viewsflash data directory. The classes include init() and destroy() methods, useful for logging and initializing.

PublishNotifier

ViewsFlash often needs to integrate into an authoring system. Using this extension, your Java code can notify the authoring system, or any other application, when a poll or survey is scheduled for publication. It can also take action right after it is actually published. With this information, the publishing system can move files from staging to production servers, for example.

Write your class starting with this skeleton code for **PublishNotifierExtension.java**. Compile the class as specified above. Use the following viewsflash servlet parameter to indicate the name of your class:
publishnotifierextension=com.cogix.vwf.PublishNotifierExtension

com.cogix.vwf.PublishNotifierExtension methods:

void **onPublishChange** (File pollfile, File resultsfile, String spotname, String pollid, Date open, Date close, Spot spot, definition def, Poll pol, ThisCall thiscall)

This event happens when the Submit button on the Publish page the ViewsFlash accepts the changes, showing Changes accepted in the browser. At this moment, you have the most current information on a poll or survey's schedule.

Parameter	Meaning
spotname	The place name
pollid	The Poll ID
open / close	Date when the poll or survey opens / closes (can be in the past or future, can be null if undefined)
pollfile / resultsfile	Where the HTML created for the voting form or the response page are. Note that at the time of this event, these files may not exist or may be outdated. If you need their content, use **OnPublishOpen**.
spot /def / pol	These objects are useful in advanced applications. Examine the code for examples of their use.
thiscall	This object provides access to environmental objects such as the request and response objects. Examine the code for examples of its use.

void **onPublishOpen** (File pollfile, File resultsfile, String spotname, String pollid, Spot spot, definition def, Poll pol)

This event happens at the designated Start Publication time, when a poll or survey opens, and its HTML form and response page are written to disk. Sometimes it is called once, for the poll page, and again for the results page; it can also be called once for both. Check carefully for null parameters.

Parameter	Meaning

spotname	The place name
pollid	The Poll ID
pollfile / resultsfile	Where the HTML created for the voting form or the response page are. If you wanted to send the files to a staging server using ftp, you could do it here. Either of these can be null.
Spot /def / pol	as above

void **onPublishClose** (File pollfile, File resultsfile, String spotname, String pollid, Spot spot, definition def, Poll pol)

When the poll or survey is closed, this method may be called, but there is no guarantee that it will be called. Its arguments are the same arguments as OnPublishOpen.

Void **onPublishExtra** (File resultsfile, String spotname, String pollid, Spot spot, definition def, Poll pol)

This event happens when an Extra report is scheduled for publication, after its HTML is written. Check carefully for null parameters.

Parameter	Meaning
spotname	The place name
pollid	The Poll ID
pollfile	Where the HTML created for the finished report page is. Can be null.
Spot /def / pol	as above

VoteNotifier

Some applications call for tracking where or when people have voted. Using this extension, your Java code can notify another application after a valid vote has been received. To validate a vote, use the Validation classes instead.

Write your class starting with this skeleton code for **VoteNotifierExtension.java**. Compile the class as specified above. Use the following viewsflash servlet parameter to indicate the name of your class:
votenotifierextension=com.cogix.vwf.VoteNotifierExtension

com.cogix.vwf.VoteNotifierExtension methods:

void **onVote** (Poll pol, ThisCall thiscall)

This event happens when a valid vote or survey is received.

Parameter	Meaning
pol	The Poll object. Use pol.pollid() to get the poll's ID.
thiscall	This object provides access to environmental objects such as the request and response objects. Examine the code for examples of its use.

RequestNotifier

This extension implements the **Request Authentication** method for authenticating visitors in the Security page. When running inside an application server, it is common for the servlet context or the request attributes to contain useful information, such as a authenticated User ID. The onRequest method allows for retrieving the information and putting in a place where it can be used by ViewsFlash earlier.

void **onRequest** (ThisCall thiscall)

This event happens right after the HttpServletRequest is parsed. This method is expected to set two servlet request attributes using HttpServletRequest.setAttribute (name,value). The two attributes have preset names: RequestNotifier.RequestNotifierTransient and RequestNotifier.RequestNotifierId. Set RequestNotifierTransient to "1" if the user ID is not a permanent one. Set RequestNotifierId to the unique UserID.

The information set in these two request attributes is used by the "Use Request Authentication" option in the Security page.

Parameter	Meaning
thiscall	This object provides access to environmental objects such as the request and response objects. Examine the code for examples of its use.

Next: Extensible Authentication

Extensible Authentication

In ViewsFlash, User Authentication is done by the Application Server, using Java's getRemoteUser() method, which normally enforces authentication and provides single sign-on.

When this is not feasible or practical, as for example when using Site Minder for authentication, ViewsFlash can be configured to use an alternative authentication method using its Extensible Authentication architecture.This allows specifying a class in the servlet parameters that will perform the same function as getRemoteUser(). The built-in OtherAuthentication class can be used, or you can even write your own.

Using the OtherAuthentication class

Include the following parameter in the ViewsFlash servlet parameters:
requestnotifierextension=com.cogix.vwf.OtherAuthentication

Include one of the following in the ViewsFlash servlet parameters. Do not include both
OtherAuthentication.HeaderName=HeaderName
OtherAuthentication.SessionAttributeName=AttributeName

Use OtherAuthentication.HeaderName if the authentication system and the web server create an HTTP header that includes the user ID, and set this parameter to the name of that header.
Use OtherAuthentication.SessionAttributeName if the authentication system and the application server create a session attribute that includes the user ID, and set this parameter to the name of that session attribute.

The OtherAuthentication class looks for the user ID as directed above. If no user ID information is present, it redirects the user to a URL specified by this servlet parameter:
#OtherAuthentication.Redirect=http://...?xxx%1....%2....%3
Set the value of this parameter to a URL. In the URL, include the values that are appropriate the authentication agent. The values %1, %2 and %3 are replaced by:

- %1 the complete original URL
- %2 the original requestURI
- %3 the original query string

With this class, the ViewsFlash application can be protected using **User Security**. However, it's probably not necessary to secure the URL pattern /servlet/vfadmin with a security-constraint in ViewsFlash web.xml. It may be desirable to set up the authentication agent so that the /ViewsFlash/servlet/vfadmin URI requires authentication by the agent.

When setting up questionnaires, if the Basic Authentication is selected,this class will authenticate respondents instead of J2EE's getRemoteUser. In web.xml, it's probably not necessary to secure the URL pattern /servlet/vfauth, and it may be desirable to secure /ViewsFlash/servlet/vfauth with the authentication agent.

Extensible Authentication allows using an authentication mechanism as well as looking up Roles independently from the Application Server.

Writing your own Authentication class

The RequestNotifier Extension mechanism can be used to verify whether a user is authenticated. When using this, be sure that web.xml does not protect the ViewsFlash application or its servlets. In surveys, use Basic Authentication as the authentication method. See **AuthenticationSkeleton.java** for sample code.

1. Create a class com.cogix.vwf.YourExtensionClass that extends RequestNotifier, and override its OnRequest method as follows:

If the user is successfully authenticated, set:
thiscall.userid = theUserID; // a String with the User ID.

If the user is not authenticated, set:
thiscall.userid=null

If browser redirection is needed to delegate authentication to an authentication server, set:
thiscall.bRediraftersetattribute = true;
thiscall.rediraftersetattribute = reqURL; // usually will contain the original ViewsFlash query string

2. Add to viewsflash.properties:

requestnotifierextension=com.cogix.vwf.YourExtensionClass

3. If there is a severe internal error, throw new vwfException(String message). The message WILL be visible to the visitor. It is better to simply set thiscall.userid=null and use a logging function to record the error.

Role Lookup

The Role Lookup Extension mechanism can be used to verify whether an authenticated user is in a particular Role. When using this, the Application Server's role lookup method, if any, will be ignored. To activate it, add the appropriate lookuprole parameter to viewsflash.properties, as follows:

1. To use an LDAP server, see **Using LDAP** and add:
lookuprole=LDAPLookupRole

2. To look up roles in a properties file, add:
lookuprole=FileLookupRole

This will look for a servlet parameter
filelookuproleproperties=<path to a file>
If this parameter is not present, the file will default to /etc/cogix/fileroles.properties

This file has the following format:
user=role,role,role,role,role (no limit on how many)
Spaces are ignored.

For example:
sally=Creator,MedicalFaculty,LawFaculty
ivan= Creator,Analyst,MedicalFaculty
peter=Analyst,MedicalFaculty
tom= VFAdministrator
eric= Creator,Analyst,Examiner,MedicalFaculty
jenny=VFAdministrator,Creator,Analyst,Examiner,Producer,MedicalFaculty,LawFaculty
sandy=Examiner,LawFaculty
bob= Analyst,Examiner,MedicalFaculty
lucy= Creator,Analyst,LawFaculty
mary= Creator,Producer,LawFaculty

Role names are arbitrary. They do not need to correspond to any role or group names in the Application Server.

3. To look up user roles in a database table, add:
lookuprole=DBLookupRole

See **DBLookupRole.java** for sample code.
These role names are arbitrary. They do not need to correspond to any role or group names in the Application Server.

4. To use your own extension, use FileLookupRole.java as an example, and create your own class, and activate it with servlet parameter:
lookuprole=YourRoleLookup

Next: Database Storage

Normalized Database Storage with extensible Java classes

To store response data in a Normalized format, the DatabaseExtension class can be extended. The **DatabaseExtension** class is provided, with **source code**, as a guide. Some database administrators prefer this method rather than creating one table per survey. Similar classes are provided for all supported databases. The correct classes are enabled by default.

User Interface

The Save page includes the following check box:
☐ Save in Normalized Database

When checked, each vote or survey response will be stored as a series of records in table VWFDATA. For example:

RECORD_ORDINAL	TIME_STAMP	SPOTNAME	POLLNAME	VOTE_ORDINAL	AUTHENTICATEDUSERID	QUESTIONNAME	ANSWER
194	25-Jun-01	DBTest	DBTest1	193	Carlos1998	radio	2
195	25-Jun-01	DBTest	DBTest1	193	Carlos1998	boxes	a
196	25-Jun-01	DBTest	DBTest1	193	Carlos1998	boxes	b
197	25-Jun-01	DBTest	DBTest1	193	Carlos1998	boxes	c

Note that neither the RECORD_ORDINAL or VOTE_ORDINAL numbers are sequential. Each response has a unique, ascending VOTE_ORDINAL. Each record in the table has a unique, ascending RECORD_ORDINAL. AUTHENTICATEDUSERID will be blank if not available.

Installation

1. Begin by studying the source code for DatabaseExtension.java. If you plan to modify it, copy it under a different name.

2. Compile your code and move the .class file to the .../ViewsFlash/WEB-INF/com/cogix/vwf directory, and restart your application server. For complete instructions, click **here**.

3. Add the following servlet parameter to viewsflash.properties and restart the application server:
databaseextension=OracleDatabaseExtension

4. If you have not created other ViewsFlash database tables, let ViewsFlash do it (either with ?Diagnose or a ?dbcmd=dbmigrate). Otherwise, create the needed table, index, and sequence with the following SQL statements:

```
CREATE TABLE VWFDATA ( RECORD_ORDINAL INTEGER NOT NULL PRIMARY KEY,
TIME_STAMP DATE, SPOTNAME CHAR (32), POLLNAME CHAR (32),
VOTE_ORDINAL INTEGER, AUTHENTICATEDUSERID CHAR (32), QUESTIONNAME CHAR (32), ANSWER VARCHAR2 (4000) )

CREATE INDEX VWFDATAINDEX on VWFDATA ( SPOTNAME, POLLNAME, VOTE_ORDINAL )

CREATE INDEX VWFDATAINDEX2 on VWFDATA ( VOTE_ORDINAL )

CREATE SEQUENCE VWFDATASEQUENCE ORDER MAXVALUE 9999999999999999
```

5. Restart the application server. If creating tables, make sure that the tables have been created. Check the settings for the Save page on the Default poll in the Default place. The Save in Normalized Database check box should appear.

DatabaseExtension class

Only the following methods are customized normally. Refer to the source code for further documentation.

String getSavePageMessage ()
This function determines the text visible on the Save page that activates saving data for that particular poll. If this extension is not used, nothing appears. The default text is "Save in Normalized Database".

boolean bDisableNativeDatabaseSave ()
If you return true, the menus on the Poll Save page that allow for storing data in a database by creating tables, etc., will all disappear. By using this option, you are effectively replacing the built-in database saving code in ViewsFlash with this class completely. The default is false.

String getVersion()
At initialization, the name of your class and this version number will be written to the ViewsFlash log. The default is 1.0.

Refer to the source code for further information. Changes that you make may alter the application's behavior.

Next: Style Templates

Style Templates

Note: when first using ViewsFlash, we recommend using the pre-written styles and becoming familiar with the entire application first, before creating your own. We recommend NOT CHANGING the original style templates, but rather creating your own from copies of the original templates, because they are subject to change as more capabilities are added.

When a Place is created, the Form style used for creating questionnaires and the Response style used to display the response page are chosen by picking the appropriate style from a menu and setting appearance options.

Styles are available from the Styles page. They are subject to access controls set by the application administrator. A survey creator who does not have the right to use a style will not have that style available in his or her menu choices.

Here's a quick description of how styles are used. When a respondent is presented with a questionnaire, Form and Question styles are used to dynamically compose each page of a questionnaire. After the questionnaire is completed, the Response style is used to compose the page presented to the respondent, including optionally a graph showing how people have answered one or more questions in real time. Archive styles are used to compose archived questionnaire listings. Here are examples of these styles using a very simple questionnaire.

<table>
<tr><th>Form style</th><th>Response style</th></tr>
<tr><td>


What is your favorite aircraft?
○ Boeing 707
○ McDonnell Douglas DC10
○ Boeing 747 Stratocruiser
○ Boeing 777 Luxury Liner


50% complete
</td><td>


What is your favorite aircraft?
Boeing 707 54%
McDonnell Douglas DC10 13%
Boeing 747 Stratocruiser 30%
Boeing 777 Luxury Liner 2%
756 responses
</td></tr>
<tr><td colspan="2"></td></tr>
<tr><th>List Archive style</th><th>Menu Archive style</th></tr>
<tr><td>
Travel Poll Archive

Click on a poll to see its results:

Airplanes
Cities
</td><td>
Travel Poll Archive

Click on a poll to see its results:
Choose one Go!
</td></tr>
</table>

There are several kinds of style templates. Form and Question styles are used to give questionnaire forms their look and feel. The default Form style is called Standard_, and it works with question styles whose names begin with Standard_, such as Standard_Matrix, Standard_Calendar, etc. See **Question Styles** for the complete list.

Style templates:

Style Name	Type	Purpose
Standard_	Form style	renders standard questionnaire forms
Results	Response style	renders response page presented to visitor after questionnaire is completed
UnivariateReport	Univariate report	renders Univariate analysis reports

XtabReport	Crosstab report	renders Cross tabulation analysis reports
DataReport	Data report	renders Data grid reports
XMLStructure	Form	renders questionnaire in XML format
XML	Response	renders live tallies in XML format
ListArchive	Archive	renders archive listings
MenuArchive	Archive	renders archive listings
See also **Question Styles**		

The Styles page

This page lists available styles. To view or edit a template, click on its name. To remove one or more templates check the box next to each, choose Remove from the pull down menu and press Go.

To create a new style, it is best to copy an existing style as follows: check the box next to the style to copy, pick Copy selected, give the style a name, and press Go. It is also possible to pick New Style, enter a name for the new style (usually with a trailing underscore), and press Go.

If the Standard_ form and question styles are not flexible enough to meet your needs, you may create your own. To do so, make a copy of the Standard_ style, and include an underscore as the last character of the name of the new style, eg., Corporate_ . When you create the style, you will notice a field that says "Compatible with Standard_". This means that all question styles that begin with Standard_, such as Standard_Email, will work with the Corporate_ form style as well as the Standard_ style, and will be available for use. If the style is intended to be incompatible with the Standard_ style, leave the field blank, and create question styles that begin with Corporate_.

The support section of **www.cogix.com** includes a Styles Library. To include a style downloaded from there, save the file in a /temp directory. Create a new style, indicate what type of style it is, and press the Upload button. Select the downloaded file from the /temp directory and press Upload.

Editing Styles

Before creating a template, check the Place Settings page to see if the appearance options are sufficient. They include, for example, the questionnaire's colors, fonts, type size and width, and the colors in the response page's graphical percentage bars. The place settings selection menu normally shows only form styles that are compatible with the currently selected style, but it includes a View All choice that allows choosing any form style.

The Style Template page is used to modify the style. Styles are HTML and Javascript code, mixed in with special ViewsFlash tags, described in **Style Tags**. ViewsFlash fills these tags with the appropriate contents, such as the title of a questionnaire or the text of a question. It is best to leave these tags alone and to work carefully around them. For example, it is safe to add colors to the foreground or background, or to insert an advertising banner and logo at the top of a page.

The Preview button may be useful in previewing changes, but depending on what they are, it may be necessary to create a test questionnaire to fully see the results produced by the template.

Next: Style Tags

Style template tags

Style Templates are HTML pages or page fragments, sprinkled with ViewsFlash-specific substitution tags. These tags tell ViewsFlash, for example, where to insert the results of a poll, such as how many people have voted.

For example, suppose that you have created a poll in which the question "Party" is asked, with two possible answers, "Democrat", and "Republican". If your Response Style template contained the following HTML:

```
Party affiliation: [/Party,pollcount] people responded
[/Party,count,Democrat] Democrats ( [/Party,pct,Democrat] %)
[/Party,count,Republican] Republicans ( [/Party,pct,Republican] %)
```

when the Poll response page is displayed, it would look like:

```
Party affiliation:1370 people responded
587 Democrats ( 43 %)
783 Republicans ( 57 %)
```

From this example, you can see that the substitution tags have been replaced with various values from the polls, such as the number of people who responded (*pollcount*), how many people (*count*) voted Democrat and Republican, and the percentage (*pct*) of people who voted Democrat and Republican.

***A tag has the form* [/tag,tag,tag,tag] *Note the* [/ *and the ending*]**

Inside the [/ and], a tag has several parts, which are keywords separated by commas; their meaning varies as described below.

The tags can be inserted anywhere on a page; they are only meaningful to ViewsFlash. If you use a web page editor, it will show the tags as text. If you prefer, there is an alternate syntax for ViewsFlash tags.

Instead of [/tag,tag,tag,tag] you can use <vwf=tag,tag,tag,tag>

Style Templates can be edited from ViewsFlash, using the Style Templates pages. You can also edit the templates in your favorite web page editor, upload them to the ViewsFlash server site, and then edit them using the Style Templates pages.

Some editors will hide the alternate syntax; some editors will highlight it. Which one you use is up to you. They are completely interchangeable. It is often desirable to put a ViewsFlash tag in between table rows (<tr></tr>) tags. Some editors will not allow the [/...] form and it is necessary to use the <vwf=...> form. If you will be using an editor to edit the Style Templates, experiment with it to see which tags work. But it may be considerably easier to just edit the HTML in the Style Template pages.

Style tags often come in pairs, such as [/allquestions] [/endallquestions], or <vwf=allquestions> <vwf=endallquestions>. In such cases, all HTML between the tags will be treated as a unit, whose meaning depends on the tag.

Different style tags are available for **poll pages**, **response pages**, and **archive pages**. Additional, special tags are available for **quizzes**.

Next: Form style tags

Form style tags

As you read this section, you may want to open the **Standard_** template and View its Source.

Tag	Replaced with
The following tags take their values from the Setup / General settings and Setup / Visual Editor pages.	
[/polltitle]	The questionnaire title
[/pollheader]	The questionnaire header
[/pollfooter]	The questionnaire footer
[/allquestions] [/endallquestions]	All HTML between these tags will be repeated, once for each Question. [/allquestions,noinclude] is used to ignore Question Styles.
[/qtext]	The question's full text
[/qname]	The question's name
[/ifqhelp][/endifqhelp]	The HTML between these tags, when question help is checked.
[/qhelpmarker]	The question help indicator, which defaults to [?]
[/qhelp]	The question help text
[/qtype]	The question's type.
Between the allquestions and endallquestions tags, the following tags take their values from each answer to the current question, as defined in the question page.	
[/atext]	The answer's full text
[/avalue]	The answer's value
[/aordinal]	The answer's ordinal number, starting with 1
The following tag pairs describe what HTML to use for each different question type:	
[/radio] [/endradio]	For radio buttons
[/checkbox] [/endcheckbox]	For check boxes
[/menu] [/endmenu]	For drop-down menus
[/text] [/endtext]	For single-line text fields
[/twidth]	The width of a single-line text field. Default is 15.
[/tmaxsz]	The maximum number of characters allowed in a text field. Default is no limit.
[/mtext][/endmtext]	For multi-line text fields
[/tacols]	The width of a multi-line text field. Default is 15.
[/tarows]	The height of a multi-line text field. Default is 4.
[/hidden][/endhidden]	For hidden fields
[/ltext][/endltext]	For HTML fields
[/qname]	Used to compose the destination anchor for skipping functionality.
[/qnamejs]	Same as qname, but without using special characters
[/ifother][/other][/endifother]	When other is checked, the text of the Other field
[/allanswers] [/endallanswers]	Everything between these tags will be repeated for each Answer to the current Question.
[/nextanswer] [/thisanswer] [/endthisanswer]	The [/nextanswer] tag moves to the subsequent answer in an [/allanswers] loop. Everything between [/thisanswer] and[/endthisanswer] is expressed for the current Answer; if there are no more answers, nothing is shown. These three tags together are useful in multiple-column layout voting forms.
The following tags are used between the above pairs, as appropriate, to provide default choices:	

[/checked]	"checked"
[/selected]	"selected"
The following tags are used in each question to provide skipping functionality.	
[/qskipper][/endqskipper]	Delimit HTML used to skip to a question
[/qskipto]	The name of the question to skip to.
The following tags provide automatic question numbering:	
[/qnumbering] [/endqnumbering]	The enclosed content will be expressed if automatic question numbering is enabled.
[/qno]	The question number
The following tags link the voting form to other parts of the voting system.	
[/vwfservlet]	ViewsFlash servlet URL
[/cmd]	The ViewsFlash command that will be used to process this voting form.
[/pollid]	The unique ID of the poll to which this vote will be posted. Defined in the place as the Poll Name.
[/allalphapollid]	The pollid without special characters, such as !
[/results]	The location of the Response Style template to be used for composing the response page when a visitor submits his vote. See also the note on the **Results** Parameter
[/statusmsg]	When a visitor submits a completed voting form, if some fields which are required are missing, the voting form can be redisplayed using this template. This field will be replaced by the message specified in the Security page.
[/dataerror]	Used with statusmsg, above. When an incomplete form is received, this tag will be replaced by the indicated string, typically a pointer such as =>. This tag should be placed next to statusmsg above and also right before each question, so it can point to the questions that need answering.
[/originanswer]	Used with statusmsg, above. This tag must be placed in the value parameter of the [/text] section, and as the displayable text for the [/mtext] section above. When a visitor submits incomplete data, these tags allow the display of their answer to questions of these types. Other question types do not use this tag.
[/vfonclick] [/endvfonclick]	Used around JavaScript used to skip between questions and to prevent selecting both a choice and a write-in value. If neither of these is used, all text between these tags is removed.
[/ifonclick] [/endifonclick]	If any question uses [/vfonclick], the JavaScript bracketed between these tags will be included; otherwise it is removed.
[/ifremaining,UNIT,OP,value] [/endifremaining]	Provides a way to show, live, the remaining amount of time still open to vote in a poll or survey. UNIT is "days", "hours", "minutes", "seconds". OP is one of "eq", "le", "lt", "ne", "gt", "ge". The HTML between these tags is shown only when the remaining unit of time has the indicated relationship to the value. For example, [/ifremaining,minutes,gt,1] will show until the last minute before the polls close. To test if a poll is closed, use [/ifremaining,seconds,eq,-1]. To test for a poll that is open without a closing time, use [/ifremaining,seconds,eq,-2]
[/remaining,UNIT]	UNIT is "days", "hours", "minutes", "seconds". Shows the specified units of time. For example, [/remaining,minutes] will show a number between 0 and 59. Combined with [/ifremaining], above, virtually any type of countdown display is possible. These tags can also be used in response pages.
[/extra0] through [/extra9]	When a question uses Extra fields, these tags display as their content, which are typically names of image or multimedia files.

[/charsetname]	Contains the character set specified in the **Language** page.
[/thispagenumber]	The number of the page currently shown while taking a survey, beginning with 1.
[/numberpages]	The number of pages in the survey.
[/percentcomplete,scaledto]	The percentage of the survey which has been completed, scaled to the given number, when present. This allows constructing a progress bar of a known maximum size. If scaledto specified, the number starts at 1 rather than 0. The last page in a survey will have a number less than 100.
[/ifmultipage,not] [/endifmultipage]	Provides a way to show something, like a progress bar, depending on whether a survey is on multiple pages. Use [/ifmultipage,not] to see if a survey is all on one page.
[/lookupi18n,xxx]	This tag will be translated using the contents of the appropriate file in the WEB-INF/classes/com/cogix/vwflanguage directory. The Text.properties English file will be used by default. If the pollid ends in __xx (two underscores and a two digit language code, such as __fr), then Text_fr.properties will be used to look up French. See **Internationalization**.
[/lookupi18n,languagei18n]	A special form of this tag returns the suffix of the pollid, such as fr. This can be used to localize elements such as image references, such as img src="Next[/lookupi18n,languagei18n]].gif", which will refer to Next.gif and Nextes.gif for an English and a Spanish questionnaire.
[/taglang]	The language field in the Language settings page
[/tagdir]	Replaced with dir="rtl" when right-to-left is set in the Language settings page

Additional results parameter values

Normally, this field contains the [/results] tag, which ViewsFlash replaces with the file name of the response page template it creates. In certain special situations, you can hand-code this field with the following values:

Content	Meaning
page.html or dir/page.html	Use this specific response page, relative to the root directory of the ViewsFlash web server. Always use / for directories. Do not use a leading /.
/dir/page.html, or C:\dir\page.html	Use this specific response page, at the file system location specified by this absolute path (as opposite to a relative path, above). Use leading / for Unix and leading C:\ for NT (or whatever drive letter is appropriate).
http://domain/page.html	Use the response page at this absolute URL. This allows for the results template to be on a different machine. Even though the http request is cached, it is not as fast as specifying a file (which is also cached).
=text	Use "text" as the response template. You can use ViewsFlash tags in the text. An easy way to test things.
?url	Redirect to this page after processing the vote. For example, ?http://www.yahoo.com displays the Yahoo home page after voting.

Custom Tags

As you examine the template, you will see that it uses [/custom] tags, described below, and that it has an architecture of nested embedded tags. This structure allows it to be used with all question types. If you create a template with certain sections removed, such as the [/mtext] section, then the template cannot be used with polls that ask that type of question. Generally, it's best to minimize the amount of change in the templates.

Next: **Multiple page tags**

Form style tags for multiple page questionnaires

As you read this section, you may want to open the **Standard** template and View its Source.

Templates for multi-page surveys are the same as Poll Style templates, except as noted here. To create your own, it's easiest to start with this one and modify it as needed. First, some general considerations:

The following hidden fields must be present, just like in the Standard template: cmd, pollid, results, pagenumber.

Each question type must include the [/originanswer] tag, which allows showing the visitor the answers they have entered so far when a page is shown again.

The next, previous, save, and submit buttons can be styled as you like. The Poll page check boxes control whether they should be used at all. When selected, they are expressed by the [/ifpage]... [/endifpage] tags. Each button can be a form button or an image button. Use HTML like the following:

<input type="submit" name="submitprev" value="Previous">

<input type="image" name="submitprev" src="Prev.gif">

The four buttons **must** be named: submitprev, submitnext, submitsave, submitsubmit.

Furthermore, you can construct the template so that the buttons only appear on the first or last pages, as appropriate. Surround the button between [/ifpage] and [/endifpage] tags.

Finally, the [/qskipper] [/endqskipper] and [/qskipto] should not be used, as questions are skipped to by simply never displaying the skipped questions.

Tag	Replaced with
[/pagebreak,XX] [/endpagebreak]	If XX is the letter 'p', what's between the tags will be expressed only during page preview. This is how the Page Break lines appear. If XX contains the letter 'r', the content will only appear during an actual survey. If XX contains "pr", the content will appear during both preview and in the real survey.
[/ifpage,AAA,WWW,not] [/endifpage]	The content between the tags will be expressed when button AAA is enabled, and the page is the first page, the last page, or a middle page. AAA must be one of: previous, next, save, submit. WWW must be one of: first, last,middle. not, if present, means express the content between the tags when the condition is false. For example, [/ifpage,previous,first,not]...[/ifpage] shows a Previous button only on all pages but the first.

Summary of button values:

Action in [/ifpage tag	button name	sample image
previous	submitprev	prev.gif
next	submitnext	next.gif
save	submitsave	save.gif
submit	submitsubmit	submit.gif

Next: **Response style tags**

Response style tags

As you read this section, you may want to examine the **Results** style in a separate window and View its Source.

Tag	Replaced with
[/statusmsg]	Used by ViewsFlash for designer error messages.
[/pendingerror]	Used by ViewsFlash for Visitor Error messages such as "you have voted twice".
The following tags take their values from the Design Response Page screen:	
[/reporttitle]	Report's Title
[/reportheader]	Report's Introductory text
[/reportfooter]	Report's Closing text
[/allquestions] [/endallquestions]	All HTML between these tags will be repeated, once for each Question.
[/qtext]	The question's full text
[/qname]	The question's name
[/allanswers] [/endallanswers]	All HTML between these tags will be repeated, once for each Answer to the current Question.
[/valuelist][/endvaluelist]	Instead of [/allanswers], these tags use the actual values entered in voting forms, but can only express the Individual Field tags (**see below**).
Between the allanswers and endallanswers tags, the following tags take their values from each predefined Answer to the current question	
[/atext]	The answer's full text
[/avalue]	The answer's value
The following tag pairs describe what HTML to use for each different question type:	
[/ashowpctbar] [/endashowpctbar]	HTML in between will be used only when "Display Percentage Bar" is chosen.
[/ashowpct] [/endashowpct]	HTML in between will be used only when "Display Percentage" is chosen.
The following tags are replaced with the tags shown, when the corresponding fields in the Design response page screen are checked. See **Individual Fields** tags for the meaning of the tags and ^ and *.	
[/acount]	[/^,count,*]
[/apct]	[/^,xpct,*]
[/aavg,DD,MM]	[/^,avg,,DD,MM] (DD and MM are optional)
[/arank]	[/^,rank,*]
[/aprevrank]	[/^,prevrank,*]
[/apctbar]	<img src="/viewsflash/pixelBlue.gif" height="10" width="<vwf=fieldname,pct,fieldvalue>">
Individual Fields in response pages	
The following Individual Field tags are replaced as follows.	
^	The current Question Name, when used between [/allquestions] [/endallquestions] or [/talliedlist] and [/endtalliedlist].
[/fieldname]	The current question name.
*	The current value, when enclosed between [/allanswers] and [/endallanswers], or [/valuelist] and [/endvaluelist]
In the following descriptions of additional Individual Fields, we use as an example a Question named "Party", with values of "Democrat" and "Republican"	
[/Party,count,Democrat]	How many people chose Democrat as the answer to Party.
[/Party,pct,Republican,SS]	The percentage of visitors who answered Republican to the question Party. Scaled to the number SS (if that parameter is present). Always rounded up, never less

	than 1. This is the best tag for scaling images. If SS is less than 0, then it is interpreted as the number of decimal places to show instead of a scaling factor.
[/Party,xpct,Republican,SS]	The percentage of visitors who answered Republican to the question Party. Scaled to the number SS (if that parameter is present). Never rounded, can be 0. If SS is less than 0, then it is interpreted as the number of decimal places to show instead of a scaling factor.
[/Party,negpct,Republican,SS]	Same as pct, but subtracted from 100.
[/Party,pctq,Republican,SS]	Same as pct, but calculated against the number of people who answered the question, rather than the total number of voting forms received. Can be zero. If SS is less than 0, then it is interpreted as the number of decimal places to show instead of a scaling factor.
[/Party,pct100,Republican]	Same as pctq above, but manipulates percentages so that they add up to 100 for a given question. Can be zero.
[/Party,rank,Democrat]	The rank of people who voted Democrat in question Party (1,2,3,etc.)
[/Party,prevrank,Democrat]	The rank of people who voted Democrat the last time the "Set Rank" operation was used.
[/QuestionName,avg,DD,MM]	The average of the answers to the question; only meaningful for numeric values. If DD included, show DD decimal places. If MM is included, all results are multiplied by MM.
[/QuestionName,value,*]	Displays the current answer's value.
[/QuestionName]	The value that a visitor entered for this question in a vote, such as their name.
[/QuestionName,enteredanswertext]	The full text of the answer chosen by the visitor
[/ifmeasure,QuestionName,Answer, MM,OP,value] [/endifmeasure]	MM is "rank" or "pctq". OP is one of "eq", "le", "lt", "ne", "gt", "ge". The HTML between these tags is shown only when, in QuestionName, the Answer's rank or percentage has the indicated relationship to the value. Use * for the current answer. For example, [/ifmeasure,Party,*,rank,eq,1] will only show the top-ranked party.
[/ifresponded,QuestionName,Answer] [/endifresponded]	The HTML between these tags is shown only when the visitor responds Answer to QuestionName. For example, [/ifresponded,Party,Democrat] for the people [/endifresponded] will display "for the people" only for those who marked Democrat for their Party. ^ and * may not be used. When Answer is omitted, as in [/ifresponded,QuestionName]...[/endifresponded], then the actual content entered by the respondent is displayed.
[/ifdefault,QuestionName,Answer,X] [/endifdefault]	If the fourth parameter X is missing or 1, the HTML between these tags is shown only when this Answer is the default answer to this question. If the fourth parameter is 0, the HTML is shown when this Answer is not the default answer to this question.
[/ifremaining] [/endifremaining] [/remaining]	These tags, dealing with the time remaining until a questionnaire closes, are also available in questionnaire pages. They are described in **Form Style Tags**.
[/extra0] through [/extra9]	When a question uses Extra fields, these tags display as their content, which are typically names of image or multimedia files.
[/QuestionName,answerextra,Answer,N]	N is a number between 0 and 9. Displays the extraN field associated with the Answer to the QuestionName. For example, [/Party,answerextra,Democrat,0] will display the "extra0" field, which might be "donkey.gif". This tag is automatically created by the [/extraN] tags.
[/QuestionName,ordinalentry,Answer]	The ordinal number of Answer in question QuestionName, starting with 1. Can also be used as [/^,ordinalentry,*]
Additional tags:	
[/pollcount]	Total number of responses received
[/ifpollcount,N] [/endifpollcount]	Until the number of responses received reaches N, show nothing. Above that, show the HTML between these tags, usually [/pollcount].
[/ifpollcount,N,M] [/endifpollcount]	Show the HTML between these tags when the number of votes cast is between N

	and M, inclusive. More than one set of these tags can be used.
[/pollcolor,CC,DD,EE,...]	Rotates between strings CC,DD,EE, during iterations among answers. Usually used for changing color bars.
[/randomnumber]	Generates a random integer such as 45521. Useful for ad engine URLs.
[/starttime]	Date and time questionnaire opened
[/currenttime]	Current date and time
[/datenow]	Current date
[/timenow]	Current time
[/elapsedtime]	Days and hours questionnaire has been opened
[/pollstatus,code,message]	If the status of this questionnaire matches number "code", display this message. Only meaningful when using the cmd=getresults API.
[/charsetname]	Contains the character set (charset, encoding) specified in the **Language** page.
avaluex, atextx, qtextx, qnamex, allquestionsx, endallquestionsx	Same meaning as the corresponding tags without "x", but values are shown regardless of the settings on the Response page.

Custom Tags

In addition, Form and Response templates can be extended with Custom Tags. By using these, it is possible to let the Editor who creates the questionnaire determine the look and feel of those attributes you want to let them have control of, such as the font color or size, for example.

The special {/ syntax

When you study the templates, you will find an occasional set of tags like this one found in the XtabReport style:
{/[/depqname,force],count]

To understand how this works, keep in mind that a results template is first interpreted when the questionnaire is Published. At that time, the {/ sequence is translated into [/ and [/depqname,force] is interpreted to its meaning, which in this case is the name of the dependent question being analyzed; if that were "performance", for example, then the tags would now look like this:
[/performance,count]

So, simply put, use {/ when you want a tag to be interpreted after Publishing rather then before.

Next: Archive style tags

Archive style tags

As you read this section, you may want to examine the **ListArchive** in a separate window and View its Source.

Tag	Replaced with
[/statusmsg]	Used by ViewsFlash for designer error messages.
[/pendingerror]	Used by ViewsFlash for Visitor Error messages such as "you have voted twice".
The following tags take their values from the Design Response Page screen:	
[/spotname]	The place name
[/pollresults]	The currently viewed poll will be shown here, using the place's Response Style. If the query string includes &showallresults=1 and this tag lies within [/allshownarchivedpolls] and [/endallshownarchivedpolls], the results of every poll in the archive will be displayed.
[/allshownarchivedpolls,order] [/endallshownarchivedpolls]	Everything between will be repeated for all Archived polls that are marked for Show. The 'order' parameter, if included can be lowtohigh or hightolow; if omitted, polls are shown oldest first.
Between the allshownarchivedpolls and endallshownarchivedpolls tags, the following tags take their values from each poll shown:	
[/shownarchivedpolltitle]	Title of the poll; if none, the poll name will be used
[/upollname]	The poll place, encoded for use in a link.
[/dateopen,dateformatstring]	The date the poll opened. The second argument is optional and can be any standard Java date formatting string, such as dd-MM-yy which is the default.
[/questiontext,questionname]	The text of the question named 'questionname' or the text of the first question if questionname is omitted.

Next: Question styles

Creating question styles

Question styles are entered in the Styles page by an administrator who has that access right. A question style uses the **Form style tags**. However, because it will be embedded within a page, it omits <head> and <body> sections. Make sure that all question types that can be used with this template are defined, such as radio buttons, checkboxes, etc. Study the Question Styles provided.

The Standard_ style include all the necessary JavaScript for validation. To create your own question styles, you must name them Standard_yourstylename. Examine the Standard_ style and note how it creates a ValidatorArray and includes an onSubmit function. Their purpose is to provide a way to check that entered values are correct before the form is submitted, and to cancel the submission if they are not.

Each Question Style includes JavaScript that adds a uniquely named function to the ValidatorArray, which performs the desired checks and displays appropriate Alerts.

To examine some existing styles, use the Styles page and examine the Standard_Number and Standard_Date styles.

Question properties

Styles use question properties to specify additional attributes of a question style, such as the format of a Date or the format of a Number. To use question properties, use the qpropertynames and xmlqproperties sections inside the comment section. Please open and print the Standard_Number and Standard_Date styles for an example, and create a question and assign the Number and Date styles to it to see how question properties work.

The qpropertynames section lists the names of the properties to be defined, separated by commas.

The xmlqproperties section is an XML fragment, beginning with an xml tag, and one element, xmlelements. Within this element, there are two elements: styledescription, which will describe what the style does under the "Select Style" drop down, and questionpropertiesform. This second element surrounds an HTML fragment in the XMLdelimiters <![CDATA[and]] so that the HTMLfragment will be interpreted correctly. The HTML fragment consists of form input fields whose names match the names listed in the qpropertynames section, and whose value expresses the value entered.

For example, this tag displays the "numberdecimals" property as an input field:
<input name="numberdecimals" tabindex="21" type="text" id="numberdecimals" value="[/qproperty,numberdecimals,0]" size="1" maxlength="3" />
The value field tag [/qproperty,numberdecimals,0] indicates that the current value of the property should be displayed, and that if there is no current value, the value 0 should be displayed.

In this example, a check box toggles the "include_in_report" property on and off:
<input type="checkbox" id="include_in_report" name="include_in_report" value="checked" [/qproperty,include_in_report,checked] />
Note that the value must be "checked".

See also **Using Question Styles**.

Next: Custom tags

Custom tags (customa - customz)

If you examine the Standard_ and Results style templates, you will find tags named [/customa],[/customb], etc.. When the template is used to create a questionnaire or response page, these fields are replaced with certain user-defined values. These values are set in the Questionnaire, Response, or Place pages.

The function and default value for each of these tags is defined in a section at the end of the template, in a section bracketed by <vwf=spotform> and <vwf=endspotform> tags. The HTML between these tags is not included in pages created with these templates, but rather is displayed in the place Settings page. Thus, when a place is designed, these custom settings, which are used to define font, size, colors, etc., can be set once and applied to all questionnaires in the place right from this page.

Similarly, on the Results template, there are <vwf=spotform> and <vwf=endspotform> tags. The HTML between these tags is included in the response page, where the questionnaire designer can change attributes directly.

It is also possible to move the HTML in both templates to the section bracketed by <vwf=pollform> and <vwf=endpollform> or <vwf=rsltform> and <vwf=endrsltform> . In this case, the HTML appears in the Setup / General and Setup / Response pages instead, and each questionnaire can have unique settings. It is even possible to split the HTML between these two sections, effectively making some options apply to all questionnaires in one place and letting other options be customizable in the individual questionnaire.

In all these sections, custom tags may include a second argument, which is the default value for that field, and is recommended.

When a questionnaires or a place is copied, its custom values are copied as well.

If a place has custom settings, but they are no longer used because the template you are now using doesn't have any, a checkbox appears in the Place page that allows you to remove any prior values that may be lingering around. The same applies to questionnaire pages.

The [/comment] and [/endcomment] tags are used in these sections to allow inserting instructions for the ViewsFlash designer. When a page is created from these templates, everything between these tags is removed.

So, when you create a place using this template, the Setup / General page now includes these questions which define the customa through customz fields and their meaning in this questionnaire. Thus, the person who is designing a survey now has the option to modify the colors, font type and size, etc., without having to either modify the style template or the settings on the place. Furthermore, each Style Template can include its own uses of the customa through customz fields, independently of any other Style templates, making for an extremely flexible scheme while reducing complexity to the questionnaire designer.

Next: Styles How to

Styles How to

This section provides snippets of HTML that you can use in your Style Templates to achieve various effects.

Form Styles **Response Styles**

Form Style Templates:

	To show results in a pop-up window after a vote, like this: 
	In a copy of the Standard_ template, modify the <form> tag to read: <form method="get" action="[/vwfservlet]" target="resWin" onSubmit="window.open('','resWin','resizable=yes,scrollbars=yes,width=300,height=300')" > Note that method="get" is preferred for short forms.
	To use a graphic vote button, like this: 
	In a copy of the Standard_ template, find the line that defines the Submit button, and replace it with: <input type="image" src="http://.../VoteButton.gif" name="image" width="47" height="21">
	To create a link that takes you right to the Edit Style Template page in the Poll Page
	In a copy of the Standard_ template, between the <vwf=pollform> and <vwf=endpollform> tags, add: <tr><td colspan="3" align="right"> Edit Style Template</td></tr>
	To show the time remaining to vote
	Add the following to your template. Many variations are possible. This displays either "3 minutes remaining", "1:34 remains", or "15 seconds remaining", as closing time nears. This should be used with the cmd=getpoll API. [/ifremaining,minutes,gt,1] [/remaining,minutes] minutes remain [/endifremaining] [/ifremaining,minutes,eq,1] 1:[/remaining,seconds] remains [/endifremaining] [/ifremaining,minutes,lt,1] [/remaining,seconds] seconds remain [/endifremaining]

Results Style Templates:

	To show multiple result color bars using a graphic instead of a colored bar, like this 
	The EmbeddedResults template does this. If you need a whole page template rather than a fragment, copy this template and add: <head></head><body> at the top and </body> at the bottom, plus any other enhancements. Then change the template in the place, and go to the response page. Change the color bar default colors to a combination of Red, Green, Blue, Black, and Mag. See also the next tip for more information.
	To show the result bars using a graphic instead of a colored bar, like this:  or like this: 
	In a copy of the Results template, replace the HTML between [/ashowpctbar] and [/endashowpctbar] with: <td><img src="http://..../PurpleBar.gif" width="[/apct]" height="20" ></td> or: <td><img src="http://..../pixelRed.gif" width="[/apct]" height="10" ></td> Note that PurpleBar is 20 pixels high by 1 pixel wide, but pixelRed.gif is only 1 pixel high by 1 pixel wide. This takes advantage that browsers expand the image to fill the designated area. Use an absolute URL to the ViewsFlash server for best performance.

	To show a graphic next to the top 3 vote getters
	Add the following to your template. This will display crown.gif only for the top 3 contenders. In the response page, choose Most Popular First. [/ifmeasure,^,*,rank,le,3] <img src="http://.../crown.gif">[/endifmeasure]
	To show the time remaining to vote
	Add the following to your template. Many variations are possible. This displays either "3 minutes remaining", "1:34 remains", or "15 seconds remaining", as closing time nears. [/ifremaining,minutes,gt,1] [/remaining,minutes] minutes remain [/endifremaining] [/ifremaining,minutes,eq,1] 1:[/remaining,seconds] remains [/endifremaining] [/ifremaining,minutes,lt,1] [/remaining,seconds] seconds remain [/endifremaining]
	To show the vote count only after it gets to be larger than 100, like this:100 votes
	In a copy of the Results template, add the following table row near the bottom of the table: <tr align="center"><td>[/ifpollcount,100][/pollcount] votes[/endifpollcount]</td></tr>
	To include a Close Window message in a pop-up, like this: **Close window**
	In a copy of the Results template, add the following table row near the bottom of the table: <tr align="center"><td> Close window</td></tr>
	To create a link that takes you right to the Edit Style Template page from the Results page:
	In a copy of the Results template, between the <vwf=rsltform> and <vwf=endrsltform> tags, add: <tr><td colspan="3" align="right"> Edit Style Template</td></tr>

About Cogix ViewsFlash™

The following libraries are included in the distribution:

ChartDirector.jar - licensed software distributed according to redistribution agreement with Advanced Software Engineering, Hong Kong

activation.jar - The JavaBeans Activation Framework under redistribution license from Sun Microsystems.

mailapi.jar, pop2.jar, imap.jar, dsn.jar, smtp.jar - Javamail, under redistribution license from Sun Microsystems.

bsh.jar - The BeanShell Lightweight Scripting For Java library, distributed by www.beanshell.org.
This library is distributed according to the terms of the GNU Lesser Public License (LGPL):
http://www.gnu.org/copyleft/lesser.html
ViewsFlash uses an extended version of the BeanShell 2.0b4 library. The source code for the extended library is available from:
http://www.cogix.com/public/BeanShell.zip
The extended library is also available subject to the terms of the GNU Lesser Public License.

ViewsFlash includes the TinyMCE editor library under the LGPG license.
See **TinyMCELicense.txt** for complete license text

ViewsFlash includes the Jazzy spell check API library jazzy-core.jar
under the LGPL license. See **JazzyLicense.txt** for complete license text.

ViewsFlash includes the Scowl American English word list word list, copyright 2000-2004 by Kevin Atkinson and others.
See **ScowlLicense.txt** for complete license text.

ViewsFlash includes the XStream library under a BSD License
See **XStreamLicense.txt** for complete license text.

ViewsFlash uses in its examples flags from IconDrawer, using a commercial license.
See **IconDrawerLicense** for complete license text.

Administrator's guide to upgrading to Release 5

Here's what's new in this release.

- New drivers for Oracle 9 and 10 are now included. Existing drivers continue to work, but these new drivers are recommended for future compatibility and performance.
- The DB2 drivers have been revised slighlty.
- A new family of styles, called Standard_, are now the default. These are new, accessible styles which also work in portals. Complete details on what to do are in the **Upgrading Styles** section.
- Installation, using a database, and clustering documentation have been revised and simplified.
- In Weblogic Server, it is now possible to deploy the ViewsFlash.war file without exploding it if ViewsFlash is running off a database. This is specially useful in cluster deployments. CAUTION: if ViewsFlash is not running off a database, you MUST deploy from an exploded war file; if you upgrade from the war file, when ViewsFlash is not running off a database, Weblogic Server will delete all the files written by ViewsFlash to the file system, requiring that every questionnaire in use be Published again.
- Application initialization now takes place during application server startup, rather than later, making it easier to diagnose installation errors.
- Data is now saved in questionnaire tables before the survey response page is shown to the user. While this introduces a small delay, it lets the user know immediately when a database error occurs. In such as case, live tallies are not updated with the record that could not be captured.
- Log files now highlight all errors in red for easier viewing.
- The servlet parameter **appcharset** now defaults to UTF-8, allowing the application to be used in any language by default.
- A new servlet parameter **usedefaultcharacterencoding**=true should be used when application servers, such as Dynamo 7, decode request parameters and form submissions.
- Secure and anonymous questionnaires can now be done by simply selecting Basic authentication or Cookie authentication in the **Security** settings.

Upgrading to ViewsFlash 5 styles

If you are not upgrading from a previous version of ViewsFlash, ignore this section.

Styles are not upgraded automatically when a new version of ViewsFlash is installed. Perform the ?Diagnose=1 command, which will display a link that you can click on to upgrade the styles listed. If you have revised any of the Cogix provided styles, the upgrade will erase the revised styles. The link that Diagnose=1 link, that you can just click on, looks like this:

Use **?cmd=upgradestyles** to update these styles, which need to be updated:
Standard_.html NEEDS UPDATING
Index.html NEEDS UPDATING

As of release 5, a new form style template, Standard_, is the new default style for constructing questionnaire forms. This style is complemented by the question styles named Standard_Matrix, Standard_Email, Standard_Number, Standard_Date, etc. New features such as question help, section 508, XHTML and are supported only in these new styles. However, existing styles such as Questionnaire, Email, Number and Date, and Portlet_Polls continue to be supported. Existing Places and Styles continue to work in ViewsFlash 5 unchanged, and new questionnaires can continue to be created in the existing places with the existing styles.

Using the new capabilities requires creating a new Place and using the new Styles, Standard_ and Results. These new styles work both in portals and non-portals. This change consolidates the standard, accessible, and portal styles into a single style and allows for simpler style management.

When a questionnaire is designed using a particular form style, only those question styles that are compatible with it are shown in the create question page. If a questionnaire is copied from a place that uses an incompatible style, the question styles used are changed automatically.

This change also affects the **Question Library**. The ViewsFlash 4 question library, named Question_Library, uses the Questionnaire style, and items in the library use question styles that are compatible with Questionnaire, such as Ranking. In ViewsFlash 5, a new library, called Q_Lib, that uses the new Standard_ styles, is installed automatically. Multiple question libraries are now supported; the question library menus display items from those question libraries that use styles that are compatible with the styles used in the questionnaire's place.

Next: **Introduction**

ViewsFlash 4.05 Release Notes

Release 4.05 simplified deployment and optimized database use in portals. Problems with updating data and user id's that contained apostrophes were resolved.
To deploy quickly, start by reading **Installation**.

In Portals, the Survey Portlet can now display all surveys that a user is elegible for, and surveys can be marked as available to everyone in the portal or use invite lists to determine who is elegible.

Release 4.01 incorporates three new Styles, named Accessible_, Accessible_LikertHeader, and Accessible_LikertBody, for creating questionnaires that comply with WCAG Priority 2 and Section 508 accessibility guidelines, including XHTML 4.01 strict. For how to use these, see **Accessibility**.

Release 4.0 incorporates two valuable new features that support using ViewsFlash as a Forms Engine: a **Script** language for branching, calculating, and scoring, and the ability to enter data into a form and to allow the user to **revise the entry** later by simply returning to the same form URL. This features is perfect for keeping and updating user profiles, for example. Note that live tallies count only the first entry, but analytical tools such as Univariate and Cross tabulation analysis analyze the modified entries.

For portals, a new Survey portlet 1.1 allows a portal visitor to see a personalized list of the surveys that he is supposed to participate in, and to click on the survey to take it. To support this, Invitation Lists can now be constructed with or without an e-mail address, and the portlet uses the list of user id's to determine who is eligible to participate in a survey.

Many international Cogix clients run their applications in a default language other than English. To change the default language of ViewsFlash, see **Localization**. This release also adds **i18n style tags** which translate the Next, Submit, and other buttons and other language-specific survey elements when using multiple language **surveys**. Existing styles need to be modified only if you want to take advantage of the new [/lookupi18n] tags.

DB2 support has improved with more careful use of varchar fields and support for long text fields.

New Custom Actions include SQLSelect, SQLUpdate, and Script. The first two allow a survey or form to lookup data in a database table to pre-fill questions and to write data gathered in the form to a table. Script allows calculations. When using User Security, the right to use Custom Actions can be controlled by the Administrator now.

Question Styles now include Date, Number, Email, which are used to validate text fields. New styles are added to the database automatically when the ?Diagnose=1 command is done after upgrading. The older styles CheckNumericText and CheckValidEmail are superseded by Number and Email, but they will continue to work if you are using them.

New **servlet parameters**:
"datareviewapi" is now required to activate the data review and modification API.
"appcharset" defines the application's default character set. Useful when the native locale is not English. The default is "UTF-8".

New license keys:
Release 4.0 requires a new license key in a new format. Contact sales@cogix.com before upgrading for an updated license key.

Release 3.9 supports Question Library, Clipboard, Staging, Reviewing and Updating Saved Data, and other new functionality

The **Upgrading** instructions have been improved substantially. If upgrading from an earlier version, please read them now!

A host of new question management functions are provided. See **Question Library and Clipboard** for their description. A new option in the Questions page allows renaming one or many questions. A question's answers can now be **uploaded** from a CSV file. A new option in the place page, **Append to Survey**, allows a survey to be put together from other surveys.

A **Staging** capability allows for running the software in a staging server and interfacing to staging systems. It can also be used to transfer surveys from one server to another.

A new question style, Ranking, allows easy creation of "rank items in order of importance" matrices. The style is included in the download or can be installed into earlier versions from the **Style Library**. Standard style templates have been fine-tuned for cleaner HTML; no functional changes are included, but please review **Upgrading Styles**. The Response style bars are now displayed using tables instead of images, giving better appearance and performance. The revised styles allow setting the **maximum** number of characters in a text field, and the number of **decimal places** to show in percentages in results pages.

Cookie authentication, when used without a cookie name, now looks for the User ID in a "ViewsFlash" cookie, and if one is not present, throws one with a random number for the user ID. This allows using the Save and Resume capability in large anonymous multiple page surveys; without this, some other form of authentication was required (session authentication could fail if the user left a survey open for more than a half hour). In places with rotating polls, a cookie that expires when the poll is closed is used.

New **Security** options allow protecting a survey with SSL Secure Socket Layer encryption (https://) and checking the "referer" header for specific content. The secure URL must be entered in the **Administration** page.

New options for showing **archived poll listings** allow showing archived polls in newest to oldest order, listing the poll opening date, and including the text of a poll question.

New **Data Review and Modification** APIs are available for system builders. These APIs allow reviewing survey data that has already been committed to its database and for editing that data are now available. These APIs do not support the Normalized data base format.

A new JSR168 compliant Survey Portlet is available as a separate download, for Portals which support this specification. A WebSphere Portal SDK spells out the required steps. Other SDKs are in preparation, but the JSR168 portlet should work with other portals. This portlet requires release 3.9 or later. Details at **www.cogix.com**.

Bugs fixed. In heavy traffic situations, Custom Actions that relied on ThisCall or setParameterValue failed because of concurrency problems, sometimes using the values from another request. Multiple problems when using Others and check box questions have been fixed. The **storeflushtime** servlet parameter now has a default value of 0. A database cursor leak has been removed.

Univariate and Cross tabulation analysis has been enhanced for multiple choice checkboxes. Percentages are now calculated against the total number of respondents who answer the question. For example, 62% can say they like sports and 75% like news. Previously, percentages were calculated relative to each other, so the numbers might have been 40% and 50% instead. Some bugs in tabulating and saving data entered in the Others section of a multiple choice question were fixed as well.

Release 3.8 is a major security enhancement release.

Application security now includes User Security, where users have Access Rights to places, styles and invite lists. For example, certain users may be able to create surveys in a place, while other users can analyze the data and yet other users can examine the raw data collected. The ViewsFlash Administrator is responsible for managing security throughout the application. The completely revised **Application Security** section describes all options available in detail. A new section on **Extensible Authentication** explains how to use custom authentication methods and custom user/role lookup methods, including an LDAP server and a properties file.

A new Where Used option in the Styles and Invite Lists pages allows Administrators to see where every style and invite list is used. When using User Security, additional options on these pages and the places page allow the Administrator to grant appropriate Access Rights to users and roles.

The Custom Action Validator class has a new method, **substitutePipedParameters**, which allows using Question Piping.

Invitation lists can now use an LDAP server as the source of people to invite to a survey. See also **Using LDAP** for configuration information.

Release 3.7.2 is a release for using surveys as ATG Dynamo Gears.

Unless you're very technical, skip down to **Release 3.7** release notes.

This release incorporates new features for running ViewsFlash surveys and polls as **ATG Dynamo Gears**. These include additional Web Services API functions. **cmd=getpollnames** has a two new parameters, title=1, which is used to retrieve survey and poll titles, and smp=1 which retrieves an indication of whether the poll is a single or multiple page survey. **cmd=ifvoted** has a new parameter, verbose=1, which returns the appropriate error text from the Security page. Two new parameters can be used in a ViewsFlash query string, typically when including ViewsFlash output in JSP pages: outputtorequestattribute= is used in JSP pages to write output to an HttpServletRequest attribute instead of the PrintWriter; and nooutputstream=1 makes ViewsFlash use an already opened PrintWriter.. Another new servlet parameter, **apiattribute**, adds security when the ViewsFlash api is used from JSP pages. Finally, the onRequest method in RequestNotifier can now throw a ServletException if under duress.

Release 3.7 is a feature release.

This release focuses on using e-mail **Invite Lists** to **Invite** people to take a survey and to send them e-mail reminders that they haven't completed a survey yet. An invite list can be a comma-delimited file or a database query, and can be previewed

in the browser or in Excel. An Invite can embed the survey itself right in the e-mail, or contain a link to the survey. When used with Authentication, survey progress is tracked and reminders can be sent selectively to those who haven't started a survey, to those who haven't finished yet, or to everyone. The invite list can be dynamically redefined during the course of a survey, allowing both inviting different groups of people at different times, or adding new people to the list even while the survey is open and e-mailing only new people on the list. By default, people not on the list cannot participate in the survey, but this can be overridden.

A new **Authentication** option, Form, complement the tracking capability. It displays a login form at the beginning of a survey, which are checked against the user ID and password in an Invite list.

A new page for diagnosing authentication problems better is included. If you are upgrading, review the Master poll and in the Security page, in part B of Authentication, enter in Display the following response page: viewsflash/unauthorized.html

Application server container managed security is now supported fully. A new servlet parameter, appsecurity=servlet, is used with new options in web.xml to provide two URI's for the viewsflash application. A public URI, servlet/viewsflash, is used for taking surveys and can be set up without authentication. An application URI, servlet/vfadmin, is protected using container managed security. See **Application Security** for how to set up this powerful option.

Other improvements include:

The ViewsFlash Administrator, using the **Administration** page, can now allow anyone to edit Styles and Invite Lists, which are normally only editable by the Administrator. **Styles** can now be entered by uploading a file as well as using cut and paste.

The two options for removing data and results in the **Publish** page have been combined into one, which now also allows sending Invites again to a list after a test period. A new option allows discarding partially completed multi-page surveys when the survey closes.

Simplified user interface. When using a database, options in the place Settings page that refer to writing HTML to disk are now removed, unless a **new servlet parameter**, writehtmltodisk=1, is used. Similarly, the seldom-used Webcast features are available only when the new servlet parameter webcast=1 is used. In the Security page, the Track duplicates in the database option is now used automatically when needed, so it has been removed from this page.

Database improvements. During servlet initialization, all ViewsFlash application database tables are created automatically and altered, if necessary, to bring them up to date automatically. However, there is one exception: when saving data in a Normalized table, the VWFDATA table index VWFDATAINDEX is no longer needed and should be removed manually. The application will work fine with or without it, but removing the index will mean less work for the database.

The menus on the Styles and places pages have been revised to make it easier to create, copy, and remove them. A new option in the Polling Places page allows very quick creation of single question polls.

If you are upgrading: see the notes above on Authentication and viewsflash/unauthorized.html, and start using that response page in all new surveys that use authentication. See also the note above about Database improvements and removing the VWFDATAINDEX from table VWFDATA. Review the new capabilities in Invite Lists and Invite. And don't forget, you can always call us for further explanation!

Release 3.6 is a maintenance release.

This release adds stronger database error recovery from dropped connections, an Others option in checkbox questions, an option to **randomize** the answers to a multiple choice question, and new Web Services API calls to retrieve all place names and to present **multiple randomly sampled** surveys. It also added support for MySQL and IBM DB2 UDB and on AS400 DB2 V5R1.

Release 3.5 is a feature release.

The focus of this release is increased complex survey functionality and administration, including dynamic question skipping and next page selection, question validation with JavaScript, copying and rearranging multiple questions, more powerful Custom Actions, data analysis with "Others", improved error reporting, more flexible support for databases, and new Style tags. Details:

- A **complete example** of Validation, Branching and Scoring, and using the new Index is included.
- A usability review simplified the user interface throughout.
- Excel XP is now supported (some functions only worked with older versions of Excel)
- A new Questionnaire Style Template is used for creating multiple page surveys. It incorporates a progress meter, page title, header and footer, a presentation-quality look for creating fully professional looking surveys out of the box, and JavaScript **Data Validation**.

- Revised **Style Templates** include Standard_, ResultsPage, QuizResults, UnivariateReport, XtabReport.
- New **Question Styles** include LikertHeader, LikertBody, CheckAlphaNumericText, CheckNumericText, CheckValidEmail, ColumnBegin, ColumnBreak, and ColumnEnd.
- The following templates are deprecated: MultiPageSurvey (superseded by Questionnaire), LikertAcross (superseded by LikertHeader and LikertBody) andLikertDown. Deprecated templates continue to work, but are no longer recommended.
- Completely flexible **Questionnaire Branching** is now possible.
- The authenticated user ID can now be used in **question piping** by using the [/authenticateduserid] tag. So can the place and poll name,with [/spotname].[/pollname], and [/pollid].
- An improved **Design Questions** page includes multiple question copying, moving and removing in one operation. Custom Actions and branching information are now highlighted in red for visibility. Automatically generated question numbers are used throughout the application, making it easier to refer to a question in large surveys, while specific questions can be marked as unnumbered. A **Questionnaire Index** provides a bird-eye's view of questions, answers, branching patterns and custom actions which is extremely helpful when working on large surveys.
- New or improved **Style Tags for poll pages**: thispagenumber, numberpages, percentcomplete. These are used to construct survey progress bars in the Questionnaire style. [/ifremaining] can now test for -1 to see if a poll is closed and -2 to see if it is open without a closing time.
- New **Style Tags for response pages**: ifpollcount now includes a range of values. The {/ syntax is **explained**.
- New **Style Tags for quizzes**. The revised QuizResults style includes new tags useful in assessments and quizzes: ifdefault, endifdefault.
- Question Styles now have a methodology for validating entered input with JavaScript **Data Validation**. Three new question styles included are: CheckNumericText, an edit box that only allows numbers to be entered; CheckAlphaNumericText, another edit box that checks for USASCII letters and numbers, and CheckValidEmail, a text box that tests for valid E-mail address syntax. From these, you can easily construct your own and add them to the Styles and share them in a new public Styles Library in the **www.cogix.com** support area.
- The Administrator can now copy a poll or survey **between places**, which allows for sharing a successful or model survey between departments, for example.
- **New Custom Actions** are included for calculating weighted scores and for complex branching.
- New **Custom Action methods** allow a execution before a page is composed, as well as when a page is submitted, and redirecting the survey to a subsequent question. Additional convenience methods have been added.
- **Repetitive Log entries** are automatically suppressed if entered too often.
- New location for viewsflash.properties makes **upgrading** easier
- **Database** connectivity through standard Named Data Sources is now available. Additional instructions are provided for MSSQL. The **Normalized database** storage option is now enabled by default, without requiring the databaseextension servlet parameter. A new **dbtablenameprefix** servlet parameter can be used to provide an explicit schema name for ViewsFlash database tables.
- Univariate and Crosstabulations now count "Others" either individually or together. When used with a question of type text or hidden, turning Live Tally on is required to count them individually. See **Scoring**. A new setting on the **Live Tally** page enables fine-grained control over how many different values to tally in a text or hidden question, with a default of 250.
- Additional information is provided on **using a question** as the authenticated user ID and the required Security page settings and special tags that can be used to construct a complete URL for resuming a survey.
- Questions of type edit, hidden, text and multiline text now allow setting a default value, by entering it into the first answer's text.
- The "maximum field size" option in the Define a Question page was removed as it is no longer relevant.
- Archiving Keywords has been removed from the Design Questions page for lack of use.
- The following bugs have been fixed. A survey that didn't tally any questions was not being saved to the database. Retrieving data saved using the Normalized database took extremely long times with large numbers of respondents. The default answer to a question didn't work with drop-down menu style questions. Validation checks are no longer made after clicking Save or Previous in a multi-page survey. Question numbering sometimes failed with Question Styles.

Release 3.4 is a maintenance release.

- Some problems with Custom Actions in multi-page surveys are solved
- Includes new **Custom Action** classes testAction, a sample class very useful for learning how Custom Actions work
- Includes atgLoadFromProfile, used with ATG Dynamo for reading Properties from a Dynamo Profile.
- A new way to do **multi-page surveys in multiple languages** is introduced.

A new way to save cookies, IP addresses, HTTP headers, and servlet session values and attributes is available in **Save Options** under Browser Value (renamed from Cookie Value).

Release 3.4 is a feature release.

- A **JSP SDK** for integrating Cogix polls and surveys with JSP pages is now available.
- An **ATG Dynamo SDK** for integrating Cogix polls and surveys with ATG's Dynamo pages is now available.
- **Question Style Templates** allow giving a portion of a survey a distinctive look and behavior. Useful in constructing Likert scales and specialized browser behaviors.
- Multi-page surveys now show Page Breaks and Question Skips visually in the Design Poll page. "NoAnswer" now allows skipping to a particular question when someone leaves a question unanswered.
- Single question poll creation is easier thanks to a "Make Single Question Poll" button on the place page.
- In Analysis, hidden fields with enumerated lists of values can be defined and analyzed using Univariate and Cross tabulation tools. Averages and Standard Deviations are now calculated on Edit fields whose values are not enumerated in advance.
- Poll naming has been enhanced so that, for example, a poll can be named "Monday" in two different places.
- Default Style Templates now have appearance settings in the place settings page, rather than on the Design Poll and Design Response pages.
- **Web Application** packaging makes it easier to install ViewsFlash in standard application and servlet servers.
- **Question piping** allows using values entered on one survey page on subsequent pages and redirects. This, combined with the Numeric Score extensible class, allow construction of elaborate Quizzes.
- **Webcast Surveys** allow for a survey to be conducted during a live webcast. In addition, the survey questions can also be presented during an archived replay of the webcast at the same time as the original presentation. See also the **comparison** with Webcast Polling.
- **New Extensible classes** include: NumericScore, IssueHttpCommand, WebcastEvent, atgOraclePooledDataSource.
- **Enhanced archive** displays. The [/pollresults] tag in an archived poll display will shows results for all polls in an archive when the query string includes &showallresults=1.
- **Additional Web Services API** parameters. 'stylesfx' allows retrieving voting forms and results using different style templates; useful for presenting a survey or poll in multiple formats (html/wml/xml) simultaneously. 'pollidsfx' allows retrieving voting forms and results using alternate wordings, eg., in multiple languages, but referring to the vote counts on the underlying poll.
- **New Servlet Parameters**: webappname, appwebroot, webcasteventextension.

Release 3.3 is a feature release.

- **Cross tabulations** and **Univariate Analysis** capabilities have been added. Researchers can, using these pages, easily and quickly slice and data data using standard single variable measurements and 2 and 3-way cross tabulations with Chi Square and t-test scores.
- **Tables created** by ViewsFlash can now contain a sequence number.
- A new option in the **place Settings** page lets you receive an e-mail when polls and surveys are opened and closed.
- Additional **database error recovery settings** allow for fine tuning. Recommended when creating your own database tables rather than letting ViewsFlash create them for you.

Release 3.2 is a feature release.

- **Internationalization** allows designing and conducting surveys and polls in any language in the world, including European and Asian, using any character set encoding supported by Java, including Unicode.
- MSSQL Support is also included in this release.

Release 3.1.1 is a feature release.

- **Extensible Database Storage** adds the ability to store collected data in a database in custom formats. The extensible class provided uses a single table to store all responses to all surveys. Some database administrators prefer this approach to storage. Source code is provided.
- In multi-page surveys, redirection after certain failure conditions was not working properly. These bugs have been fixed.

Release 3.0 is a major architectural release.

Peer Architecture. When used with a database, multiple ViewsFlash instances on multiple machines can store all data in the database, including configuration, poll and survey data, in an industry-standard peer architecture. Should one server fail, the remaining ones take its place seamlessly. Migration utilities for prerelease 3.0 data are provided. JDBC 2.0 connection pooling and caching is supported.

- **Additional API commands**. cmd=geteither, allows retrieving the voting form if a visitor has not voted, or the current results if they have voted already. Cmd=ifvoted tests this condition and retrieves 1 or 0.
- **Integrating with Application Servers**. This new section explains how to communicate, using the Web Services API, between a hosting Application Server such as Dynamo, and the ViewsFlash servlet, especially when they both reside on the same VM.
- **ATG Dynamo integration**, including using the "repository id" from the Profile for authentication, saving data right to the Profile, embedding polls and personalized quizzes in dynamic pages, and presenting randomized, interstitial surveys, personalized based on Slots programmed by Scenarios.
- **Additional Java extensibility** is available for integrating with application servers. New features include extensible authentication code, passing parameters using servlet request attributes, writing response data to the servlet context. Cmd=geteither, nooutputclose, nooutputheaders
- **Record who voted on database**, a new option on the Security page, allows for scalable authenticated voting when used with authenticated surveys and voting. This setting must be used for reliable results in distributed configurations.
- **Polling during Webcasts**. Live focus groups and audience surveys are now possible using browser JavaScript and new publishing commands.
- Miscellaneous.
  - A txtpath= servlet parameter allows portending a different directory for .txt files that save survey or poll response data.
  - A hidden field can have a default value; just enter one answer. The poll style template you use must have an [/allanswers] block in its [/hidden] section; see the revised Standard_.html template.
  - The default method for reading servlet parameters has changed to the more standardized getParameterNames(); the legacy method of reading the request input stream directly is still available by using servlet parameter servletrequestinputstream=1.

Release 3.0.1

- is a maintenance release. It fixes import problems that affected clients migrating from earlier versions. On distributed database systems, it offers better real-time results display and fixes a problem that rendered random sampling inoperative.

Previous release notes.

Next: **Introduction**

Custom Actions: WeightedScore

This custom action allows calculating a score based on how a visitor answers a series of questions with numeric answers. Each question can be assigned a specific weight. The result can be the sum or an average.

WeightedScore target.decimalplaces,op,question*weight,question*weight,...
The value entered in each question listed is multiplied by its optional weight. The values are added if "op" is "sum", or averaged if "op" is "average". The result s rounded to the indicated number of decimal places or to the nearest integer if decimalplaces is not specified. The result is stored in question "target", which is usually defined as a hidden question. The target question can be saved, used as a **piped question**, and used in Univariate and Crosstabulation analyses.

Example:

	No	Some	Lots
Do you smoke?	○	○	○
Do you drink?	○	○	○
Are you overweight?	○	○	○
Submit			

Hidden Question: combined_risk_score

Custom Action:
WeightedScore combined_risk_score,sum,smoke*.4,drink*.3,overweight*.3

Three questions of type radio are named smoke, drink, and overweight. The three questions have values 1, 2, and 4 corresponding to No, Some, and Lots. Note that the value 4 (rather than 3) indicates the aggravated importance of an excessive risk factor. When calculating a combined health risk score, "smoke" is given a higher weight to account for its specially pernicious toxicity. Note how the weights are decimal numbers, .4 and .3. Because no decimal points are indicated, the combined_risk_score hidden question will store the score rounded to the nearest integer.

Custom Actioon:
WeightedScore combined_risk_score.2,average,smoke*4,drink*3,overweight*2

In this example, each of the questions's values will be multiplied by the appropriate factor, and the result will be averaged. If a question was not answered, it will not be used when calculating the average.

For an additional example, see **Branching**.

Next: Event Notification

Installing the sample Setup.exe

The setup.exe file includes ViewsFlash and all necessary componets to deploy it on a personal computer for evaluation. It includes an application server and a database.

Download the setup.exe file and open it. Follow the instructions in the installer.

When installation is complete, there should be an item in the Start menu under Cogix / ViewsFlash.
Run that program, and wait for the browser to open up. It will point to
http://localhost:88

From there, follow the instructions in the browser.

ViewsFlash 5 sample viewsflash.properties file
These parameters can also be entered as init-properties in WEB-INF/web.xml in clustered deployments

A # at the beginning of a line turns it into a comment which is otherwise ignored.
To enable a parameter, remove the # at the beginning of the line.

'data' must point to a directory where the ViewsFlash application has full access rights
In Unix:
data=/etc/cogix
In Windows. Note capital drive letter. Folder names are case sensitive:
data=E:/etc/cogix

License key. Needed to remove the "Evaluation Copy" message. Contact sales@cogix.com for a key. Do not use this key!
#license=who=Demo_License;type=unlimited;exp=07/31/05;bdb=y;bua=y;brs=y;bwp=y;licensekey=63-85-90-91-0E-07-2E-AC-E7-9A-F1-BB-17-C5-AB-87-

##

If using a database, use one of these:
#database=Oracle
#database=DB2
#database=MySQL
#database=MSSQL
#database=Cloudscape
#database=Derby

If using a database, enter the JNDI name of a data source defined in the application server,
as specified below. If your data source's JNDI name is "MyDataSource", then use:

In Tomcat:
datasource=java:comp/env/MyDataSource

In JBoss:
datasource=java:/MyDataSource

In WebSphere 5:
datasource=MyDataSource

In WebSphere 6:
datasource=MyDataSource

In Weblogic:
datasource=MyDataSource

If your data source's JNDI name is "jdbc/MyDataource", then use jdbc/MyDataSource instead of MyDataSource.

If the datasource parameter is wrong, you'll get a NameNotFoundException. In that case try setting
dataource to each of these: java:comp/env/MyDataSource, MyDataSource, java:/MyDataSource, jdbc/MyDataSource

Use this to connect directly to Oracle using OraclePooledDataSource,
instead of using a JNDI Data Source (Oracle only).
If you use this, make sure that database= above and datasource above remain commented out
#dbservername=databaserver.yourcompany.com
#dbusername=yourdatabaseuserorschemaname
#dbpassword=yourdatabasepassword
#database=OraclePooledDataSource
#dbdatabasename=yourdatabase
#dbdrivertype=thin
#dbprotocol=tcp
#dbport=1521

If not using a database, leave the database parameters above as comments

##

If using User Security, review ViewsFlash/WEB-INF/web.xml, and use these parameters:
appsecurity=user
administrators=UserName1,UserName2,UserName3,@Role1,@Role2
If having difficulty logging in to the application, remove these parameters and add them later.

If not using User Security, enable password security by entering a ViewsFlash Administrator password:
adminpw=xxxxxxx

Modifying the ViewsFlash.war file

How to repack the ViewsFlash.war file

Follow these instructions one how to unpack the ViewsFlash.war file, modify and/or add files, and repack the war file for deployment.

Linux systems (using a command shell).
1. Change to a temporary directory, "tmp" in this example, with:

```
cd /tmp
```

2. Create a ViewsFlash subdirectory with:

```
mkdir ViewsFlash
```

3. copy the original ViewsFlash.war file into /tmp

4. rename ViewsFlash.war to ORIGViewsFlash.war with:

```
mv ViewsFlash.war ORIGViewsFlash.war
```

5. extract ViewsFlash.war with:

```
cd ViewsFlash  $JAVA_HOME\bin\jar -xvf ../ORIGViewsFlash.war
```

6. edit or replace /tmp/ViewsFlash/WEB-INF/web.xml as needed and save.

7. Repack the war with:

```
cd /tmp  $JAVA_HOME/bin/jar -cvf ViewsFlash.war -C ViewsFlash .
```

The repacked war will be /tmp/ViewsFlash.war

Windows systems (using a command window).
1. Change to a temporary directory, "tmp" in this example, with:

```
cd \tmp
```

2. Create a ViewsFlash subdirectory with:

```
mkdir ViewsFlash
```

3. Use Explorer to copy the original ViewsFlash.war file into C:\tmp

4. rename ViewsFlash.war to ORIGViewsFlash.war with:

```
ren ViewsFlash.war ORIGViewsFlash.war
```

5. extract ViewsFlash.war with:

```
cd ViewsFlash  %JAVA_HOME%\bin\jar -xvf C:\tmp\ORIGViewsFlash.war
```

6. use notepad to modify C:\tmp\ViewsFlash\WEB-INF\web.xml as needed and save.

7. Repack the war with:

```
cd \tmp  %JAVA_HOME%\bin\jar -cvf ViewsFlash.war -C ViewsFlash .
```

The repacked war will be C:\tmp\ViewsFlash.war

Next: Setup

```
<!--  When using appsecurity=user, enable this to protect product documentation
   <security-constraint>
     <web-resource-collection>
       <web-resource-name>ViewsFlash documentation</web-resource-name>
       <url-pattern>/viewsflash/docs/*</url-pattern>
     </web-resource-collection>
     <auth-constraint>
       <role-name>ViewsFlashUser</role-name>
     </auth-constraint>
     <user-data-constraint>
       <transport-guarantee>NONE</transport-guarantee>
     </user-data-constraint>
   </security-constraint>
-->
<!--  When using appsecurity=user, enable this to protect the vfupload servlet
   <security-constraint>
     <web-resource-collection>
       <web-resource-name>ViewsFlash upload servlet</web-resource-name>
       <url-pattern>>/servlet/vfupload</url-pattern>
     </web-resource-collection>
     <auth-constraint>
       <role-name>ViewsFlashUser</role-name>
     </auth-constraint>
     <user-data-constraint>
       <transport-guarantee>NONE</transport-guarantee>
     </user-data-constraint>
   </security-constraint>
-->
```

Upgrading ViewsFlash from versions older than 3.5

Upgrading from ViewsFlash versions earlier than 3.5

The only change in configuration introduced with 3.5 is the default location of the viewsflash.properties file. Before 3.5, this file (by default) was found, and had to be changed, in the WEB-INF directory. The problem with this is that an upgrade often would overwrite this file, requiring its backup and restore during the upgrade process, which is an error-prone procedure. Beginning with 3.5, a directory /etc/cogix holds the viewsflash.properties by default. This makes upgrading ViewsFlash 3.5 with later versions much easier -- just redeploy the new ViewsFlash.war.

If you have been running ViewsFlash as a servlet rather than as a web application, we recommend installing the upgrade as a web application for current and future convenience.

The document includes instructions for following upgrade scenarios:

ViewsFlash version 3.x to version 3.5 as a web application
ViewsFlash version 2.x to 3.5 as a servlet (not recommended)
ViewsFlash version 2.x to version 3.5 as a web application (recommended)
ViewsFlash version 3.x servlet installation to version 3.5 servlet installation (not recommended)
ViewsFlash version 3.x servlet installation to version 3.5 as a web application (recommended)

ViewsFlash version 3.x web application to version 3.5 web application

1. Copy viewsflash.properties to /etc/cogix. If you were using the servlet parameter "propertiesfile", remove it.

If you prefer to keep viewsflash.properties in some other location, then make sure that the propertiesfile parameter is preserved after upgrading. For example, if this parameter is specified in WEB-INF/web.xml, it will be lost during the upgrade, so you must save and restore web.xml. This is why the easiest way to upgrade is to let ViewsFlash find its viewsflash.properties file in /etc/cogix.

2. Make backup copies of the files in viewsflash/styles2, and the WEB-INF directory.

3. Re-deploy the ViewsFlash web application using the standard procedure for your application server.

4. Examine the new /viewsflash/styles2 directory and the files you backed up in step 2. If you have made changes to any of the original files in that directory, and the upgraded version has changed, it's best to rename your files something else and to change references to them in the places or other places where they are used.

5. Read the updated documentation and if any new ViewsFlash **servlet parameters** need to be added, do so now.

6. Restart the application server. Test your installation as described in the **installation** section and **troubleshoot** if necessary.

ViewsFlash version 2.x to 3.5 as a servlet (not recommended)

Please make note of the following points before upgrading:

- If you had configured ViewsFlash 2.8 or earlier to store data in a database, go to the Database Setup page and print it. This database configuration information on this page is replaced by servlet parameters.
- The ?Diagnose command has changed to ?Diagnose=1.
- Before proceeding please read **Upgrading Style Templates.**
- Contact Cogix for a new license key - version 2.x keys do not work with version 3.x

1. Stop the application server where ViewsFlash resides.

2. Make a backup copy of all data in the ViewsFlash data directory (specified in the servlet configuration parameters). Backup the /viewsflash directory and the viewsflash.jar file (or the /com/cogix/vwf servlet directory for older releases) so you can roll back the upgrade to the previous release. Backup the web server directories where ViewsFlash writes data.

3. Unpack the files in the upgrade, as described in the **Installation** procedure. The files in /ViewsFlash/WEB-INF/classes/com/cogix/vwfhtml replace and add to the corresponding files in the /existing/webroot/viewsflash directory; same with the /ViewsFlash/viewsflash/docs and /ViewsFlash/viewsflash/styles2 files.

4. The /ViewsFlash/WEB-INF/lib/viewsflash.jar file must be added to the application runner's VM classpath, and so must

/ViewsFlash/WEB-INF/lib - Use the complete name for these directories, such as /usr/Cogix/ViewsFlash... etc, or E:\Cogix\ViewsFlash\...

5. If you had configured ViewsFlash to store data in a database before, please read the **Database Use** section and add the new servlet parameters described there to your viewsflash servlet parameters. The previous method for describing database attributes via an HTML page is no longer available.

6. Replace the existing 2.9 license key in your viewsflash servlet parameters with the new version 3 license key.

7. Turn the application server back on, and continue the Installation procedure. If the Diagnose=1 command does not produce successful results, it is best to stop the web server or servlet runner which runs the ViewsFlash VM, make changes, and restart the web server.

All polls that were running before the upgrade will continue running as before. If there's a problem, you can use the backup files from step 2 to restore things to their prior state.

ViewsFlash version 2.x to version 3.5 as a web application (recommended)

Please make note of the following points before upgrading:

- If you had configured ViewsFlash 2.8 or earlier to store data in a database, go to the Database Setup page and print it. This database configuration information on this page is replaced by servlet parameters.

- The ?Diagnose command has changed to ?Diagnose=1.
- Before proceeding please read **Upgrading Style Templates.**
- Contact Cogix for a new license key - version 2.x keys do not work with version 3.
- When upgrading to a web application installation from a servlet installation the Viewsflash URLs will change to include the name of the web application. For example: http://yourdomain.com/servlet/viewsflash becomes http://yourdomain.com/ViewsFlash/servlet/viewsflash - after the upgrade you'll need to republish all existing polls and surveys.

1. Make a backup copy of all data in the ViewsFlash data directory (specified in the servlet configuration parameters). Backup the /viewsflash directory that is in the web server's document root directory. Also backup the viewsflash.jar file so you can roll back the upgrade to the previous release. Backup the web server directories where ViewsFlash writes data. Also, record all of the servlet parameters used in the 2.x installation as these will need to go into the viewsflash.properties file.

2. Remove viewsflash.jar from any classpaths.

3. Install ViewsFlash as a Web Application as described in the **Installation** section of this documentation.

4. Add the existing servlet parameters to the /etc/cogix/viewsflash.properties file. Review the **servlet parameters** section and add any new parameters that may be required at this time. Note also that a new version 3 license key will be required as the existing version 2 key will not work.

5. Copy any templates that you have created in /existing/webroot/viewsflash/styles2 to /ViewsFlash/viewsflash/styles2 being careful to not overwrite new templates with old ones with the same name. Please refer to **Upgrading Style Templates**.

6. Restart the application server. Test your installation as described in the **installation** section and **troubleshoot** if necessary.

7. After a successful ?Diagnose=1, click on the Administration and Setup link and change items #2 and #3 as needed to reflect the new /ViewsFlash URI.

ViewsFlash version 3.x installation to version 3.5 as a servlet

1. Make a backup copy of all data in the ViewsFlash data directory. Backup the /viewsflash directory and the viewsflash.jar file so you can roll back the upgrade to the previous release. Backup the web server directories where ViewsFlash writes data.

2. Unpack the Viewsflash war file distribution in a convenient location (like /tmp) using:
jar -xvf ViewsFlash.war

3. Copy files and directories as follows:

Note: on Windows systems, use back slashes (\) instead of forward slashes (/).

/tmp/ViewsFlash/WEB-INF/lib/viewsflash.jar to /existing/Cogix/ViewsFlash/lib/viewsflash.jar

/tmp/ViewsFlash/WEB-INF/lib/* to /existing/Cogix/ViewsFlash/lib/*
/tmp/ViewsFlash/WEB-INF/classes/com/cogix/vwfhtml/* to /existing/webroot/viewsflash (replace existing files of the same name)
/tmp/ViewsFlash/viewsflash/styles2/* to /existing/webroot/viewsflash/styles2 (replace existing files of the same name)
/tmp/ViewsFlash/viewsflash/docs/* to /existing/webroot/viewsflash/docs (replace existing files of the same name)

where /tmp is the location in which the ViewsFlash.war file is unpacked in step 2 and/existing/Cogix/ is where you unpacked your current Viewsflash 3.x distribution and /existing/webroot is the existing document root directory of your web server. Take into account that your current 3.x installation may exist in a different directory structure than shown here.

4. Restart the application server and test with /servlet/viewsflash?Diagnose=1 then **troubleshoot** any reported errors.

ViewsFlash version 3.x servlet installation to version 3.5 as a web application

When upgrading to a web application installation from a servlet installation the Viewsflash URLs will change to include the name of the web application. For example: http://yourdomain.com/ViewsFlash/servlet/viewsflash becomes http://yourdomain.com/ViewsFlash/servlet/viewsflash

1. Make a backup copy of all data in the ViewsFlash data directory. Backup the /viewsflash directory and the viewsflash.jar file so you can roll back the upgrade to the previous release. Backup the web server directories where ViewsFlash writes data.

2. Remove viewsflash.jar from any classpaths.

3. Install ViewsFlash as a Web Application as described in the **Installation** section of this documentation.

4. Copy your existing viewsflash.properties file to /etc/cogix/viewsflash.properties

5. Copy any templates that you have created in /existing/webroot/viewsflash/styles2 to /ViewsFlash/viewsflash/styles2 being careful to not overwrite new reference templates with old ones with the same name.

Any existing templates will need to incorporate the new tag [/webappname] so that wherever there were references to "/viewsflash" now there are references to "[/webappname]/viewsflash".

6. Restart the application server. Test your installation as described in the **installation** section and **troubleshoot** if necessary.

7. After a successful ?Diagnose=1, click on the Administration and Setup link and change items #2 and #3 as needed to reflect the /ViewsFlash URI as it is installed as a web application.

Next: Upgrading Style Templates

Webcasting with polls

By combining a set of frames, JavaScript, new API commands, and new poll scheduling options, it is now possible to conduct live polls or surveys while an online audio or video presentation is presented at the same time.

1. Create a place. In option 6, choose "Use the place name".

2. Create different polls to be presented at different times during the live presentation. A poll can be one or several questions that will be presented at once. Do not use multi-page polls. Publish the polls but do not specify a start or end date. On the Security page, turn on duplicate vote checking.

3. Become familiar with the files in /ViewsFlash/webcast. Copy all these files to a working directory and customize them to suit your needs. PollPresentation.html is frame set configured for presentation of polls during a webcast. The left frame uses other.html, which you should customize to show a video stream, for example. The right frame uses PollFrame.html, which is a frameset for presenting the polls in synchronization with the webcast moderator.

4. Open a browser to the place page and click on Webcast. On this page you have the following options:

Start	Click on this shortly before the start of the webcast. This clears the poll display and marks the beginning of the webcast.
Blank	Click on this anytime during the presentation to remove the poll display.
Show Show » Show	Click on one of these links to advance the presentation to that question. The question currently shown is marked »
End	Click on this to clear the display and mark the end of the webcast.

Webcast Participants should open a browser window go to the poll presentation you just created. Its URL is:

/ViewsFlash/webcast/PollPresentation.html?eventid=PPPP

where PPPP is the name of the place created in step 1 above.

5. For best results, create an introductory and and ending poll that say "Welcome" and "Thank you" (use the 'Just HTML' question type).

Next: **Webcast Surveys**

Comparative Ratings

In addition to the ability to put up product rating forms and displays on a product by product basis, ViewsFlash can also create comparison tables automatically and learn product information dynamically.

As with basic ratings, the product ID will be used as the poll ID. In addition, you will need a way to group products to be compared. For example, all 2000 model European convertibles; these will be grouped automatically into a" place". Then you can make comparisons between all the products in a group in a single call to ViewsFlash.

1. Follow all the steps in **Ratings**, construct a simple system without comparative rankings, and make sure it works.

2. Add to your fetchpoll and fetchresults commands the parameter "&spot=GGGG", where GGGG is the ID of the product group they belong to. ViewsFlash will automatically create a place for them, copying the settings from the place where the "likeid" poll belongs.

3. To fetch the comparative results table, create a template as defined below, and compose a page that includes the command:
/servlet/viewsflash?cmd=fetchtable&... (other parameters)

The additional parameters are:

spot	name of the product category	required
templatefile	name of response page template, relative to the viewsflash web server root. This is not the name of a style template!	required
rank	name of field to rank by	optional
max	maximum number of items to display	optional; default is 10
method	what to rank by; either *average* or *count*	optional; default is average
minvote	minimum number of votes required to display	optional; default is 1
order	sorting order; can be *top* or *bottom*	optional; default is top (to bottom)

Constructing a ratings table template.

Create a response page template and surround it with [/rankedpolls] and [/endrankedpolls] tags. Note that this is not a Response Style template; you will typically only use tags the "avg" and "count" tags. Store this template in the viewsflash web server root directory structure.

Additional parameters

The example on the Cogix site includes elements such as Price and the name product name. Where do they come from? ViewsFlash can learn these as it goes along. Here's what you do:

1. Use the fields "param0" through "param9" to hold properties of each item, such as its name and its price.

2. In both the cmd=fetchpoll and cmd=fetchresults calls, include the appropriate characteristics. For example:
cmd=getpoll&pollid=Racing!Carrera&likepoll=Convertibles&param0=Porsche 911E Carrera Convertible&param1=90000

As ViewsFlash serves voting forms and ratings results tables, it will learn the names and prices of the rated items automatically.

3. Use the param0 through param9 field names in the ratings table templates.

Here's a simplified version of the template used in the Ratings example, using param0 for the product name and param1 for its price. Overall, Ease, Quality, Ease, Durability and Value are the names of the Questions that every ratings form includes. The "avg,1" notation means include one decimal place in the average score displayed. Because ratings are between 0 and 5 stars, files named 0star.gif through 5star.gif are served.

[/rankedpolls]
<tr><td>[/param0] </td>
<td align="right">$[/param1]</td>
<td align="center">[/Overall,count]</td>
<td ><img src="/viewsflash/[/Ease,avg,0]star.gif"></td>
<td > [/Overall,avg,1]</td>
<td > [/Quality,avg,1] </td>
<td > [/Ease,avg,1] </td>
<td > [/Durability,avg,1] </td>
<td > [/Value,avg,1] </td>
</tr>
[/endrankedpolls]

Cogix Client services is available to assist with all phases of creating a comprehensive Ratings system.

Next: Quizzes

Automated Poll Rotation control flow

Back to the **Web Services API**

This chart illustrates the events, from top to bottom. For simplification, caching is implicit.

Visitor's browser	Web Server	ViewsFlash Server
anywhere	main.big.com	vf.big.com
Goes to dynamic page http://main.big.com/sports.dyn, handled by the Web Server		
	Constructs dynamic sports page which includes the embedded poll for "sports" place. Because it is first time on this page, there is no VFPOLLS cookie. Issue http call: http://vf.big.com/ViewsFlash/servlet/viewsflash?cmd=getpoll&spotname=sports	
		Returns HTML fragment for the voting form to main.big.com
	Inserts HTML fragment with voting form into page and returns full page to visitor	
Visitor sees page with embedded poll. Fills out poll and Submits voting form to vf.big.com		
		Tallies vote. Sends VFPOLLS cookie with domain "big.com" back to visitor, and redirects visitor browser to redisplay the original page http://main.big.com/sports.dyn
Visitor's browser quietly goes to http://main.big.com/sports.dyn again.		
	Constructs dynamic page again. This time, there is a VFPOLLS cookie. Its contents indicate that the current poll has not expired. Therefore, issues http call: http://vf.big.com/ViewsFlash/servlet/viewsflash?cmd=getresults&spotname=sports	
		Returns HTML fragment with poll results display to main.big.com
	Inserts HTML fragment with poll results display into page and returns full page to visitor.	
Visitor sees the original sports page, but now sees the results instead of a voting form.		
... (a short time passes)		
Later, visitor returns to this page.		
	Because the VFPOLLS cookie indicates poll has not expired yet, issue getresults http call to vf.big.com	
..(even more time passes; poll expires in the meantime)...		
Visitor return to page		
	VFPOLL cookie indicates poll has expired. Issue	

	getpoll http call to vf.big.com.	

Next: Extending ViewsFlash

```
package com.cogix.vwf;

/*
	Sample Custom Action class to illustrate best practices
	Revised 12/2/02
	Requires ViewsFlash 3.5 or later

	See Extensibility in the ViewsFlash documentation:
	http://www.cogix.com/viewsflash/docs?Extensible.html

	Construct a poll with 5 text fields.
	Name them destination, source1, source2, returnvalue, action.
	Add a Custom Action question and use the following
		to activate this class:

		testAction destination,source1,source2,returnvalue,action

		These five parameters are the names of the 5 questions
		defined in the test survey.

	Experiment with this class in both single page and multi-page surveys.

	This example class will compare the contents of source1 with source2.
	and use the comparison to set validity.

	It will use the returnvalue field to return true or false to the application
	Enter t or f.

	It will also call setAction() with the value of the action field.
	Enter a number from these values:
		ActionUseInvalidSettingsonSecurityPage = 0;
		ActionShowPage  = 1;
		ActionRedirect  = 2;
		ActionAcceptWhatThereIs = 3;
		ActionRefill = 4;
		AcceptAndRedirect = 5;

	It also uses setParameterValue to set destination to the
	concatentation of source1 and source2.
*/


class testAction  extends Validator {


	//	If necessary, add a throws clause.  If an exception is thrown,
	//	a message will be written to the poll log.
	//	It's best to not throw exceptions in a production environment.
	boolean onPageReceived (RequestParams reqx, String questionname, String spec) {

		//	Enable the following to write to the Poll log, for debugging:
		//	Dirs.getErrorLog().WriteLog ( pollid , "606I","At testAction # 1");
		//	Dirs.getErrorLog().WriteLog ( pollid , "606E","At testAction # 1");
```

```
        //      Messages ending in E are emailed to administrator immediately

                //      Perform integrity check on all parameters first.
        //      Return false to indicate that the check is indeterminate,
        //      i.e., ignored.
        String [] s = Misc.cogixParse (spec, true);
        if ( s == null || s.length < 4 ) {
                WriteLog ("301E","testAction didn't have enough parameters");
                return false;   //      do nothing
                }

        String destination = reqx.getParameter ( s [0] );
        String source1 = reqx.getParameter ( s [1] );
        String source2 = reqx.getParameter ( s [2] );
        boolean bSame = Misc.isSame ( source1, source2 );
        String retvalue = reqx.getParameter ( s [3] );
        String actionp = reqx.getParameter ( s [4] );
        boolean bReturn = ! ( ! Misc.isEmpty ( retvalue ) && retvalue.charAt (0) == 'f' );
        String setto = source1 + source2;

        //      Use this to set a value. Value can be saved, piped, and tabulated.
        setParameterValue (  destination, setto );

        //      Use this to indicate that the action has detected no problems (true)
        //      or not (false).
        //      If not called, the default is false !
        setValid ( bSame );

        //      Examples of other methods for certain cases.
        setMessage ("Custom Action 'testAction' finds data inValid.");
        setAction ( Integer.parseInt ( actionp ) ) ;    //      can throw NumberFormatException
        setRedir ( "http://www.yahoo.com");     //      easily visible
        switch ( action ) {
                case Validator.ActionRefill:
                        setErrorTemplate (null);        //      null means use poll's Style Template
                                                        //      can be used to use special Poll Style Template
                        //      compose list of questions that need refilling,
                        //      surrounded by blanks on both sides
                        String missentries = " " + source1 + " " + source2 + " ";
                        setMissingEntries (missentries);
                        setMessage ("Correct marked fields ");
                        break;
                case Validator.ActionShowPage:
                        setErrorTemplate ("viewsflash/vfincomplete.html");
                        break;
                default:;
                }
        //      A return of false means to ignore whatever this Action says to do.
        //      think of it as "Cancel".
        return bReturn;
        }
}
```

```
package com.cogix.vwf;

/*
	BeforePageComposedGoTo
	Illustrates a class that prevents an entire page from displaying.

	Usage:
	BeforePageComposedGoTo destinationquestion,testquestion,value,value,value
	(no value means when blank)
	Use ! in front of testquestion to mean NOT.

	Skips showign this page when the condition is true

	Revised 10/8/02
	Requires ViewsFlash 3.5 or later

	See Extensibility in the ViewsFlash documentation:
	http://www.cogix.com/viewsflash/docs?Extensible.html

*/


class BeforePageComposedGoTo extends Validator {

	//	This function is called when the survey arrives at the page containing
	//	this Custom Action.  By default, it returns false. When it returns true,
	//	the page that contains the question set in the skipToQuestion method is displayed.
	//	Note that the destination page can contain additional Custom Actions,
	//	which in turn can themselves cause further redirection.


	boolean onPageComposition (RequestParams reqx, String questionname, String spec) {
		//	Perform integrity check on all parameters first.
		//	Return false to indicate that the condition is not met.
		boolean bSkip = false;
		String [] s = Misc.cogixParse (spec, true);
		if ( s == null || s.length < 1 )
			return false;	//	do nothing

		String destinationquestion = s [0] ;
		if ( Misc.isEmpty (destinationquestion) )
			return false;

		if ( s.length < 2 ) {	//	unconditional skip
			skipToQuestion ( destinationquestion );
			return true;
			}

		String testquestion = s [1] ;
		boolean bIsNot =	testquestion.charAt (0) == '!';
		if (bIsNot) {
			if ( testquestion.length() < 2 )
				return false;
```

```
            testquestion = testquestion.substring (1);
            }
        int nparams = s.length - 2;

        //      Get the values for that question.
        //      A multiple valued answer returns xx,xx,xx, so use
        //              boolean valuesContain (String values, String avalue)

        String values = getParamValue ( reqx, testquestion );
        boolean bNoValues = Misc.isEmpty (values);
        //      First try it with a single valued question.
        if ( nparams == 0 ) {   //      Test for empty or not
            bSkip = bIsNot != bNoValues;
            }
        else {
            if ( bIsNot ) { //      See if responses don't match any of the values
                if ( bNoValues ) {
                    bSkip = true;
                    }
                else {
                    boolean bMatch = false;
                    for ( int k = 0; ! bMatch && k < nparams; k++ ) {
                        if ( valuesContain ( values, s[k+2] ) )
                            bMatch = true;
                        }
                    bSkip = ! bMatch;
                    }
                }
            else {  //      See if responses match any of the values
                if ( ! bNoValues ) {
                    for ( int k = 0; ! bSkip && k < nparams; k++ ) {
                        //      if ( values.equals ( s[k] ) )
                        if ( valuesContain ( values, s[k+2] ) )
                            bSkip = true;
                        }
                    }
                }
            }
        if ( bSkip )
            skipToQuestion (destinationquestion);
        return bSkip;
        }
}
```

```
package com.cogix.vwf;

/*
	WhenPageReceivedGoTo
	Illustrates a class that directs the next page to show.

	Usage:
	WhenPageReceivedGoTo destinationquestion,testquestion,value,value,value
	(no value means when blank)
	Use ! in front of testquestion to mean NOT.

	Goes to the page that contains destinationquestion when the condition is true

	If no testquestion and values provided, acts as an unconditional Go To

	Revised 10/8/02
	Requires ViewsFlash 3.5 or later

	Uses skipToQuestion (questonname) and then returns true immediately.
*/


class WhenPageReceivedGoTo extends Validator {

	//	This function is called when the survey arrives at the page containing
	//	this Custom Action.  By default, it returns false. When it returns true,
	//	the onPageReceived method is called before the page is displayed,
	//	See testShouldPageBeShown class for example.

	boolean onPageReceived (RequestParams reqx, String questionname, String spec) {
		//	Perform integrity check on all parameters first.
		//	Return false to indicate that the condition is not met.
		boolean bSkip = false;
		String [] s = Misc.cogixParse (spec, true);
		if ( s == null || s.length < 1 )
			return false;	//	do nothing
		String destinationquestion = s [0] ;
		if ( Misc.isEmpty (destinationquestion) )
			return false;
		String testquestion = s.length < 2 ? "" : s [1] ;
		boolean bIsNot = testquestion.length() > 0 &&	testquestion.charAt (0) == '!';
		if (bIsNot) {
			if ( testquestion.length() < 2 )
				return false;
			testquestion = testquestion.substring (1);
			}
		if ( s.length == 1 ) {	//	skip to destination unconditionally
			bSkip = true;
			}
		else {
			int nparams = s.length - 2;
			//	Get the values for that question.
			//	A multiple valued answer returns xx,xx,xx, so use
```

```
        //          boolean valuesContain (String values, String avalue)
        String values = getParamValue ( reqx, testquestion );
        boolean bNoValues = Misc.isEmpty (values);
        //      First try it with a single valued question.
        if ( nparams == 0 ) {   //      Test for empty or not
            bSkip = bIsNot != bNoValues;
            }
        else {
            if ( bIsNot ) { //      See if responses don't match any of the values
                if ( bNoValues ) {
                    bSkip = true;
                    }
                else {
                    boolean bMatch = false;
                    for ( int k = 0; ! bMatch && k < nparams; k++ ) {
                        if ( valuesContain ( values, s[k+2] ) )
                            bMatch = true;
                        }
                    bSkip = ! bMatch;
                    }
                }
            else {  //      See if responses match any of the values
                if ( ! bNoValues ) {
                    for ( int k = 0; ! bSkip && k < nparams; k++ ) {
                        if ( valuesContain ( values, s[k+2] ) )
                            bSkip = true;
                        }
                    }
                }
            }
        if ( bSkip )
            skipToQuestion (destinationquestion);
        return bSkip;
        }
}
```

```
package com.cogix.vwf;
import java.text.*;

class WeightedScore extends Validator {

    //    WeightedScore targetquestion[.decimalplaces],{sum|average},questionlist...
    //    Calculates the sum or average of the questions listed,
    //        showing 'decimalplaces' after the result's decimal point
    //        If .decimalplaces omitted, rounds to nearest integer and omits the decimal point.

    //    Each question on the list can indicate a weight to multiply the question by,
    //        using the syntax question*weight with or without a decimal point,
    //        e.g., question*2 or question *.4 or question*1.03 or question*.30

  //  Each question on the list can refer to the answer value, or to the Extra value
  //    associated with the answer. To refer to the answer value, just use the question name.
  //    To refer to its Nth extra value, follow the question name with [N], eg: income[0].

    //    Examples.    Assuming that questions smoke, drink, and exercise
    //                 have values between 1 and 5, then:

    //        WeightedScore targetquestion,sum,smoke*.40,drink*.40,exercise*.20
    //            gives scores between 1 and 5
    //        WeightedScore targetquestion.2,average,smoke*4,drink*4,exercise*2
    //            gives scores between 3.33 and 16.67

  boolean onPageReceived (RequestParams reqx, String questionname, String spec) {
    //
    String [] s = Misc.cogixParse (spec, true);
    if ( s == null || s.length < 3 || s[0].length () < 1 || s[1].length () < 1 || s[2].length () < 1 ) {
      //  WriteLog automatically includes the poll ID and writes to the Poll's log
      WriteLog ( "330E","WeightedScore arguments missing");
      return false;
    }
    String targetquestion = s[0];
    int decplaces = 0;
    int kdecperiod = targetquestion.indexOf ('.');
    if ( kdecperiod > 0 ) {
      String decimals = targetquestion.substring ( kdecperiod + 1 );
      targetquestion = targetquestion.substring ( 0, kdecperiod );
      try {
        decplaces = Integer.parseInt (decimals);
      }
      catch ( NumberFormatException nfe) {
        WriteLog ( "330E","WeightedScore decimal places not a number: " + s [0] ) ;
        return false;
      }
    }
    //    Turn on tallying for targetquestion
    String msg = setTallying ( targetquestion );
    if ( msg != null ) {
      WriteLog ( "330E","WeightedScore " + msg );
      return false;
```

```
}

boolean bSum = true;
boolean bAverage = false;
if ( s.length >= 2 ) {
   if ( "average".equals ( s[1] ) ) {
      bAverage = true; bSum = false;
   }
}
//     Go trough all questions and calculate score
double fscore = 0;
int nQuestions = 0;
for ( int j = 2 ; j < s.length ; j ++ ) {
   String qtoinclude = null;
   String weight = null;
   double fweight = 1.0;
   String elmt = s [j];       //     qname or qname*2 or qname*.6
   int times = elmt.indexOf ('*');
   if ( times == 0 ) { //     bad syntax: * at the beginning
      WriteLog ( "330E","WeightedScore question name missing: " + elmt ) ;
      return false;
   }
   else if ( times < 0 ) {     //     no weight
      qtoinclude = elmt;
   }
   else if ( times >= elmt.length()-1 ) {      //     bad syntax: * at the end
      WriteLog ( "330E","WeightedScore weight missing: " + elmt ) ;
      return false;
   }
   else {      //     questionname*weight
      qtoinclude = elmt.substring (0,times);
      weight = elmt.substring ( times + 1 ) ;
      try {
         fweight = Double.parseDouble ( weight );
      }
      catch ( NumberFormatException nfe ) {
         WriteLog ( "330E","WeightedScore weight not numeric: " + elmt ) ;
         return false;
      }
   }
   //  3.9 Add [N] syntax to use extra value N
   int nExtra = -1;
   String sExtra = null;
   int iLB = qtoinclude.indexOf ('[');
   if ( iLB >= 0 ) {
      if ( iLB == 0 ) {
         //  error $$$$
         WriteLog ( "330E","$$$: " + "[" ) ;
         return false;
      }

      int iRB = qtoinclude.indexOf (']',iLB);
      if ( iRB <= 0 ) {
         //  error $$$$
```

```
            WriteLog ( "330E","$$$: " + "[" ) ;
            return false;
         }
         sExtra = qtoinclude.substring (iLB+1,iRB);
         try {
            nExtra = Integer.parseInt(sExtra);
            if ( nExtra < 0 )
               throw new NumberFormatException ();
         } catch ( NumberFormatException nfe ) {
            //  error $$$$
            WriteLog ( "330E","$$$: " + elmt ) ;
            return false;
         }
         qtoinclude = qtoinclude.substring(0,iLB);
      }

      String answer = getParamValue (reqx,qtoinclude);
      int questionValue = 0;
      if ( Misc.isEmpty ( answer ) )
         continue;      //      ignore not answered
      try {
         if ( nExtra < 0 ) {
            questionValue = Integer.parseInt(answer);
         }
         else {
            //  definition.getAnswerText with 3 arguments returns the sExtra extra value
            definition def = getDefinition();
            String questionval = def.getAnswerText (qtoinclude,answer,sExtra);
            questionValue = Integer.parseInt (questionval);
         }
         fscore = fscore + questionValue * fweight ;
         nQuestions ++ ;
      }
      catch ( NumberFormatException nfe ) {
         WriteLog ( "330E","WeightedScore question value not numeric, ignored."
               + " Question: " + qtoinclude + ", value: " + answer
               + ( nExtra >= 0 ? ( ", extra: " + sExtra ) : "" ) ) ;
         continue;
      }
   }
   if ( bAverage && nQuestions > 0 )
      fscore = fscore / nQuestions;
   //     set value to empty if nothing was calculated
   String setvalue = "" ;
   if ( nQuestions > 0 ) {
      NumberFormat nf = NumberFormat.getInstance();
      nf.setMaximumFractionDigits ( decplaces );
      nf.setGroupingUsed(false);
      setvalue = nf.format (fscore);
   }
   setParameterValue ( targetquestion, setvalue );
   setValid (true);
   return true;
}
```

```
String checkArguments (String questionname, String spec) {
	//
	String [] s = Misc.cogixParse (spec, true);
	if ( s == null || s.length < 3 || s[0].length () < 1 || s[1].length () < 1 || s[2].length () < 1 ) {
		return "WeightedScore arguments missing in question " + questionname ;
		}
String targetquestion = s[0];
	int decplaces = 0;
	int kdecperiod = targetquestion.indexOf ('.');
	if ( kdecperiod > 0 )
		targetquestion = targetquestion.substring ( 0, kdecperiod );
	//	Turn on tallying for targetquestion; if error, returns msg
	return setTallying ( targetquestion );
}
}
```

```
package com.cogix.vwf;
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;


//      The ViewsFlash mechanism for publishing extensibility
//      Please use indicatd WriteLog message codes
//      Note addition of "definition def" to most methods 10/9/00

class PublishNotifierExtension extends PublishNotifier {

    void init (ServletConfig conf) {
        Dirs.getErrorLog().WriteLog ("3000I", "initializing " + this.getClass().toString() );
    }

    void destroy () {
    }

    void onPublishChange ( File pollfile, File resultsfile,
                           String spotname, String pollid, Date open, Date close,
                           Spot spot, definition def, Poll pol, ThisCall thiscall ) {
        Dirs.getErrorLog().WriteLog ("3001I", "onPublishChange " + pollid );

        //      Common things to access:
        PrintWriter out = thiscall.out;
        HttpServletRequest req = thiscall.req;
        HttpSession session = req.getSession (false);
        HttpServletResponse res = thiscall.res;
        ServletContext servletContext = Dirs.getServletContext ();
        ServletConfig servletConfig = Dirs.getServletConfig ();

        //      To access custom fields.
        //      note that custom c/b null, custom[x] c/b null
        String valueOfCustomA = def.custom [0]; // customa is 0, customz is 25

        //      Note that the Files being referred to here may not exist yet;
        //      They will be created at "open" time and destroyed at "close" time.
        //      No need to do anything here, but you can if you want to:
        Date now = new Date();
        if ( open != null ) {
            if ( now.after (open) ) {
                //      poll may be open
                if ( close != null ) {
                    if ( now.before (close) ) {
                        // poll is open for sure.
                    }
                    else {
                        ;//     poll is closed for sure
                    }
                }
                else {
```

```
            //  poll is after opening date, and there is no closing date, so it's probably open.
          }
        }
        else {
          //      poll is not open yet
        }
    }
    else {
      //  no opening date, poll should be presumed closed.
    }
}

void onPublishOpen ( File pollfile, File resultsfile,
                     String spotname, String pollid,
                     Spot spot, definition def, Poll pol ) {
    Dirs.getErrorLog().WriteLog ("3002I", "onPublishOpen " + pollid );

    //      Include this code to send an e-mail to the recipients named in
    //      the Spot.emailnotify field ( set in Place settings ) .
    //      This is called twice; once with pollfile != null, one with resultsfile != null.
    if ( pollfile != null ) //      This avoids sending out two emails.
        emailNotification (spot, "Opened", spotname, pollid);

    //      pollfile contains the html of the voting form or page; can be null
    //      resultsfile contains results page; can be null
    //      This method called once for the poll page, once for results page.
    if ( pollfile != null ) {
        FileInputStream fis = null;
        try {
            if ( pollfile.lastModified() > 0 ) {    //      can be compared against new Date() too.
                //  File exists
                long size = pollfile.length();
                fis = new FileInputStream (pollfile) ;
                byte [] content = new byte [(int)size];
                fis.read (content, 0, (int)size );
                fis.close();
                //  Now you can write the file, or do whatever needs to be done.
            }
        }
        catch (IOException ioe) {
            //      Do something like writing to the ViewsFlash log.
            //      Use xxxxI for info, xxxxE for serious error with operator e-mail.
            Dirs.getErrorLog().WriteLog("3003I", "Message " +  ioe.toString());
        }
        finally {
            if ( fis != null )
                try { fis.close (); } catch ( IOException ignored) {}
        }
    }
    //      similar code to above for resultsfile

    //      How to get popular things:
    //      definition def is the survey/poll definition ("Poll page").
    qdefinition qdef = def.findqdefinition("maxparticipants");      //      a question's definition ("Question page")
```

```
    answer ans = (answer) qdef.answers.elementAt(0);                    //    first answer to that question
    String value = ans.text;                                            //    value of that answer

    String pollname = def.pollname; //    same as pollid
    String polltitle = def.polltitle;     //    poll title
    String pollheader = def.pollheader;    //    poll header
    String pollfooter = def.pollfooter;    //    poll footer

    ScheduledPoll sp = spot.getScheduledPoll(pollid);
    Date publishdate = sp.begin;    //    publish date
    Date unpublishdate = sp.end;    //    unpublish date
  }

  void onPublishClose ( File pollfile, File resultsfile,
                String spotname, String pollid,
                Spot spot, definition def, Poll pol ) {
    Dirs.getErrorLog().WriteLog ("3004I", "onPublishClose " + pollid );
    //    Include this code to send an e-mail to the recipients named in
    //    the Spot.emailnotify field ( set in place settings ) .
    //    This is called twice; once with pollfile != null, one with resultsfile != null.
    emailNotification (spot, "Closed and Archived", spotname, pollid);

    //    When the poll is scheduled to Close,
    //    pollfile and resultsfile - similar handling to Open, either could be null
  }

  void onPublishExtra ( File resultsfile,
                String spotname, String pollid,
                Spot spot, definition def, Poll pol ) {
    if ( Dirs.bLogRequests() )
      Dirs.getErrorLog().WriteLog ("3003I", "onPublishExtra " + pollid );
  }

}
```

```
package com.cogix.vwf;
//the mechanism for extensibility when publishing and unpublishing.
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

class VoteNotifierExtension extends VoteNotifier {

	VoteNotifierExtension () {}

	void init (ServletConfig conf) {
//		if ( Dirs.bLogRequests() )	Dirs.getErrorLog().WriteLog ("3001I", "VoteNotifierExtension.init " );
		}

	void destroy () {
		}

	void onVote ( Poll pol, ThisCall thiscall) {
		String pollid = pol.pollid();
//		if ( Dirs.bLogRequests() )	Dirs.getErrorLog().WriteLog ("3002I", "VoteNotifierExtension.onVote " +
pollid );
		//	See OnPublishChange for commonly used code examples
		}
   }
```

```
package com.cogix.vwf;
/**
 * (c) 2001-7 cogix Corporation All Rights Reserved
 * May be used and modified when used with a licensed copy of ViewsFlash
 */

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletResponse;
import java.io.PrintWriter;

/**
 * Example of how to replace J2EE getRemoteUser() authentication with a custom authentication system
 * This example assumes that the logged in user ID is stored in a cookie
 */

public class AuthenticationSkeleton extends RequestNotifier {

    AuthenticationSkeleton () {}

    /*
     * Use this for any global initialization required.
     * This class is instantiated at servlet initialization time, and this method is called only once right after construction
     *
     */
    void init (ServletConfig conf) throws ServletException {
        super.init (conf);
    }

    /*
    * Called during servlet destroy()
    */
    void destroy () throws ServletException {
        super.destroy();
    }

    /*
    * Called at the beginning of request processing
    * if user is already authenticated, save their user ID
    * If not, redirect or present a login form
    *
    */

    public void onRequest ( ThisCall thiscall ) throws ServletException {

        HttpServletRequest request = thiscall.req;
        HttpServletResponse response = thiscall.res;

        /*
            Determine if the URL requires simulated J2EE authentication or not.
            In a J2EE environment, this is done using web.xml security-constraint elements; be sure to remove such
```

constraints from web.xml

servlet/vfadmin is the application, and require authentication. appsecurity=user must be set in viewflash.properties
servlet/vfauth is for questionnaires that use the Basic authentication option; these require authentication
servlet/viewsflash is for questionnaires and other commands that do not use J2EE authentication. Do not authenticate these.
*/

```
String requesturi = request.getRequestURI();   //  ViewsFlash/servlet/vfadmin, etc
if ( requesturi == null ||  requesturi.endsWith("servlet/viewsflash") )
   return; //  no authentication required

//  Determine which is the correct logged in state.
String cookiename = "COOKIE";
String userid;
Cookie[] c = request.getCookies();
for ( int k = 0 ; c!= null &&  k < c.length; k++ ) {
   Cookie ck = c [k];
   if ( cookiename.equals(ck.getName())) {
      userid = ck.getValue();
      thiscall.userid = userid;
      //  Tell ViewsFlash that this is the user ID of the logged in user.
      // ViewsFlash uses this value instead of request.getRemoteUser();

      //  Optional: enable this to allow requesting this authentication method explicitly in Security page,
      // in place of "Portal". See getUIText() , below.  There is no visible benefit in allowing this.
      //  thiscall.req.setAttribute (RequestNotifierId, "");
      return;
   }
}

//  User is not logged in.
//  Two ways to present the user with a log in  form.
//  Method 1: use a redirect
//  Method 2: write the login form now
boolean bRedirectMethod = true;
boolean bWriteTheForm  = false;

if ( bRedirectMethod ) {
   //  Redirect to a login form, passing along the original request.
   // The login form, when successful, should set the cookie and redirect to the original requests' URL
   String originalurl = request.getRequestURL().toString();  //  Ok in Java 1.4 and 1.5
   String redirecturl = "http://www.yourserver.com/Authenticate?redirect=" + originalurl;

   redirecturl = response.encodeRedirectURL (redirecturl);
   thiscall.bRediraftersetattribute = true;    //  signals redirect wanted immediately upon return
   thiscall.rediraftersetattribute = redirecturl ;  //  where to redirect to
}
else if ( bWriteTheForm ) {
   PrintWriter out = thiscall.out;
   out.print("The login form");
   //  do not close out
   thiscall.bEndRequestProcessing = true;  //  ends request processing immediately upon return
```

```
        }
    }

    public String getUIText () {
        return "<b>Custom authentication</b>";
    }

}
```

```
package com.cogix.vwf;

import javax.servlet.*;
import java.sql.*;

public class DBLookupRole extends RoleLookup {

   //  Test table created with
   //  CREATE TABLE VWFUSERROLES ( VFUSER VARCHAR (32) NOT NULL, VFROLE VARCHAR (32)
NOT NULL );
   //  CREATE INDEX USERINDEX ON VWFUSERROLES (VFUSER);
   //  CREATE INDEX ROLEINDEX ON VWFUSERROLES (VFROLE);

   //  Customize the following line
   private static final String userinroleQuery
         = "SELECT COUNT(*) FROM VWFUSERROLES WHERE VFUSER ='%1' AND VFROLE='%2'";

   void init() throws UnavailableException {
      if ( Dirs.getdbStore() == null )
         throw new UnavailableException ("Database connector not available");
   }

   //  Query: select count(*) from tablename where userid=userid and role=role

   //  Extensions must use userid explicitly, not tc.getremoteuser!
   //  ThisCall is available in case it's needed
   //  Let it throw any errors encountered
   boolean bIsUserInRole(ThisCall tc, String userid, String role) throws Exception {
      Statement stmt = null;
      ResultSet rs = null;
      int count = -1;
      String query = userinroleQuery;
      dbStore dbs = Dirs.getdbStore();
      Connection con = null;
      try {
         if ( Misc.isEmpty (userid) || Misc.isEmpty (role) )
            return false;

         query = Misc.stringReplace ( userinroleQuery, "%1", userid );
         query = Misc.stringReplace ( query, "%2", role );

         con = dbs.getConnection();  //  from pool
         stmt = con.createStatement();
         rs = stmt.executeQuery (query);
         if ( rs.next() ) {
            count = rs.getInt (1);
            return count > 0;
         }
      } catch (Throwable e) {
         //  To disable logging, remove this:
         Dirs.WriteLog ("631I","Exception looking up user in role with " + query + ": " + e.toString() );
         return false;
      } finally {
```

```
        if ( rs != null ) try { rs.close(); } catch (Exception ignored) {}
        if ( stmt != null ) try { stmt.close(); } catch (Exception ignored) {}
        dbs.freeConnection (con);   //  back to pool, null is ok

        if ( Dirs.bLogRequests3 () ) {
           String msg = "DBLookupRole.bIsUserInRole: " + query + " returned " + count ;
           Dirs.WriteLog ("667I", msg );
        }

      }
      return false;
   }

}
```

```
package com.cogix.vwf;
import java.util.*;
import java.sql.*;
import java.io.*;
import javax.servlet.*;
import java.text.*;

/**
 * Prerequisite:
 * ViewsFlash 5 or higher
 *
 * Revised April 2008
 */

class DatabaseExtension {
   protected dbStore db;
   protected DateFormat dfdate = Dirs.getDateFormat();
   protected DateFormat dftime = Dirs.getTimeFormat();

   protected int coltype;
   protected String coltypn;
   int typeblob = java.sql.Types.CLOB; //  2005: DB2
   int typevarchar = java.sql.Types.VARCHAR;   //  12: Oracle
   int typelongvarchar = java.sql.Types.LONGVARCHAR;   //  -1 MySql

   public String TableName;
   String OverridenTableName;
   String UnqualifiedTableName;

   public boolean bAnswerIsBlob;
   //  Returns whether answers are searchable; not so when BLOB has been set
   public boolean bFilterAnsweredOnly () {
      return bAnswerIsBlob;
   }

   boolean bDisableNativeDatabaseSave () {
      return false;
   }


   //  Will be written to the log with the class name
   String getVersion () {
      return "0.0";
   }

   void preinit (ServletConfig conf) {
      //  Do nothing in generic DB
   }

   public String TableName () {
      if ( this.TableName == null )  {
         UnqualifiedTableName = OverridenTableName != null ? OverridenTableName : "VWFDATA";
         TableName = Dirs.getdbStore ().getTableName ( UnqualifiedTableName);
```

```
        }
        return TableName;
    }
    //  Called at end of servlet initialization
    void init (ServletConfig conf) throws ServletException {
        db = Dirs.getdbStore(); //      must do this to use built-in database access
        if ( db == null )              //      Quit like this on serious error:
            throw new ServletException ("Database Extension requires using a database");
        Connection con = null;
        try {
            con = db.getConnection();
            TableName() ;   //  Primes UnqualifiedTableName
            if ( ObjectStorage.TableExists(con, UnqualifiedTableName ) == null )
                return;
            createTables () ;   //  SQL Statements are written just to the log, unfortunately
        } catch (SQLException e) {
            throw new ServletException (e.toString());
        } finally {
            db.freeConnection(con);
        }
    }

    void destroy () {
    }

    //  return true if tables aren't there
    boolean TablesNeedCreating () {
        return false;
    }

    //  Create the tables and return the SQL used if Ok, throw otherwise
    String createTables () throws SQLException {
        throw new SQLException ("No DatabaseExtension class provided. Use servlet parameter 'databaseextension= '
to disable. ");
    }

    //  Return SQL if Ok, or message if not Ok. Do not throw SQLException.
    String removeTables () {
        return "No extensible class provided";
    }

    //  This text describes an additional button on Save page; mandatory
    String getSavePageMessage () {
        return "Save in Normalized Database";
    }

    //  Save one entire survey response.
    //  Return null if Ok, or error message otherwise.
    String saveOneResponse ( String spotname, String pollname, String authenticateduserid, Vector items ) throws
SQLException {
        return "No extensible class provided";
    }
```

```
   //  return null if Ok, msg otherwise
   String getOneResponse ( String voteid, Vector fields, Vector build, boolean bUsePre, String spotname, String
pollname ) {
      return "No extensible class provided";
   }

   //  Remove all data associated with a given pollid. Called when questionnaire  removed
   //        or when [x] Remove Data is checked on Publish page.
   //  return null if Ok, msg otherwise
   String removeAllResponses ( String spotname, String pollname ) {
      return "No extensible class provided";
   }

   //  Rename and Move a place.  If newspotname is null, don't use it; same with newpollname.
   //  return null if Ok, msg otherwise
   String moveAllResponses ( String spotname, String newspotname, String pollname, String newpollname ) {
      return "Not implemented";
   }

   public String SQLgetvoteidsforDataGathering(dbStore db, String spotname, String pollname, String field, String id)
{
      //  Note that all parameters must be checked for possible SQL injection before invoking this method
      String sqlgetvoteids;
      //  This is the query to use, produces the most recent entry for a given user.
      //  SELECT VOTE_ORDINAL from vwfdata where spotname='ccsv' and pollname='one' AND
AUTHENTICATEDUSERID='2pQLyr0z9Y' AND ROWNUM <=1 order by TIME_STAMP
      sqlgetvoteids = "SELECT VOTE_ORDINAL, MIN (TIME_STAMP) FROM " + db.getTableName
("VWFDATA") + " WHERE " +
            "SPOTNAME='" + spotname +
            "' AND POLLNAME='" + ObjectStorage.storedpollid (pollname) +
            "' AND " + field + "='" + id + "' GROUP BY VOTE_ORDINAL";
      return sqlgetvoteids;
   }

   public String SQLgetvoteids(dbStore db, String spotname, String pollname, String filterwhereclause ) {
      //  Note that all parameters must be checked for possible SQL injection before invoking this method
      String sqlgetvoteids;
      sqlgetvoteids = "SELECT VOTE_ORDINAL, MIN (TIME_STAMP) FROM " + TableName() + //
db.getTableName ("VWFDATA") +
            " WHERE SPOTNAME='" + spotname +
            "' AND POLLNAME='" + ObjectStorage.storedpollid (pollname) + "'" +
            filterwhereclause +
            " GROUP BY VOTE_ORDINAL";
      if ( Dirs.bLoggingDB() ) {
         Dirs.WriteLog ("7788I","SQLgetvoteids using " + sqlgetvoteids );
      }
      return sqlgetvoteids;
   }

   public String SQLFilternotresponded(dbStore db, definition def, String filtid) {
      String xxx = "AND VOTE_ORDINAL NOT IN "
            + "( SELECT VOTE_ORDINAL FROM "
            + db.getTableName ("VWFDATA")
            + " WHERE SPOTNAME='" + def.spotname
```

```
        + "' AND POLLNAME='" + ObjectStorage.storedpollid (def.pollname) + "'"
        + " AND QUESTIONNAME='" + filtid + "' ) ";
   return xxx;
}

//  Overload when using CLOBs
public String getFilterColumnName ( boolean bIsLongText ) {
   return "ANSWER";
}

public String  getOneValue ( String sqlquery ) throws SQLException {
   Connection con = null;
   Statement getoneresponse = null;
   ResultSet questions = null;
   try {
      con = db.getConnection();
      getoneresponse = con.createStatement();
      questions = getoneresponse.executeQuery(sqlquery);
      if ( questions.next() ) {
         return questions.getString(1);
      }
      return null;
   }
   finally {
      db.close(questions);
      db.close(getoneresponse);
      db.freeConnection (con);
   }
}

public int UpdateOneAnswer(String sqlupdatestmt, String value, boolean bShouldBeClob) throws SQLException {
   throw new SQLException("UpdateOneAnswer method not implemented" );
}

void InsertOneAnswer(UpdateNormalized unz, String questionname, String content) throws SQLException {
    throw new SQLException("InsertOneAnswer method not implemented" );
}

int RemoveOneAnswer ( String sqlstmt ) throws SQLException {
   Connection con = null;
   Statement removeanswer = null;
   try {
      con = db.getConnection();
      removeanswer = con.createStatement();
      return removeanswer.executeUpdate (sqlstmt);
   }
   finally {
      db.close(removeanswer);
      db.freeConnection (con);
   }
}

public boolean bCanUpdateInPlace () {
   return false;
```

```
    }

    public String TrimToMaxVWFDATASize(String content) {
        return content;
    }

    boolean bIsTextField ( String pollid, String qname )  {
        definition def;
        try {
            def = Dirs.getDefinitionIfExists (pollid);
        } catch (vwfException e) {
            return false;
        }
        if ( def == null )
            return false;
        qdefinition qdef = def.findqdefinition(qname);
        return qdef != null && qdef.isLongText ();
    }

}
```

Quizzes, tests and assessments

To construct quizzes and assessments with ViewsFlash, follow these guidelines.

Create a multiple-page questionnaire with the test questions

Create radio button and checkbox questions with extra values to indicate correct and incorrect answers

Assign numeric values to single-choice (radio button, drop-down menu) answers

Use branching to ask in depth questions based on previous answers

Use question piping to alter the text of questions based on previous answers

Create hidden questions for calculating scores

Calculate scores using the Script action

Use question piping in the response page to display calculated scores

For examples, contact Cogix.

Next: XML data exchange

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is

modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the

ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on

which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not

excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status

of all derivatives of our free software and of promoting the sharing
and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the library, if
necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the
library `Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice

That's all there is to it!

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is

modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the

ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on

which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not

excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status

of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Spell Checking Oriented Word Lists (SCOWL)
Revision 6
August 10, 2004
by Kevin Atkinson

The SCOWL is a collection of word lists split up in various sizes, and other categories, intended to be suitable for use in spell checkers. However, I am sure it will have numerous other uses as well.

The latest version can be found at http://wordlist.sourceforge.net/

The directory final/ contains the actual word lists broken up into various sizes and categories. The r/ directory contains Readmes from the various sources used to create this package.

The other directories contain the necessary information to recreate the word lists from the raw data. Unless you are interested in improving the words lists you should not need to worry about what's here. See the section on recreating the words lists for more information on what's there.

Except for the special word lists the files follow the following naming convention:
  <spelling category>-<classification>.<size>
Where the spelling category is one of
  english, american, british, british_z, canadian,
  variant_0, varaint_1, variant_2
Classification is one of
  abbreviations, contractions, proper-names, upper, words
And size is one of
  10, 20, 35 (small), 40, 50 (medium), 55, 60, 70 (large),
  80 (huge), 95 (insane)
The special word lists follow are in the following format:
  special-<description>.<size>
Where description is one of:
  roman-numerals, hacker

When combining the words lists the "english" spelling category should be used as well as one of "american", "british", "british_z" (british with ize spelling), or "canadian". Great care has been taken so that that only one spelling for any particular word is included in the main list. When two variants were considered equal I randomly picked one for inclusion in the main word list. Unfortunately this means that my choice in how to spell a word may not match your choice. If this is the case you can try including the "variant_0" spelling category which includes most variants which are considered almost equal. The "variant_1" spelling category include variants which are also generally considered acceptable, and "variant_2" contains variants which are seldom used.

The "abbreviation" category includes abbreviations and acronyms which are not also normal words. The "contractions" category should be self explanatory. The "upper" category includes upper case words and proper

names which are common enough to appear in a typical dictionary. The "proper-names" category included all the additional uppercase words. Final the "words" category contains all the normal English words.

To give you an idea of what the words in the various sizes look like here is a sample of 25 random words found only in that size:

10: began both buffer cause collection content documenting easiest equally examines expecting first firstly hence inclining irrelevant justified little logs necessarily ought sadly six thing visible

20: chunks commodity contempt contexts cruelty crush dictatorship disgusted dose elementary evolved frog god hordes notion overdraft overlong overlook phoning poster recordings sand skull substituted throughput

35: aliasing blackouts blowout bluntness corroborated derrick dredging elopements entrancing excising fellowship flagpole germination glimpse gondola guidebook madams minimalism minnows partisans petitions shelling swarmed throng welding

40: altercation blender castigation chump coffeehouse determiners doggoning exhibitor finders flophouse gazebo lumbering masochism mopeds poetically pubic refinance reggae scragglier softhearted stubbornness teargassed township underclassman whoosh

50: accumulative adulterant allegorically amorousness astrophysics camphor coif dickey elusiveness enviousness fakers fetishistic flippantly headsets liefs midyears myna pacification persiflage phosphoric pinhole sappy seres unrealistically unworldly

55: becquerel brickie centralist cine conveyancing courgette disarmingly garçon gobstopper infilling insipidity internationalist kabuki lyrebirds obscurantism rejigged revisionist satsuma slapper sozzled sublieutenants teletext vino wellness wracking

60: absorber acceptableness adventurousness antifascists arrhythmia audiology cartage cruses fontanel forelimbs granter hairlike installers jugglery lappets libbers mandrels micrometeorite mineshaft reconsecrates saccharides smellable spavined sud timbrel

70: atomisms benedict carven coxa cyanite detraining diazonium dogberry dogmatics entresol fatherlessnesses firestone imprecator laterality legitimisms maxwell microfloppies nonteaching pelerine pentane pestiferousness piscator profascist tusche twirp

80: cotransfers embrangled forkednesses giftwrapped globosity hatpegs hepsters hermitess interspecific inurbanities lamiae literaehumaniores literatures masulas misbegun plook prerupt quaalude rosanilin sabbatism scowder subreptive thumbstalls understrata yakows

95: anatropal anientise bakshi brouzes corsie daimiote dhaw dislikened
ectoretina fortuitisms guardeen hyperlithuria nonanachronistic
overacceleration pamphletic parma phytolith starvedly
trophoplasmic ulorrhagia undared undertide unplunderously
unworkmanly vasoepididymostomy

And here is a rough count on the number of words in the "english" spelling category for each size:

Size	Words	Proper Names	Running Total
10	5,000		5,000
20	8,700		14,000
35	34,500	200	48,000
40	6,000	500	55,000
50	23,200	17,200	95,000
55	7,500		103,000
60	16,000	12,800	132,000
70	45,100	34,300	211,000
80	137,000	30,400	379,000
95	198,000	51,800	628,000

(The "Words" column does not include the proper name count.)

Size 35 is the recommended small size, 50 the medium and 70 the large. Sizes 70 and below contain words found in most dictionaries while the 80 size contains all the strange and unusual words people like to use in word games such as Scrabble (TM). While a lot of the the words in the 80 size are not used very often, they are all generally considered valid words in the English language. The 95 contains just about every English word in existence and then some. Many of the words at the 95 level will probally not be considered valid english words by most people. I don't recommend anyone use levels above 70 for spell checking as they contain rarely used words which can hide misspellings of similar more commonly used words. For example the word "ort" can hide a common typo of "or". No one should need to use a size larger than 80, the 95 size is labeled insane for a reason.

Accents are present on certain words such as café in iso8859-1 format.

CHANGES:

From Revision 5 to 6 (August 10, 2004)

Updated to version 4.0 of the 12dicts package.

Included the 3esl, 2of4brif, and 5desk list from the new 12dicts package. The 3esl was included in the 40 size, the 2of4brif in the 55 size and the 5desk in the 70 size.

Removed the Ispell word list as it was a source of too many errors. This eliminated the 65 size.

Removed clause 4 from the Ispell copyright with permission of Geoff

Kuenning.

Updated to version 4.1 of VarCon.

Added the "british_z" spelling category which it British using the "ize" spelling.

From Revision 4a to 5 (January 3, 2002)

Added variants that were not really spelling variants (such as forwards) back into the main list.

Fixed a bug which caused variants of words to incorrectly appear in the non-variant lists.

Moved rarly used inflections of a word into higher number lists.

Added other inflections of a words based on the following criteria
If the word is in the base form: only include that word.
If the word is in a plural form: include the base word and the plural
If the word is a verb form (other than plural): include all verb forms
If the word is an ad* form: include all ad* forms
If the word is in a possessive form: also include the non-possessive

Updated to the latest version of many of the source dictionaries.

Removed the DEC Word List due to the questionable licence and because removing it will not seriously decrese the quality of SCOWL (there are a few less proper names).

From Revision 4 to 4a (April 4, 2001)

Reran the scripts on a never version of AGID (3a) which fixes a bug which caused some common words to be improperly marked as variants.

From Revision 3 to 4 (January 28, 2001)

Split the variant "spelling category" up into 3 different levels.

Added words in the Ispell word list at the 65 level.

Other changes due to using more recent versions of various sources included a more accurete version of AGID thanks to the word of Alan Beale

From Revision 2 to 3 (August 18, 2000)

Renamed special-unix-terms to special-hacker and added a large number of communly used words within the hacker (not cracker) community.

Added a couple more signature words including "newbie".

Minor changes due to changes in the inflection database.

From Revision 1 to 2 (August 5, 2000)

Moved the male and female name lists from the mwords package and the DEC name lists form the 50 level to the 60 level and moved Alan's name list from the 60 level to the 50 level. Also added the top 1000 male, female, and last names from the 1990 Census report to the 50 level. This reduced the number of names in the 50 level from 17,000 to 7,000.

Added a large number of Uppercase words to the 50 level.

Properly accented the possessive form of some words.

Minor other changes due to changes in my raw data files which have not been released yet. Email if you are interested in these files.

COPYRIGHT, SOURCES, and CREDITS:

Alan Beale <biljir@pobox.com> also deserves special credit as he has, in addition to providing the 12Dicts package and being a major contributor to the ENABLE word list, given me an incredible amount of feedback and created a number of special lists (those found in the Supplement) in order to help improve the overall quality of SCOWL.

The 10 level includes the 1000 most common English words (according to the Moby (TM) Words II [MWords] package), a subset of the 1000 most common words on the Internet (again, according to Moby Words II), and frequently class 16 from Brian Kelk's "UK English Wordlist with Frequency Classification".

The MWords package was explicitly placed in the public domain:

You can verify the public domain status by contacting

Grady Ward
3449 Martha Ct.
Arcata, CA 95521-4884

grady@netcom.com
grady@northcoast.com

The "UK English Wordlist With Frequency Classification" is also in the Public Domain:

Date: Sat, 08 Jul 2000 20:27:21 +0100
From: Brian Kelk <Brian.Kelk@cl.cam.ac.uk>

> I was wondering what the copyright status of your "UK English
> Wordlist With Frequency Classification" word list as it seems to
> be lacking any copyright notice.

There were many many sources in total, but any text marked "copyright" was avoided. Locally-written documentation was one source. An earlier version of the list resided in a filespace called PUBLIC on the University mainframe, because it was considered public domain.

Date: Tue, 11 Jul 2000 19:31:34 +0100

> So are you saying your word list is also in the public domain?

That is the intention.

The 20 level includes frequency classes 7-15 from Brian's word list.

The 35 level includes frequency classes 2-6 and words appearing in at least 11 of 12 dictionaries as indicated in the 12Dicts package. All words from the 12Dicts package have had likely inflections added via my inflection database.

The 12Dicts package and Supplement is in the Public Domain.

The WordNet database, which was used in the creation of the Inflections database, is under the following copyright:

The 40 level includes words from Alan's 3esl list found in version 4.0 of his 12dicts package. Like his other stuff the 3esl list is also in the public domain.

The 50 level includes Brian's frequency class 1, words words appearing in at least 5 of 12 of the dictionaries as indicated in the 12Dicts package, and uppercase words in at least 4 of the previous 12 dictionaries. A decent number of proper names is also included: The top 1000 male, female, and Last names from the 1990 Census report; a list of names sent to me by Alan Beale; and a few names that I added myself. Finally a small list of abbreviations not commonly found in other word lists is included.

The name files form the Census report is a government document which I don't think can be copyrighted.

The file special-jargon.50 uses common.lst and word.lst from the "Unofficial Jargon File Word Lists" which is derived from "The Jargon File". All of which is in the Public Domain. This file also contain a few extra UNIX terms which are found in the file "unix-terms" in the special/ directory.

The 55 level includes words from Alan's 2of4brif list found in version 4.0 of his 12dicts package. Like his other stuff the 2of4brif is also in the public domain.

The 60 level includes Brian's frequency class 0 and all words appearing in at least 2 of the 12 dictionaries as indicated by the 12Dicts package. A large number of names are also included: The 4,946 female names and the 3,897 male names from the MWords package.

The 70 level includes the 74,550 common dictionary words and the 21,986 names list from the MWords package The common dictionary words,

like those from the 12Dicts package, have had all likely inflections added. The 70 level also included the 5desk list from version 4.0 of the 12Dics package which is the public domain

The 80 level includes the ENABLE word list, all the lists in the ENABLE supplement package (except for ABLE), the "UK Advanced Cryptics Dictionary" (UKACD), the list of signature words in from YAWL package, and the 10,196 places list from the MWords package.

The ENABLE package, mainted by M\Cooper <thegrendel@theriver.com>, is in the Public Domain:

> The ENABLE master word list, WORD.LST, is herewith formally released into the Public Domain. Anyone is free to use it or distribute it in any manner they see fit. No fee or registration is required for its use nor are "contributions" solicited (if you feel you absolutely must contribute something for your own peace of mind, the authors of the ENABLE list ask that you make a donation on their behalf to your favorite charity). This word list is our gift to the Scrabble community, as an alternate to "official" word lists. Game designers may feel free to incorporate the WORD.LST into their games. Please mention the source and credit us as originators of the list. Note that if you, as a game designer, use the WORD.LST in your product, you may still copyright and protect your product, but you may *not* legally copyright or in any way restrict redistribution of the WORD.LST portion of your product. This *may* under law restrict your rights to restrict your users' rights, but that is only fair.

UKACD, by J Ross Beresford <ross@bryson.demon.co.uk>, is under the following copyright:

> Copyright (c) J Ross Beresford 1993-1999. All Rights Reserved.
>
> The following restriction is placed on the use of this publication: if The UK Advanced Cryptics Dictionary is used in a software package or redistributed in any form, the copyright notice must be prominently displayed and the text of this document must be included verbatim.
>
> There are no other restrictions: I would like to see the list distributed as widely as possible.

The 95 level includes the 354,984 single words and 256,772 compound words from the MWords package, ABLE.LST from the ENABLE Supplement, and some additional words found in my part-of-speech database that were not found anywhere else.

Accent information was taken from UKACD.

My VARCON package was used to create the American, British, and Canadian word list.

Since the original word lists used used in the VARCON package came from the Ispell distribution they are under the Ispell copyright:

The variant word lists were created from a list of variants found in
the 12dicts supplement package as well as a list of variants I created
myself.

The Readmes for the various packages used can be found in the
appropriate directory under the r/ directory.

FUTURE PLANS:

There is a very nice frequency analyse of the BNC corpus done by
Adam Kilgarriff. Unlike Brain's word lists the BNC lists include part
of speech information. I plan on somehow using these lists as Adam
Kilgarriff has given me the OK to use it in SCOWL. These lists will
greatly reduce the problem of inflected forms of a word appearing at
different levels due to the part-of-speech information.

I also plan on perhaps putting the data in a database and use SQL
queries to create the wordlists instead of tons of "sort"s, "comm"s,
and Perl scripts.

RECREATING THE WORD LISTS:

In order to recreate the word lists you need a modern version of Perl, bash, the traditional set of shell utilities, a system that supports symbolic links, and quite possibly GNU Make. Once you have downloaded all the necessary raw data in the r/ directory you should be able to type "rm final/* && make all" and the word lists in the final/ directory should be recreated. If you have any problems fell free to contact me; however, unless you are interested in improving the scripts used, I will likely ignore you as there should be little need for anyone not interested in improving the word list to do so.

The src/ directory contains the numerous scripts used in the creation of the final product.

The r/ directory contains the raw data used to create the final product. In order for the scripts to work various word lists and databases need to be created and put into this directory. See the README file in the r/ directory for more information.

The l/ directory contains symbolic links used by the actual scripts.

Finally, the working/ directory is where all the intermittent files go that are not specific to one source.

XStream is open source software, made available under a BSD license.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of XStream nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Thank you for purchasing flag icons.

Agreement

This is a legal agreement between you, the purchaser, and the IconDrawer.com. By purchasing or downloading any royalty-free icons from our website you agree to the following: All of the icons remain the property of IconDrawer.com. The icons can be used royalty-free by the licensee for any personal, commercial project including web design, software, application, advertising, film, video, and computer game. The license does not permit the following uses: The icon may not be sublicensed, resold, rented, transferred, or otherwise made available for use or detached from a product, software application, or web page. The Icons may not be placed on any electronic bulletin board or downloadable format. You may not include the icons in any web template, including those which are web based, but not limited to website designs and presentation templates. You may not use,or allow anyone else to use The Icons to create pornographic, libelous, obscene, or defamatory material. All icon files are provided "As is." you agree not to hold IconDrawer.com liable for any damages that may occur during to use, or inability to use, icons or image data from IconDrawer.com.

Example of Validation, Branching and Scoring

Note: This example does not use the new custom action Script, which is easier to use.

Open the **Example and Index** displays.

The display on the left is the Questionnaire Preview, showing what the survey will look like when it is presented to the participant. Page Breaks indicate what questions will be shown on what page. The buttons at the bottom will be presented as appropriate, e.g., no Previous button on the first page. The progress meter starts at 0% complete. If a question is Required or has a Range of valid values, pressing Submit will carry out the desired validation checks.

The display on the right is the Questionnaire Index, showing a bird-eye's views of the questions, their numbers and names, and the values that are recorded for the answers. If the question must be answered or has a set range, it is indicated, for example, as Required. Must be 18-30. Custom Actions are displayed in red. This display is a tremendous aid when dealing with larger and more complex surveys.

Notice how the is_drinker, is_smoker and is_exercise Custom Actions prevent their pages from displaying when the response to the initial question indicates that the visitor doesn't drink, smoke, or exercise. Notice how the if_covered Custom Action branches skips to show_score if the answer to 'insurance' indicates that the individual has adequate insurance.

This example also illustrates the use of WeightedScore in the calc_score question, where it calculates the sum of the values entered for questions amount_drink, amount_smoke and amount_exercise, weighted by appropriate values, and stores the result in a hidden question risk_score. This calculated value is used in the last page in question show_score, which uses question piping to show the value of the calculated score as [/risk_score]. Note that it is necessary to turn on the Tally option for risk_score to use it in Univariate and Crosstabulation analysis.

Next: Invite and Track

Webcasting with surveys

By combining a set of frames, JavaScript, new API commands, and new Web Survey options, it is now possible to conduct live polls or surveys while an online audio or video presentation is presented at the same time.

1. Create a place. Use the WebcastSurveyPage poll style template. In option 6, choose "Use unique poll names". Create and use a multipage survey template with the history parameter set to redirect to a standby page after each vote, such as:
<input type="hidden" name="history"
value="1,WebcastCookie,[/spotname],http://www.yourco.com/ViewsFlash/webcast/standby.html">

Set option 5 to write files to the webcast/events directory.

2. Create a multi-page survey. Each page of the survey will be available for presentation separately during the webcast.

In Security #3, you must use an Authentication method; if you have no preference, choose "Use a servlet session value".

It is also recommended to turn on duplicate checking, using the "User Authentication method specified below".

In the Save page, make sure all fields you are interested in are saved, including if appropriate, the unique user id.

Publish the survey, including appropriate Start and End times; the surveys are not available until the Start time.

3. Become familiar with the files in /ViewsFlash/webcast. Copy all these files to a working directory and customize them to suit your needs. SurveyPresentation.html is frame set configured for presentation of a survey during a webcast. The left frame uses other.html, which you should customize to show a video stream, for example. The right frame uses SurveyFrame.html, which is a frameset for presenting a survey in synchronization with the webcast moderator.

Once you have tested your modifications and you're ready for the webcast, give participants the URL to SurveyPresentation.html.

4. In Actions, click on Webcast. On this page you have the following options:

Start	Click on this shortly before the start of the webcast. This clears the question display and marks the beginning of the webcast.
Blank	Click on this anytime during the presentation to remove the question display.
Show Page Show Page » Show Page	Click on one of these links to advance the presentation to that page. The question currently shown is marked » and the question text is listed.
End	Click on this to clear the question display and mark the end of the webcast.

Webcast Participants should open a browser window go to the survey presentation you just created. Its URL is:

/ViewsFlash/webcast/SurveyPresentation.html?eventid=SSSS&surveyspot=PPPP

where SSSS is the name of the survey and PPPP is the name of the place. Note: surveyspot=PPPP is optional. If omitted, the default place of 'WebcastSurveys' will be used.

In this window, you will see the results of clicking on each of the options above.

5. For best results, create an introductory and and ending poll that say "Welcome" and "Thank you" (use the 'Just HTML' question type).

Webcast Support

IMPORTANT NOTE: To enable Webcast Support, the Administrator must install ViewsFlash with the webcast=1 servlet parameter.

Cogix ViewsFlash can be used to ask participants questions during live webcasts. When viewing an archived webcast, the questions can be presented at the same point in the presentation as the original performance.

Both surveys and polls can be combined with a webcast, with different features.
If you have to pick one, we recommend Surveys:

Feature	Polls	Surveys
Show live results after each question	Yes	No
Show question during archived webcast	No	Yes
Store data as one table	No	Yes
Analyze data	No	Yes
Built using	place	Survey
What is being shown each time	A poll	A page of questions
Step-by-step instructions	**For polls**	**For surveys**

To make all this work, you need to create a set of frames. A frame will host the video or audio stream; another frame will host the ViewsFlash polls; other frames could host other applications. We have provided complete framesets that you can adapt for your own use.

During the webcast, each visitor's browser continually asks the ViewsFlash server "what is the current question". A presenter's console provides a way to "Mark" the current question, so that visitor's browsers can then show it. The latency between presenter choices and visitor experiences depends on how often this query is made, and can be adjusted.

Next: **Webcast Polls**

Integrating with Application Servers

This page is obsolete. Please go to **How to use the Web Services API**.

Next: Embedding Questionnaires in JSP pages

ViewsFlash 2.93 Release Notes

The current release is ViewsFlash 2.93, Build 918. The software build number appears at the left of the Administration screen. All suggestions for clarification are welcome. **Bugs Fixed** are at the bottom of the page. In the documentation, items introduced in 2.9 through 2.93 are marked .

Release 2.93 added significant functionality:

Top N display. New style tags allow showing a different display for the top 3 winners in a contest, for example, or anyone who receives more than 15% of the vote.

Time remaining display. New style tags allow showing a poll or survey's opening and closing time, as well as the time remaining before polls close, in real time. They can be used in both poll and response pages.

Easy redirection. An option in the Response page lets you redirect the visitor to a different page after they vote. Previously, this required the use of the history tag; that still works, but this makes it easier to do. All the same redirection options as the Security page are available, and no template changes are required to use this option.

Very frequent extra reports can now be produced every second, minute, or hour. Reports can be included in other pages using server side includes.

Extra fields. Each answer to a question can now have additional fields, such as a URL or the name of a gif. Makes it very easy to construct attractive presentations.

Enhanced Java Extensibility. Similar to the Validator classes, these new application "hooks" let you execute your own Java code when a poll or survey is published (OnPublish) , and when a valid vote is received (OnVote).

Synchronization. In distributed configurations where a shared file system is not allowed, this feature replicates data between the different machines' file systems automatically in real time, using an HTTP protocol. Contact Cogix for further information.

Release 2.92 is a minor release with incremental improvements:

Extraneous values in voting forms are ignored.

Live Results has been enhanced to show question names; percentages add up to 100.

When answers' values are changed after Publishing, a subsequent Publish that resets counts removes obsolete values from results lists.

Release 2.91 is a minor release with incremental improvements:

A **monitoring API** that can be used by data center administrators to diagnose the health of a ViewsFlash machine.

A "**Write-in**" choice automatically adds an "Other" button and a text field to a multiple choice question. New Poll Style Template tags [/ifonclick] and [/vfonclick] added to support this capability, and they are included in the Standard_ Poll Style template.

A **new method** for using a single cookie to detect duplicate voting in many polls. See **Cookie Flavors**.

Save the IP address of a visitor by entering "ipaddress" in the Cookie Name field in the Save page.

Enhanced Custom Validation. You can now access all values in a request when validating a vote, such as the cookies and the HTTP headers. Custom validation classes compiled with versions prior to 2.91 should be recompiled.

New style templates. All templates have been revised to use HTML classes and style sheets, in keeping with contemporary HTML usage.Templates are now installed in a directory named styles2, so as not to conflict with previous templates, which continue to work unchanged.

Zero All button at the bottom of the place page allows resetting the counts of all current and future polls at once.

Release 2.9 is a major release combining usability enhancements and relational database support. Main enhancements are:

Application security. A new optional servlet parameter can be used to require passwords to be re-entered after a specified interval.

Database support. With this add-on module, you can save each collected survey as one database record. You can also save

results dynamically moment by moment and use the live information in real-time systems.

Usability improvements. Menu wordings are shorter. places and style templates are listed in alphabetical order. Polls are sorted by start times. Save, Tally and E-mail pages include a Select All button.

Default settings. All polls in the Master polling place are now listed in the "Copy options from" pull-down that you use when creating a poll. If you pick one, the next time you create a poll ViewsFlash remembers which one you picked the last time. These two features, combined, make it much easier to create standard polls and use them throughout the site.

Publish override. When you Publish a poll, if the dates conflict with a previously scheduled poll, a pop-up window gives you the option to reschedule the other polls automatically.

Question editing. You can now copy a question and its values, rearrange answers up and down, and edit an answer's value.

Field sizes. You can specify the width of a text field and the height and width of a text box right in the question.

View results as spreadsheet. On the Live Results page, you can view your stored data in the browser, as before, or in your spreadsheet. The data is displayed as a .csv file which most browsers translate into a sequence of operations that brings up excel and loads the data right into it. You can view either the data saved to ASCII files or the data saved to the database.

API Enhancements and new Reference Guide. Several new API commands are provided that allow discovering what polls are scheduled and when. The "**history**" parameter for redirection now substitutes URLs such as ?action=r and preserves other parameters in the command tail.

Bugs fixed.

In 2.91
When a poll or survey is removed permanently, its database table is now removed if ViewsFlash created it originally.

When poll settings were changed after publishing, satellites were not always aware of the changes.

In 2.90

Cookies. When a poll was published without an Ending date, cookies used in Security sometimes would expire immediately. This did not affect cookies thrown by the history parameter. Cookies thrown by the history parameter after a second attempt at a duplicate vote, were set incorrectly, allowing for a duplicate vote every other time.

Pop-up polls form handling. The revised EmbeddedPoll.html template contains the recommended HTML for displaying results in pop up windows. This entails using method=get, hiding the window location bar. In polls sent by e-mail and with some ill-behaved AOL 4.0 browsers, this method gives much more consistent results than method=post.

Master poll test voting. If you didn't press submit on the Master Poll's Response page, votes would not be tallied until you did. This only affected the Master poll.

Publishing overlap. When two polls were scheduled in a place back to back with no interval in between, occasionally the earlier poll was ended after the new poll was started, resulting in an incorrect new poll. Also, when using the getpoll API, there was a 10 second window during a changeover before the new poll appeared. Similarly, when using the getresults API, it took up to 10 seconds before the old results went away and new results were shown; sometimes this caused results to be shown with no votes but the previous poll's legends.

Previous release notes.

Next: **Introduction**

ViewsFlash 2.8 Release Notes

Release 2.8 is a major release combining features from intermediate releases 2.6 and 2.7. Main enhacements are:

Simplified UI. Pages with many options now offer a toggle for Fewer Choices and More Choices.

Additional Response options. Specify different actions after duplicate voting, missing or invalid data, poll closed. Showing different pages and redirecting the browser are possible.

Question skipping. In a survey, when a visitor clicks on a "no" answer, for example, the voting form scrolls to the indicated next question.

Additional question types. Radio buttons and checkboxes can now be laid out across (horizontally) or down (vertically).

Bulk Voting. When a web site poll is complemented by a live TV or radio show, that invites visitors to vote using an 800 number, this feature adds the votes from the 800 service to the online web totals in one single screen, with an appropriate audit trail. The Web Services API can then construct a TV-ready image to complete the simulcast.

XML. Style Templates can be written in XML. An extension of the Web Services API allows expressing the survey or poll as XML, which can be parsed by other applications.

Ratings Comparisons. The ratings system can now automatically display comparisons between items. For example, five sacred South American Inca sites can be rated by visitors and ranked by beauty, value, and difficulty.

Maintenance and Tuning. Performance logging, tuning, migrating between a development and a production system.

Extensible Validation and Calculated Fields. This extension to the Web Services API allows writing custom Java code to enhance the software to perform custom data validation and calculating derived fields. For example, the provided **Weighted Ranking** class allows a visitor to vote for 3 candidates for 1st, 2nd, or 3rd place, and the combined weighted votes produce a winner's list.

Distributed Architecture.For extremely demanding voting volumes, ViewsFlash now runs in a distributed configuration, with multiple Satellite boxes tallying and responding to votes in parallel, and a single Master box combining the tallies from the satellites for display and storage.

Release 2.5 (January'00) incorporated the main enhancements:

Installation tips. Explicit instructions provided for most popular web servers, including Apache, Netscape, IIS, and the popular servlet runners, JRun and Servlet Exec.

Navigation. Pages now have a "trail crumb" across the top to help to navigate more easily through the application. On each page, you can go up one level by clicking on the appropriate link at the top of the page.

How to make Style Templates. This new documentation section gives tips for commonly requested template modifications.

User Authentication. In response to demand for intranet capabilities, we now support the ability to identify the person who is voting by: a) reading their User ID as provided by the browser and the Web server's Basic HTTP Authentication mechanism. b) reading their User ID as provided by IIS' NT Challenge/Response mechanism (NT Server only) c) extracting the User ID from a cookie, and d) extracting the User ID from a form field, including a hidden field.

Personalized, pre-filled Questionnaires. A new API allows a custom form to be presented to a visitor. For example, a content management system can create a URL that, when visited, makes ViewsFlash dislay a survey with data already filled in. You can use the new question type "Hidden" to thus identify a visitor with their unique customer id, for example, and capture this identity using the Authentication capability, thus tying ViewsFlash into your data collection capabilities very tightly. The Publish page shows how to use this.

Enhancements in questionnaire design. When creating a multiple-question survey, you'll appreciate the ability to move questions around by simply clicking on an up or down arrow, and the ability to insert a new question anywhere among the questions you have already entered. Both a header and footer are provided. HTML is encouraged, to embellish pages. A new question type, Just Text, allows to insert text and HTML anywhere in a questionnaire.

More powerful style templates. Create a polling place, and use the new templates Standard_, ResultsPage, or EmbeddedPoll and EmbeddedResults. When you do, the Poll Page and the response page will ask you questions about font type and size, and other appearance details. Each template you create can contain its own set of customizable options, thus making it very easy to standardize on a look, yet give a known degree of customization to each different styles template. A new option in the Styles page lets you create a new Style Template and copy the contents from an existing template, which can be very useful.

Templates are now stored by default in a folder called viewsflash/styles. When you upgrade, these style templates are added automatically. The Custom 0-9 fields are no longer supported.

Application security. Optionally, you can require that all application pages be addressed by a different TCP port, which you can also configure in your firewalls to limit access to the application.

Problems resolved:

Build 686
When using View Data to retrieve saved data, some unusual character combinations could cause data to be missing from the browser even if it was stored properly on the server. Also, carriage returns in user-submitted forms confused the View Data display. When ViewsFlash was invoked using the HTTP API from WebLogic, an exception was thrown.

Build 660
When Previewing your changes, sometimes it was necessary to use the browser's Refresh command to see the result of your changes; this has been fixed. Also, when logging in to a password-protected bookmarked page in the application, and entering the wrong password, sometimes the application threw a confusing, but harmless, "unexpected error" message before letting you in.

Build 523
Corrected miscounted single checkbox fields in surveys where not all questions were answered. Corrected incorrect preview of response style templates. Corrected ocasional problems with passwords in places.

Build 522
Added additional capabilities for dealing with advanced polls created without the authoring tools. In places whose name begins with "ByHand", polls that are handcrafted can be included in them, after they have submitted at least one vote, and can be managed accordingly. This is useful for migrating polls from release 1.X.

Added documentation to Web Services API on how to force real-time display of poll results when desired.

Build 521
Corrected misbehavior of [/aavg] Style Template tag
Recognizes all file names ending in html as templates, such as shtml, not just .html.

Build 520
Corrects bug where archive poll listings contained current and sometimes even future polls.

Next: **Introduction**

Master/Satellite Architecture

In very high volume situations, such as Internet top-100 sites, it is possible to receive very heavy traffic, on the order of tens of thousands of votes per minute or more. In these situations, ViewsFlash can be configured as a truly distributed, fully scalable system.

Using a load-balancing technique, such as DNS load balancing, submitted votes are routed to a bank of ViewsFlash servers. These servers count votes in parallel and either redirect a visitor's browser to a dynamic publishing system which generates the response page or else compose a response page locally. In both cases, full parallelism is achieved.

In order to provide real-time results, the ViewsFlash servers transmit their results to a Master ViewsFlash server periodically, rather than on every vote. This Master server aggregates the results of the Satellite ViewsFlash servers and maintains the authoritative copy of live results. The Master is also responsible for storing data gathered from the satellites.

If visitors' browsers are redirected to a dynamic page, that page is responsible for querying the Master server periodically, such as once per second, for latest results. If the satellite ViewsFlash boxes are responsible for composing results, they periodically perform this service themselves.

Finally, back-end systems can query the Master ViewsFlash server to display results on live television, for example, using both HTML and XML.

Complete configuration and planning assistance is available from Cogix.