BackOffice ASSOCIATES

DSP™ Application Development Guide

BackOffice Associates, LLC
12000 Findley Road
Suite 200
Johns Creek, GA 30097

T +1 770-246-1287
F +1 770-246-1289
info@boaweb.com
www.boaweb.com

BackOffice ASSOCIATES

Document History

Version	Author	Date	Reviewer	Date	Approver	Date	Comments
1	Nick Fischer	3/9/2016	Matt Wagnon, Tyler Warden, Chris Smith	03/10/16	Mike Avon	3/21/2016	First release

BackOffice ASSOCIATES

Table of Contents

Overview

The Data Stewardship Platform (DSP™) architecture creates powerfully simple, codeless web applications, known as WebApps, with inherent business logic. The platform offers a robust way to dynamically design scalable and secure WebApps that complement or extend business processes.

The DSP™ is the framework upon which all WebApps, including itself, are built. Though the architecture is similar to that of nearly every other web based product, DSP™ pages are never created with interpreted or compiled code from files stored on the web server. The engine directly renders WebApp pages in the client browser based on metadata stored within DSP™. All the information used to create the functionality and controls of an application is stored as dynamically updateable data, not as code, and the content seen in a client web browser is generated on the fly in the browser itself.

There are many ways to build a DSP™ WebApp; however, this manual documents the optimal method by applying Rapid Application Deployment (RAD) for new applications. This method defines the steps necessary to create a new application, which are performed with concise and accurate probability. The following steps apply the RAD philosophy to build DSP™ WebApps quickly:

1. **Create a Data Source –** Each WebApp in DSP™ is based on a database. Refer to *Create Tables in SQL* and *Create Views in SQL* for more information.
2. **Create the WebApp** –Refer to *Create a WebApp in DSP™* for more information.
3. **Register Menus** – *Horizontal* or *Vertical* Menus determine where links are available on the current page. Menus can link to pages or menus in other WebApps within the site. Refer to *Configure the Navigation Pane*, *Submenus and Links to Pages* for more information.
4. **Register Pages and Assign Page Properties** – Performed through page properties, page registration is the foundation of a WebApp and contains most of the primary information that controls the web environment including security and database connection to tables. All pages in the WebApp must be registered to access views. Page properties are then assigned to determine page functionality. Refer to *Design Pages* for more information.
5. **Modify Menus** – Update menus to include new pages.
6. **Assign Column Properties** – Column properties are used to control the behavior of fields and how data displays on a page using control items, such as list boxes, buttons, labels, page links and images. Column properties can be applied to the *Horizontal* View and/or *Vertical* View. Refer to *Assign Column Properties* for more information.
7. **Create and Assign Validation Rules** – Validation rules are defined through database views and applied to a page or a field. Errors or warnings display messages based on the criteria specified in the rule. Refer to *Validation Rules* for more information.
8. **Create and Assign Business Rules** – Corporate business rules can be incorporated into a page or a specific field triggering an action or event based on a business process. Refer to *Business Rules* for more information.
9. **Create and Assign Control Views** – Control views provide a way to define column control based on what is known about the data, page, and/or user. Refer to *Control Views* for more information.

Following these steps when designing a new WebApp provides the necessary details about core functionality as a foundation to deliver a WebApp in the shortest time possible.

This manual also contains information about *Set Security*, *Dashboards and Charts*, *Catalogs*, and other features.

Intended Audience

The manual is intended for both novice and experienced DSP™ designers.

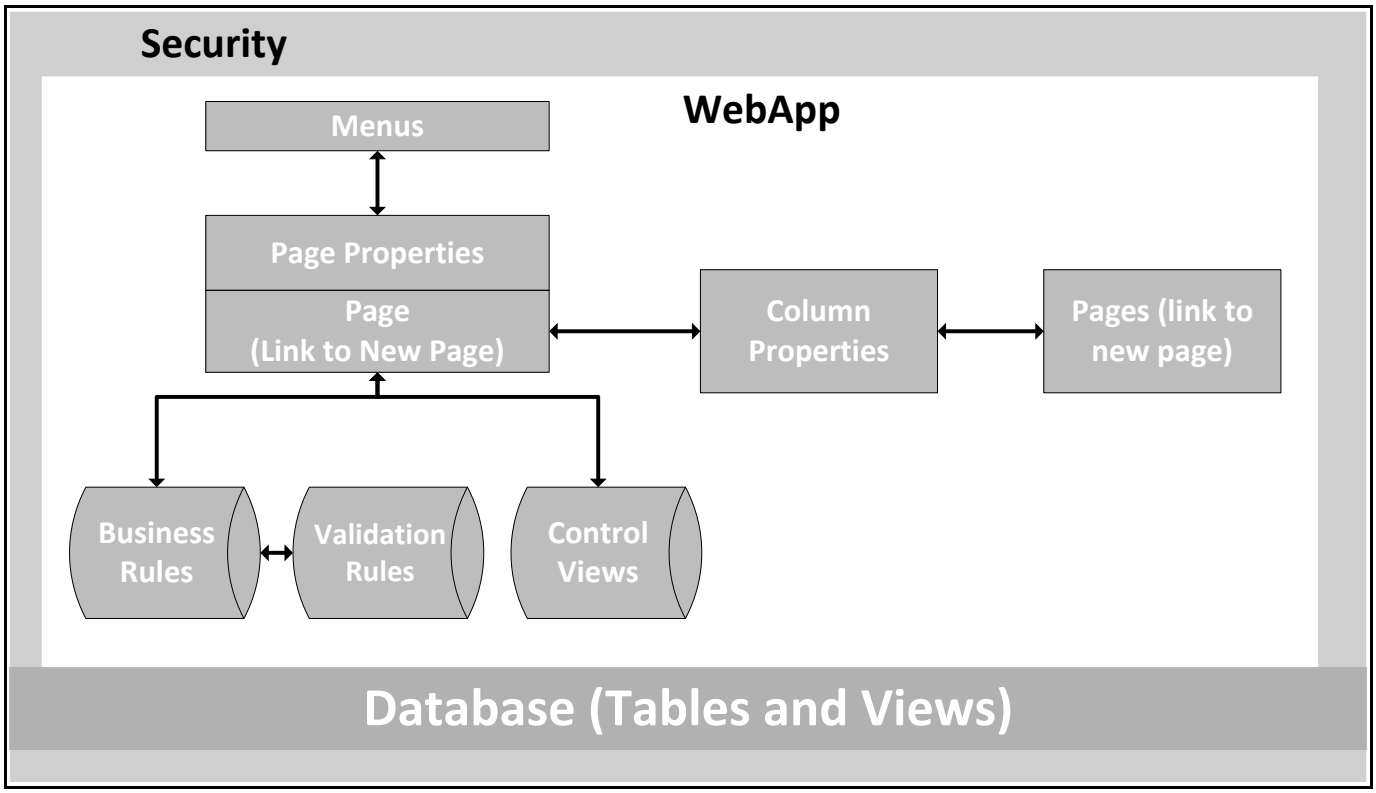
For the novice developer who has never designed a DSP™ WebApp, step-by-step instructions are provided to create and build a complete WebApp. No web programming experience is necessary.

For the individual who is an experienced DSP™ designer but needs help in specific areas, a table of contents provides a cross-reference to specific functions along with documentation and examples.

While general guidelines for working in SQL are included in this manual, step by step instructions for working in SQL are not. It is assumed that the audience has a working knowledge of SQL, including creating tables, and creating and optimizing views and stored procedures. The performance of a WebApp as measured by the responsiveness of the web pages and the overall user experience is directly affected by, and very sensitive to, the performance of the views and stored procedures created for the WebApp by the DSP™ Designer. When building complex WebApps and/or working with large data sets, careful attention should be paid to optimizing the table structures, indices and SQL views.

WebApp Layout

Every WebApp contains pages and every page requires a page definition that represents the core function of the DSP™ operation. Pages are based on views in a database. Each page can contain column properties, events and control views as well as additional features depending on the page type.



Building a WebApp

Database Design

All WebApps are based on a database which is registered as a data source in DSP™. Each page in the WebApp is based on a table and view in the database. Additional tables and views can be created and registered as list boxes, combo boxes, business rules, validation rules, workflows, control views and more.

NOTE: This manual does not provide steps for working in SQL.

Create Tables in SQL

The majority of the pages in a WebApp are Dynamic page types and require an underlying table and a view.

NOTE: This document does not provide steps for working in SQL.

Guidelines for Creating Tables

When creating tables.

- Capitalize field names in such a way that DSP™ correctly displays the label. For example, for the column name **ProductHistory**, DSP™ automatically inserts a space between Product and History.
- Consider a naturally occurring and meaningful primary key.
- Add BOA reserved columns to every table manually, or use the Append Columns feature to automatically include BOA reserved columns to all tables. Refer to *Append BOA Reserved Columns to Tables* for more information, including a list of BOA reserved columns.
- Use default field sizes or common field sizes as much as possible unless the exact size of the field contents is known, e.g., NVARCHAR(50) instead of NVARCHAR(45) or NVARCHAR(255) instead of NVARCHAR(200).
- Normalize data:
  - Avoid tables that have common keys; put all the data in one table.
  - Avoid repeating data groups in tables.

Table Design

Well-designed tables allow for an efficient and easy to develop WebApp. Both naming conventions and standard SQL design practices can contribute to the longevity and extensibility of an application.

Table Naming Conventions

Across the applications, a common naming convention is to name a table with the singular form of the underlying data. For instance, if the table is meant to store a data set representing customers, name the table "Customer." Additionally, prefixes are commonly attached to the table name. "zt" represents application configuration data, whereas "tt" represents expected user data. Configuration data may include user data, but it's still utilized by the application as configuration. For the Customer example, the full table name is "ttCustomer." This allows developers to quickly identify and filter tables to learn and develop the application further.

When naming tables:

- Consistently name tables with small data prefix (2 characters) followed by a descriptive name of the contents of the table.
- Capitalize the first letter of each word, if multiple words are used in the table name, e.g., OrderDetail.
- Do not abbreviate.
- Use singular table names.
- Do not use hyphens ("-"), spaces or slashes ("/", "\") in the table name. Underscores are acceptable.

NOTE: Special characters in SQL Object names will break functionality in DSP™.

The following table outlines the basic table naming conventions:

Condition	Convention	Example
Tables delivered with the application without data. **NOTE**: These tables often store user-entered data.	ttXXX, where tt defines the application and XXX describes the purpose of the table.	nwOrder
Tables delivered with the application with configuration data.	ztXXX, where XXX describes the purpose of the table.	ztParam
Dynamic cross reference tables used as a base and configured onsite.	xtXXX, where XXX describes the purpose of the table.	xtControlStatus
Archive tables	rtXXX, where XXX describes the purpose of the table.	rtRequst

Data Types

Data types are an important characteristic from both a back end and front end perspective. Minimizing data types to their smallest relevant size can improve performance and reduce database bloat. One example of appropriate size limiting is within the System Administration WebApp in the DSP™. Fields that contain SQL View Names (e.g. *Horizontal* View) will only ever contain view names directly from SQL Server. Since SQL Server object names are limited to 128 characters, this field can and should be limited to 128 characters.

Character data types are important for application behavior as well. Most times, nvarchar is preferred over varchar due to the Unicode support of nvarchar. Names and descriptive fields should be nvarchar to ensure they support the widest range of character sets.

Column Collation

Column collation allows for overriding the databases default collation with a custom one, which can impact behaviors like case-sensitive comparisons. This can be considered during application design to determine if a field's case sensitivity is relevant at a business logic level.

Optimization

Indexes and data type minimization can both lead to a high performing table even with millions of records. Everything else in the application will be built on these tables, so if this foundation is rushed or not given enough thought, the result may be excessive technical debt in the future.

Create Views in SQL

All Dynamic pages in DSP™ are based on a view.

NOTE: This document does not provide steps for working in SQL.

SQL View/Function/Procedure Design

Non-schema object design is the next most important thing above table design. When lightweight and efficient, even complex functions, views, and procedures can be implemented with little overhead.

View and Stored Procedure Naming Conventions

Naming conventions are essential as they make development easier, and a standardized set of naming conventions allows other application developers to understand behaviors and configuration.

Views have different sets of naming conventions. The default naming conventions display on the *WebApps* page's *Vertical* View on the Naming Conventions tab. When naming views, follow these conventions.

By default, view names start with web* which allows a developer reviewing/debugging SQL to differentiate between table/view queries. Views must also include the relevant table name after the "web" to denote which base record set the view operates against if the target WebApp has the Enforce Strict Naming option enabled. The view may also contain a brief (one or two word) description of the view's selection criteria. For instance, if a customer is "Inactive," the view may start with "webCustomer_Inactive." Refer to *Naming Conventions and the Enforce Strict Naming Feature* for more information.

Lastly, view names are suffixed with an operation abbreviation for their associated usage. For application development, a list of these can be found on the *Vertical* View of the *WebApps* page (Admin > WebApps > Vertical View > Naming Conventions tab). Other non-enforced naming conventions that are commonly used are "Sel" (to indicate it is **sel**ecting a subset of data) and "Count" (to indicate it is selecting some aggregate or metric data based on the primary table). Underscores can be used to increase readability of the view name. For example, the full named view "webCustomer_NameStartsWithLetterASel" returns a record set of all customers with names starting with "A."

Procedures have similar naming conventions, but suffix with Upd (for update), Del (for delete), and Ins (for insert logic).

Partition Logic Semantically

An application's value is not measured in SQL object count. Centralizing logic can help consolidate an application into smaller, more manageable chunks.

For example, several "Customer" based pages select records from the Customer table. The table has an "Active" bit, and can be modified so that only "Active" customers show up on these pages. Many of these existing "webCustomer*Hor" views may already have complex WHERE clauses. Appending "[Active] = 1" to every WHERE clause adds complexity to the WHERE clause, and makes it less readable and cumbersome to maintain. Instead, create a supplemental active view such as "webCustomer_ActiveSel" and replace all of the "FROM [Customer]" sections with FROM [webCustomer_ActiveSel]. This way, if the definition of "Active" changes, it's an easy change in one view, instead of many.

Procedures are another common place to consolidate logic. SQL stored procedures can be considered object oriented code in the eyes of the developer. Consolidate any logic that is more complex than a single SQL statement into a procedure. This greatly reduces the chance of bugs occurring, and centralizes the logic so that a developer doesn't have to remember where everything is.

For example, place logic such as "When I see an update to set a field to 1, I also now need to go set this other field to 2" into a stored procedure. Updates to this logic can then be added in a single place.

Avoid Global Operators

Any SQL procedures or functions that have to enumerate or parse full sets of table data are dangerous due to their inability to scale efficiently.

Guidelines for Creating Page Views

When creating page views.

- Do not include Order By statements. The **Order By** page property in DSP™ must be used. Refer to *Order By* for more information.
- Name the column names in the table so that the contents of the column are self-explanatory.
- Do not alias column names in views solely for display purposes unless it is necessary to avoid duplicate column names in a view or to avoid adding labels and tabs. Translations in DSP™ can be defined to change a column name. Refer to *Catalogs* for more information.
- Only include the necessary tables.
- For labels and tabs (column properties in DSP™), enter NULL in the column and enter the name of the label or tab in the Alias field. Refer to *Create a Label* and *Add a Tab* for more information.
- Only fields that are present in the page's view and associated table from the page definition are supported for input and edit. However, other tables and/or fields can be joined in the view to provide and display read-only supporting information.
- Functions can be used to display content in a column, but can affect performance. A field that displays data from a function is read-only and cannot be updated.
- Use a view to reference tables in another database. The most common use of this practice is when referencing Collect (dgSAP) tables or DSP™ tables.
- A view can also be used in a page property as a table selection as long as the required key(s) are listed in the view and properly defined in a column property.
- Do not add BOA reserved columns to views (except for the BoaStatus column). Refer to *Append BOA Reserved Columns to Tables* for more information.

Interface Minimization

View Type refers to both the configuration of the page's data views (Horizontal, Vertical, etc.) as well as the Page Column View Type specification. During page design, the Application Designer has to consider fast page load, versus the quantity of relevant data that displays on the page.

Horizontal vs Vertical Views

Every dynamic page in DSP™ must have either a *Horizontal* and/or a *Vertical* View.

The basic principle of page design is to keep the *Horizontal* View simple, and have more lengthy elaborate data on the *Vertical* View. Multi-row rendering of the data is responsive and less overwhelming to the user, while the slower computation / dense data displays on the *Vertical* View, which is loaded on demand.

For example, if a table contains many columns for a single record, to avoid any horizontal scrolling and foster a better user experience, create *Horizontal* and *Vertical* Views. The *Horizontal* View contains the basic information about the columns, while the *Vertical* View contains the columns on the *Horizontal* View and all additional columns.

Horizontal Page Views

When creating a *Horizontal* Page View:

- Make the first column(s) in the views the keys from the table that will be registered to the page to which the *Horizontal* View is assigned.
- Limit the number of fields on the view to avoid horizontal scrolling in DSP™
- Add columns in an order so that the flow of the page processes from left to right.
- Use the naming convention **webXXXHor**, where **XXX** is the name of the table.
- The name of the underlying table, which will be registered to the page, must be included in the name of the view.

Use these controls sparingly on large data sets in *Horizontal* Views:

- List/Combo Boxes (especially with Dynamic WHERE clauses)
- nvarchar(max)
- Other extremely lengthy data type fields

Vertical Page Views

The *Vertical* View should contain the columns on the *Horizontal* View and all additional columns. Additional properties can be applied to organize the *Vertical* View to eliminate vertical scrolling.

When creating a *Vertical* Page View:

- Make the first column(s) in the view the keys from the table that will be registered to the page to which the *Vertical* View is assigned.
- Add columns in an order so that the workflow of the page processes from top to bottom.
- Include all field outputs in the *Horizontal* View assigned to the same page as the *Vertical* View, if applicable.
- Organize data groups into tabs so that each page does not require excessive vertical scrolling or to provide categorical grouping.
- Separate logical groups of data in a tab using labels.
- Use the naming convention **webXXXVer**, where **XXX** is the name of the table.
- The name of the underlying table, which will be registered to the page, must be included in the name of the view.

Improve Page Load Performance Related to List Boxes

List boxes can often be a source of performance degradation due to their elaborate configurations and lengthy lookup times that may have to occur at a cell level. Even if disabled, on the loading of the page, the cell value must be looked up to display to the user. If several list boxes are located on the *Horizontal* View, page load time can be severely

inflated. An easy solution is to have a read-only view-level resolution of the list source (assuming it is a view in the same database) on the *Horizontal* View, and an editable list box on the *Vertical* View. This keeps the description field visible on the *Horizontal* View of the page, making it filterable and searchable, while greatly increasing performance.

Alternatively, the list box on the *Horizontal* View can itself be simplified when it is disabled. There is no need to include a complex WHERE clause that may be present in the list box configuration, if it is only meant to resolve values and not restrict user input or security. Even this simple change can improve page load.

Create a WebApp in DSP™

Once the database is created, the WebApp can be created in the DSP™. The database must be registered and the WebApp created and added to the *Navigation* pane.

Register the Database

The database must be registered as a data source in DSP™ to create a WebApp. In most cases, the System Administrator registers the database.

After a user logs in to the DSP™, the *Site* page displays. The *Navigation* pane on the left provides access to all WebApps in the user security profile. Access System Administration by clicking **Admin** on the *Navigation* pane. To return to the *Site* page or main menu, click the BackOffice Associates® logo on the Site toolbar.

The database contains the tables and views needed to design the WebApp. Additional views are automatically added from the DSP™ after the data source is added and the WebApp is created and validated. These views pull in BOA views from the platform database. This is beneficial for development when a user may not have security to the DSP™ database or if the WebApp is on a different server than the DSP™ database.

To register the database:

1. Log in to the DSP™.
2. Click **Admin** > **Data Sources** in the *Navigation* pane.
3. Click **Add**.
4. Enter a unique name in the **DATA SOURCE NAME** field.

NOTE: The Data Source Name is a logical name and may contain any alpha/numeric character, including spaces.

5. Click **Save**, the *Vertical* View displays.
6. Enter the server address where the data source is stored in the **Server Address** combo box.
7. Enter the database name in the **Database** field.

NOTE: The database name must match the actual name of the database located on the SQL Server.

8. Enter the user account to access the database in the **User ID** field.
9. Enter the password to access the database in **Password** field.

NOTE: The Security or System Administrator may have to establish the **User ID** and **Password** to the database.

10. Click **Save**, a message displays.

NOTE: An informational message displays indicating some views need to interact with other databases, such as DSP™, on the server and need to be enabled. This enables views necessary to build the WebApp to populate the data source.

11. Click **OK.**
12. Click **Advanced Properties** tab.
13. Click **System Views** check box to enable it.

NOTE: Having **System Views** checked will automatically install all BOA views into the data source. They are useful for applications that need to select data from the DSP™ database.

14. Click **Test Connection** on the Page toolbar; a confirmation message displays.

NOTE: If the connection is not made, verify the database name and login information are correct and try again.

15. Click **Append Columns** on the Page toolbar, a confirmation message displays.

NOTE: Refer to *Append BOA Reserved Columns to Tables* for more information.

16. Click **OK**.

NOTE: Use the **Recompile Objects** icon on the Page toolbar to recompile every object in a database to ensure they are still valid, if needed. A list of object names is returned if they fail to compile. The failed objects need to be manually fixed.

Append BOA Reserved Columns to Tables

BOA reserved columns are:

- boaStatus
- AddedOn
- AddedVia
- ChangedBy
- ChangedOn
- ChangedVia
- Locked On
- Locked By

The first six columns above must be added to every table.

When the user clicks the Append Columns icon on the Page toolbar of the *Data Sources* page's *Vertical* View, most of these reserved columns are appended to all of the tables that do not already have them in the database. A Designer can also add these columns manually.

NOTE: **Locked On** and **Locked By** should be added manually where appropriate. Add these columns for any table where multiple users may be editing the same record at the same time.

With the exception of boaStatus, the appended column should not display on the page so should not be added to the view.

NOTE: If the columns exist on the table, they display when a user hovers the mouse cursor over the row edit pencil icon. This allows the developer to exclude the columns from the view, but still take advantage of them.

NOTE: Reserved columns are only automatically added to tables when the Append Columns icon is clicked on the *Data Sources Vertical* View. Columns are not automatically appended to tables added after this operation is performed. In other words, if tables are added to the data source later, click the Append Columns icon again to add the reserved columns to the new tables.

Create a WebApp

Create a WebApp and associate it with a registered data source.

To create a WebApp:

1. Click **Admin** > **WebApps** in the *Navigation* pane.
2. Click **Add**.
3. Enter a name in the **WEB APP NAME** field.

NOTE: Make sure it is typed correctly; changing the WebApp name later is **not** recommended.

4. Select the data source from **DATA SOURCE ID** list box.

NOTE: The **Data Source ID** is the logical name of the SQL Server Database.

5. Click **Save**, the *Vertical* View displays.
6. Enter a number in **Version** field.
7. Select a date in **Release Date** field.
8. Click **Save**.
9. Click **Naming Options** tab.
10. Verify **Enforce Strict Naming** is checked.

NOTE: Refer to *Naming Conventions and the Enforce Strict Naming Feature* for more information.

11. Verify the value in the **Table Prefix Size** field is **2**.

NOTE: The **Table Prefix Size** indicates the first two characters (prefix) are for registration purposes and are not included as part of the table name for consideration with Enforce Strict Naming.

12. Click **Save**.

Naming Conventions and the Enforce Strict Naming Feature

The DSP™ is delivered with a BOA pre-defined naming convention for Horizontal, Vertical, Stored Procedures, Validation and Control Views to name a few. The defined naming conventions on the **Naming Conventions** tab on the *Vertical* View of a WebApp simplifies list box options in DSP™ by limiting the objects associated with any given page by enforcing a simple naming convention.

Views in the underlying database must be named with the associated table name from the page registration. If **Enforce Strict Naming** is checked, it forces DSP™ to filter applicable list boxes based on the naming conventions. For example, a WebApp has a page using the **nwEmployee** table. With Enforce Strict Naming enabled, all views registered to this page **must** contain the word Employee within the name of the view and with the proper naming convention prefix and suffix. For a Horizontal page, *web* is the prefix and *Hor* is the suffix. For a Vertical page, *web* is the prefix and *Ver* is the suffix.

Correct	Incorrect (Will not display)
webEmployeeHor	Employee
webEmployeeHor	EmployeeHor
webEmployeeVer	webWorkerVer
webEmployeeTypeHor	webTypeHor

The default **Table Prefix Size** is **2** for the WebApp and indicates the first two characters (prefix) are for registration purposes and are not included as part of the table name for consideration with Enforce Strict Naming. For example, the tables in **Northwind** are prefixed with **nw**, **xt** or **zt**. The **nw** is ignored when filtering any views based for a page where the **nwEmployee** table is registered.

If, when selecting a view from a list box, the needed view is not available, check the Naming Conventions tab for the active WebApp and compare it to the table and view names for that page.

Configure the Navigation Pane, Submenus and Links to Pages

SQL table data is presented through standard views when registered in the DSP™ server throughout the web interface. Create pages and additional menus by registering tables and views from the underlying database to link them together.

Once built, the WebApp layout contains menus that link to pages. Some menus are automatically created and added to the *Navigation* pane when the WebApp is built. Additional pages and menus are manually added to the *Navigation* pane. WebApps commonly contain a *Configuration* Menu.

Guidelines for the Navigation Pane

The *Navigation* pane on the left displays page links to the WebApp.

The following are the suggested menu links on the *Navigation* pane for each WebApp (in this order, from top to bottom):

- **WebApp Name** – Displays the name of the current WebApp with the proper trademark. This menu item links to the WebApp home page or message board.
- **Configuration** – A submenu often created to link to the multiple pages available under Configuration. Each page under Configuration should display the *Navigation* pane at the left, with links to all configuration tables specific to the WebApp. It is suggested that every WebApp contain a link to the *Parameters* page that contains, at a minimum, the following fields: **Application Name**, **Version** and **Release Date**.

NOTE: The *Configuration* submenu often provides links to other pages that can be used for list boxes on other pages within the WebApp.

The following are general menu guidelines:

- Use a standard *Navigation* pane on all pages. The menu items on the *Navigation* pane vary depending on the WebApp.
- Do not modify the *Navigation* pane labels that are automatically added by DSP™ when a WebApp is created. If copying an existing WebApp to create a new WebApp, the *Navigation* pane links must be modified to link to the appropriate pages within the new WebApp. Refer to *Copy Page* for more information.
- Do not use labels for menus unless the name of the page needs to be different only when displayed on the menu.

NOTE: When a user navigates from one WebApp to another, the menu displayed in the navigation pane is chosen from the target page the user is loading. Most WebApps are accessed via the *Navigation* pane with a link to the WebApp's home page. The Horizontal Menu of that target page is chosen, unless it is null where it will fall back to the Vertical Menu. This menu will be the menu used for the entirety of the time the user remains in this WebApp. The most common usage is to define the main pages on the *Switchboard* menu and use that on most pages as the Horizontal View. Submenus can be created and then registered on the *Switchboard* menu to provide access to other top-level pages.

Add a Menu to the Navigation Pane

By default, a new WebApp has one menu created called *Switchboard*. Menus can be registered within menus to allow sub menus to drill through to other options inside the *Navigation* pane.

To create a menu:

1. Click **Admin** > **WebApps** on the *Navigation* pane.
2. Click **Menus** for a WebApp.
3. Click **Add**.
4. Enter a name in **MENU NAME** field.
5. Click **Save**.
6. Click **Menu Links**.

NOTE: When a GUID displays in the **Link to Page ID** column, it indicates the page is an external WebApp. Do **not** alter or remove this menu link; it could render the WebApp not editable.

7. Click **Add**.
8. Enter a number in the **PRIORITY** field.

NOTE: The priority determines the order of the page links from top to bottom. Priority automatically sorts in an ascending order. The higher the priority, the lower an item displays in the *Navigation* pane.

9. Select an image name in the IMAGE ID list box (optional).

NOTE: This image displays to the left of the menu item in the *Navigation* pane.

10. Enter the name that displays on the *Navigation* pane in **Label** field, if necessary. Leaving this field blank will display the name of the target page in the **Link to Page ID** field, and this will be translated automatically if enabled.
11. Select an option from **SUBMENU ID** list box.

NOTE: The **Submenu ID** determines the options on the main menu. For example, all menu links defined in Configuration display as a submenu under *Configuration* on the *Navigation* pane.

12. Click **Save**.

Design Pages

Guidelines for naming pages

- Use parentheses rather than hyphens to delineate the description of the page, e.g., Request (Role) rather than Request – Role.
- Pluralizing page names is acceptable.

Page Types

Each page type displays information to an end user in a specific format. The DSP™ uses the following page types:

- **Chart** – Requires a view. Used to display a graph with an X/Y-axis. Refer to *Chart Types* for more information.
- **Dynamic** – Requires a view and an underlying table. Renders on the fly using the configured settings in the administrative interface. This is the most commonly used page type. Refer to *Dynamic Views for Multiple Control Fields* for more information.
  - Displays appropriate data using the *Horizontal* and *Vertical* Views in the database.
  - Enables proper controls for adding, editing, and deleting records.
  - Sorts the page data as configured.
  - Determine ability to add, edit, delete or sort records on a page.
  - Displays permitted links and appropriate attributes per column as designed.
- **Layout –** Displays two linked pages to display parent and child (dependent) pane. Refer to *Create Header Detail Pages* for more information. A layout:
  - Does not require a table.
  - Can be added to the *Navigation* pane.
  - Requires two pages with views and tables registered to the WebApp but not registered to the *Navigation* pane.
  - Requires that the parent page links to the dependent pages via the Linked to Page ID column property.
  - Requires that the parent page is assigned to the parent frame for the layout page.
  - Requires the Linked To Page ID column is selected to display the child pane.
- **Report** – Requires a view only. Compiles the records on the page and displays them as a single report under the *Report* menu, located on the *Navigation* pane. Refer to *Reports* for more information.
- **Service** – Requires a view, but is not displayed in the DSP™. The OnValidate event for a Service page runs for every record that appears in its view on a regularly scheduled basis based on the page properties. A Service page can be as simple as a view. The view could point to all records in a table where an OnValidate event sends an email for each record in the table.
- **Static** – Links to traditionally created web pages stored and registered on the DSP™ server.

NOTE: Standard WebApp guidelines dictate all WebApps must have a Parameters page that contains, at a minimum, the following fields: **Application Name**, **Version** and **Release Date**. Link the *Parameters* page to the Configuration Menu once the WebApp is created.

Guidelines for Developing Pages for a Quality WebApp

Follow these steps to design a fully implemented page with all available and relevant features to ensure a high-quality user experience for both business users and site administrators.

- If applicable, determine an Update and Insert method relevant to the page's underlying data set. For simple record sets, solely Horizontal Updates/Inserts may be sufficient. If the record requires several or lengthy fields to be filled out, a Vertical is a good page supplement. If the fields to be filled out are dependent on a choice made on the horizontal, a dynamic substituted vertical view is a good option. Refer to a *Dynamic Views for Multiple Control Fields* for more information.
- Add in all Page Column Properties to ensure the page flow is user friendly. Verticals should make use of tabs and labels to avoid the use of vertical scrolling. Horizontal views should utilize Rowspan and Colspan to ensure the record length doesn't exceed a reasonable resolution. Prioritize important text-based contextual information on the left, and actions on the right. Refer to *Assign Column Properties* for more information.
- Determine which column(s) comprise a logical representation of the record set. Create or modify the Page Column to check "Show In Navigation", which will cause a secondary text color to be applied to them, visually indicating to the user that these columns are a representation of the record's identity. This will also cause them to show up concatenated within the Navigation pages. Refer to *Enable Show in Navigation* for more information.
- Review existing WebApp Security Groups. Determine which groups apply to the page and apply them. Ensure that the relevant groups will have a valid navigation path to the page. Refer to *WebApp Security Group Design* for more information.
- Create any control status modifiers (DCV, PCV, UCV, Control Status Field) adjustments required to ensure that only the relevant fields are editable in the right contexts.
- Identify any columns whose meaning cannot be easily inferred by context. A bit field like "Enable Advanced Tracing" should contain Column Help to indicate to the end user what this field does. Refer to *Add Column Help Text* for more information.
- Identify which columns may benefit from a watermark, such as list boxes that search for particularly named values/objects. Watermarks can display the expected formats, to ensure that the user understands what's expected. Also, suggested text formats can be useful. If the user is expected to use a namespace convention, indicating it during the add/edit is more useful than providing a warning after the save. Refer to *Add a Watermark* for more information.
- Identify which columns may benefit from a tooltip, such as on images that represent a status. The status can be elaborated on when the status icon is hovered over. For instance, a red light in a column called "Status" could have tooltip that may say "X templates failed to process". A tooltip can also be useful when a field is disabled due to lack of prerequisites. If a "Submit" button is disabled because the record is "Inactive", then the tooltip can state "Disabled due to being inactive". Refer to *Add a Tooltip* for more information.
- Identify which columns may benefit from implementing a Default View. If there's any input that can be inferred based on drill down criteria or other environment information, it should be included in a default view. Input formats should be conveyed via Watermark, whereas defaults or suggested values should be conveyed via Defaults.
- Identify which columns may contain data that should be translated. Because of potential for translation overhead as well as the Pareto principle, cell (and other) translations are opt-in. For example, if a list box within the application has the options "Active" and "Inactive," any application string literals or user input that may be used as configuration should be translated. However, do NOT translate everything by default. This will result in a slower page load and a more encumbered cache. Refer to *Catalogs* for more information.

Consider which supplemental page features are relevant for the page.

- Support Download –A page, such as a metric-related page, may be useful to download, while a page containing sensitive information may not.
  - If Download is supported, are there fields, such as mages and buttons, that shouldn't be included in the downloaded file? These fields can be excluded i.e., hidden via View Type on the *Page Columns* page. Refer to *Column Control Status* for more information.
  - Alternatively, are there fields that should be included on the Download page that are not available on the horizontal or vertical? Hide fields using the page columns All Views view type, but then enable them on the Download for this level of control. Refer to *File Upload/Download* for more information.
- Support Report – Is this a page that can provide additional information about this record / and or its children? A page with the Support Report feature enabled shouldn't solely contain the information that could be obtained via a download.

If the page supports reporting:

- Should an alternative view be used for the report? If the columns on the *Horizontal* View differ greatly from the columns on the report, create a separate view to avoid bloat to the standard *Horizontal* View.
- As with Download, which columns should be shown, which can be ignored? Which should be added for reporting purposes?
- Should Report Groups be used to include child records per reported item? If the page already has a link to a subpage that should be included, the link can be configured to "Report Follows Link" to append children data to this page's report. Refer to *Report Follows Link* for more information.

- Does Search (System Admin feature) need to be enabled?
- Full Page Update – Some pages have interdependent rows and should not allow editing.
- Persistent Insert – When the user navigates to this page, is it likely they will add several records at once? If so, enable this option. Refer to *Enable Persistent Insert* for more information.
- Support Direct Update – Direct update allows a user to enable or disable a check box without having to edit the page. Enable this setting if there is a check box(es) on the page, and updating the check box(es) is commonly the only change that a user makes on the page. Disable this setting if it is expected that the user will update additional information on the page.
- Allow Explicit Quick Link – If this page requires binding to function correctly (*Page Control View* for example) then this should be unchecked. If not, it can be enabled.
- Should the page support Excel upload? If so, what columns should be available to upload?
- Are there any List Filter-enabled combo or list boxes on the page? Are there fields to add to supplement the List Filter feature? For example, layout pages within System Admin generate a Template Preview exclusively for the "Filter (Control)" view type so that there's a visual example of what to expect when using that particular value. Refer to *List Boxes, Combo Boxes and Filter Features* for more information.
- Are there fields that should not be filterable? Perhaps there are columns to filter against that do not exist in the page's view. For example, the Messages page contains a "Date Received" datetime property on the Filter panel, which allows filtering on a range of Date Time values, even though this field is not present on the *Horizontal* View. Control which fields can be used as a filter on the *Page Columns* page's *Vertical* View for a column with the View Type of Filter (Form). Refer to *List Boxes, Combo Boxes and Filter Features* for more information.
- Does the page contain referential data to another application that can be implicitly secured? For example, if the application is meant to supplement functionality based on a Wave, bind the Wave security definition shipped with

Console to the page. When a user is granted access to a particular Wave, the user can view and edit data in the WebApp that relates to that Wave.

- Does the page need Dynamic Subtitle to explain the core concept of what the user will be doing on this page? Applications like Automate use Page Subtitle on their landing page to give a brief explanation of the application's usage. Dynamic Subtitle is defined on the *Catalog Page Help* page's *Vertical* View (Admin > Catalogs > WebApps > Pages > Page Help),
- Would the page benefit from a Hover View? When a record is selected on the page, if the user hovers the cursor over the page title, after a brief delay, the record's keys display by default in the Information pane (below the *Navigation* pane). These values are usually GUIDs, which are meaningless to the end user. When debugging, the Metrics panel works just as well as title hover for obtaining table key values (the Metrics panel can be accessed from the Change Settings icon in the Site toolbar when in debug mode). Instead, the Hover View can be populated with additional technical information about the selected record.
- Finally, test for page performance under load while in Debug mode. Review the Page Trace (found on the Pages page Vertical view under the Tracing tab) to determine which portions of the page run slowly when the record set (or user set) increases dramatically. The page designer must know the anticipated largest reasonable record set. Some pages can be complex, as they never expect to have more than a handful of records, whereas others may have tens of thousands.

NOTE: Being functionally complete does not mean code complete. It's possible the page behaves correctly until it is scaled. The designer may implement the page in a more efficient way. This happens often in application design, regardless of the development medium.

Register Pages and Assign Properties

All pages in the WebApp, with the exception of Reports, Charts and Header Detail page types, must be registered by selecting tables and views already created in the database (Reports, Charts and Header Detail pages need views but do not need a table to be selected). Page properties are assigned to determine page functionality.

There are two ways to register pages and assign properties in DSP™:

- Select Admin > WebApps > Pages, or
- Click the WebApp in the *Navigation* pane, click the **Design Page** icon on the Site toolbar and select **AddPage**.

Basic Page Properties

The following are basic page properties which determine the data and functionality on the page.

- **Table** – Determines where the data for the page is stored.
- **Horizontal View** – Determines what fields from the table are to display on the page and field order.
- **Vertical View** – Determines if there is a Vertical View for the table displaying additional information.
- **Horizontal Menu ID** – Determines what menu displays in the *Navigation* pane. See *Configure the Navigation Pane, Sub Menus and Links to Pages* for more information.
- **Vertical Menu ID** – Determines what menu, if any, displays at the left of the page.
- **Insert Method** – Determines the behavior when adding a new record. Options are:
  - **Not Supported** – No records can be added to the page. The **Add** icon on the Page toolbar does not display.
  - **Horizontal Insert** – Once a record is added and saved the current *Horizontal* View persists.

    - **Horizontal Insert/Switch to Vertical** – Once the record is added and saved, the *Vertical* View displays. The *Vertical* View must be saved to save any information on the *Vertical* View only. If the *Vertical* View is not saved, the data entered on the *Horizontal* View persists and still displays.
    - **Horizontal/Switch to Vertical** – Once the record is added and saved, the *Vertical* View displays. The *Vertical* View must be saved to save any information on the *Horizontal* **and** *Vertical* Views. If the *Vertical* View is not saved, the data entered on the *Horizontal* View is not saved.
    - **Switch to Vertical Immediately** – Upon adding a record, the *Vertical* View displays.
- **Update Method** – Determines the behavior when editing a record. Options are:
    - **Not Supported** – No records can be edited. The **Edit** icon on the Page Toolbar does not display.
    - **Horizontal Update** – Once a record is edited and saved the current *Horizontal* View persists.
    - **Horizontal Update/Switch to Vertical** – Once the record is edited and saved, the *Vertical* View displays. The *Vertical* View must be saved to save any information on the *Vertical* View only. If the *Vertical* View is not saved, the data edited on the *Horizontal* View persists and still displays.
    - **Horizontal/Switch to Vertical** – Once the record is edited and saved, the *Vertical* View displays. The *Vertical* View must be saved to save any information on the *Horizontal* **and** *Vertical* Views. If the *Vertical* view is not saved, the data edited on the *Horizontal* View is not saved.
    - **Switch to Vertical Immediately** – Upon editing a record, the *Vertical* View displays.
- **Support Delete** – Determines if records on the page can be deleted. When unchecked, the **Delete** icon does not display.

Register all pages and assign properties for the WebApp. Once the pages are registered, they are then linked to the *Navigation* pane and *Configuration* Menus.

NOTE: The steps below describe some basic setup options for a Dynamic page. Other page types are described in *Header Detail Pages* and *Dashboards and Charts*.

Register a Dynamic Page

The most commonly used page type, a Dynamic Page requires a view and an underlying table. It renders on the fly using the configured settings in the administrative interface.

To register a Dynamic page:

1. Click the WebApp name on the *Navigation* pane.
2. Click the **Design Page** icon on the Site toolbar.
3. Select **AddPage**, the *Pages* page displays in a new window.
4. Enter a page description in **DESCRIPTION** field.
5. Verify **Dynamic** is selected in **PAGE TYPE** list box.
6. Select the page's underlying table from the **TABLE** list box.
7. Click **Save**, the *Vertical* View displays.
8. Select the view for the page created in SQL server from **Horizontal View** or **Vertical View** list box.

NOTE: All dynamic pages must have a table and at least one view. These list boxes are filtered based on the settings on the Naming Conventions tab on the *Vertical* View of the WebApp. Refer to *Naming Conventions and the Enforce Strict Naming Feature* for more information.

9. Select an option in the **Insert Method** list box.

10. Select an option from **Update Method** list box.
11. Click **Support Delete** check box to disable it, if needed.
12. Click **Save**.

Add Pages to the *Navigation* Pane

Provide access to WebApp pages via the *Navigation* pane by linking to them.

NOTE: Pages can also be registered to the *Configuration* submenu, added as layout pages, or accessed by clicking images or buttons. Refer to *Configure the Navigation Pane, Submenus and Links to Pages* and *Link to a Page Using Buttons and Images* for more information.

To add menu links to the *Navigation* pane:

1. Click **Admin** > **WebApps** on the *Navigation* pane.
2. Click the **Menus** icon for a WebApp.
3. Click the **Menu Links** icon for a menu name.
4. Click **Add**.
5. Enter a number in **PRIORITY** field.

NOTE: Priority determines the order the menu links display on the *Navigation* pane, by Priority value ascending.

6. Select the page name from **LINK TO PAGE ID** list box.
7. Click **Save**.

Create Header Detail Pages

Header Detail pages are a Page Type that displays two linked pages in parent and dependent panes.

Guidelines for Header Detail Pages

- A table is not required.
- The two pages with views and tables (i.e., the parent and child page) must be registered to the WebApp but not registered to the Navigation pane.
- The two pages must be linked with the Link to Page ID column property.
- The parent page must be assigned to the Parent frame for the Header Detail page.
- The Link To Page ID column must be selected to display in the Dependent pane.
- If multiple columns are linked it will link to the first column by default.
- The Header Detail page is added to the Navigation pane.

Create a Header Detail Page

Create Header Detail pages to display the parent page in the top frame and the child page in the bottom frame.

To create a Header Detail page:

1. Click **Admin** > **WebApps** on the *Navigation* pane.
2. Click the **Pages** icon for a WebApp.
3. Click **Add**.

4. Enter a description of the header page in **DESCRIPTION** field.
5. Select **Header Detail** from **PAGE TYPE** list box.

NOTE: A table is not necessary for Header Detail Page Types.

6. Click **Save**, the *Vertical* View displays.
7. Click **Save**.
8. Click the **Frames** icon.
9. Highlight the row with Priority 10.
10. Click the **Edit** button.
11. Enter a header page description in **DESCRIPTION** field.
12. Select the header page ID from the **DEFAULT PAGE ID** list box.

NOTE: The **Default Page ID** is the parent page.

13. Click **Save**.
14. Repeat steps 10-13 for the detail page with the priority of 20.
15. Link the pages. Refer to *Link to a Page Using Buttons and Images* for more information.

Assign Column Properties

Column Properties are used to control the behavior of fields on a page. Control items, such as List Boxes, Buttons, Labels, Page Links and Images, define how columns and fields work with selected properties. Column Properties can be applied to the *Horizontal* View, *Vertical* View, and more.

There are three ways to add column properties to a WebApp:

- Log in to System Administration. Navigate to WebApps, and click **Pages** for a WebApp. Locate the page and click **Column Properties**.
- Navigate to the page in the WebApp where the column property is to be applied. Click **Design Page** on the Site Toolbar, then click **Column Properties**.
- Navigate to the page in the WebApp where the column property is to be applied, right-click the column heading, and click Add or Edit (Control Name) if the column property already exists.

Link to a Page Using Buttons and Images

To complete the overall navigation of the WebApp, buttons and images can be added to columns and the Page toolbar to use as page links. Use an existing column in a view, or a count, or a NULL and assign an alias so a Button, Image or Toolbar column property can be added to the page in the DSP™. Buttons can display text or a counter to indicate the number of records on the linked page. Links also control the display for a Layout page type which displays the parent data in the parent pane and the dependent data in the child pane.

NOTE: It is recommended to implement the page links in column properties prior to building additional page properties. Testing additional properties on pages that have not been linked within the WebApp cannot be performed during implementation, leaving possible errors undiscovered until later.

NOTE: A *NULL* column with an alias must be added to the underlying view to add a button or an image to a column that does not contain data. The view must be modified before a link from a button can be added.

NOTE: This manual does not provide steps for using SQL.

Assign an Image Link

To assign a link from a button on a header page to a detail page:

1. Navigate to the header detail page in the WebApp.
2. Right click the column heading to add the link.
3. Select Add > **Image**.
4. Select the image name from the **Image ID** list box.
5. Select the detail page name from **Link to Page ID** list box.
6. Click **Save**.

Assign a Toolbar Button Link

Toolbar buttons can be added to the Page Toolbars. The buttons can be used for page navigation, to link to additional pages or to run a rule.

NOTE: A *NULL* column with an alias must be added to the underlying view to add a Toolbar button to a column that does not contain data. The view must be modified in SQL before the Toolbar button can be added. This manual does not provide steps for using SQL.

To create a Toolbar button:

1. Access the page in the WebApp.
2. Right-click the column heading on the page that should be a Toolbar button.
3. Select **Add > Toolbar**.
4. Verify **Page** is selected from **Move to Toolbar** list box.
5. Select the button image from **Image ID** list box.
6. Select the page to open when the toolbar button is clicked from **Link to Page ID** list box.
7. Click **Save**.

Column Control Status

Control Status is a Column Property option that determines the visibility of a column on a page. Options are:

- **Enabled** – Displays the column on the page. This is the default value.
- **Hide** – Hides the column on the page.
- **Disabled** – Protects the column. Displays as read-only columns when in edit mode.

To alter a column's Control Status:

1. Access the page in the WebApp.
2. Right-click the column heading for the column to change.
3. Select **Add > Text Box**, or **Edit > Text Box**, if there is a text box control already.
4. Select **Hide, Disabled, or Enabled** from **Control Status** list box.
5. Click **Save**.

Display a Column in Numeric Format

Fields can be configured to display in numeric formats such as Currency, Numeric, Numeric (Positive), Percent or Zero-Padded Integer.

To apply a numeric format to a column:

1. Access the page in the WebApp.
2. Right-click column heading
3. Select **Add >Numeric**.
4. Select an option from **Format** list box.

NOTE: Depending on the Format selected and the purpose of the field, set these additional options.

5. Enter a value in the **Numeric Scale** text box.

NOTE: The Numeric Scale field determines the number of decimal places to display after the decimal separator.

6. Click **Display Thousands Separator** check box.

NOTE: The Display Thousands Separator will force a number to use the current user's locale to display a number such as 4213.22 as 4,213.22 given that English – United States is selected in the Currency Locale ID list box.

7. Click **Total** check box.

NOTE: A totals column recalculates when records are added, deleted, or modified.

8. Click **Save**.

Set Default Field Value for a Column

Default Field Value is a Column Property that populates a field with a pre-determined value.

To default a value in a column:

1. Access the page in the WebApp.
2. Right-click the column heading.
3. Select **Add > Text Box**.
4. Enter the default value for the field in **Default** field.
5. Click **Save**.

Add an HTML Area to a Field

HTML Area is a Control Type that provides a tool to format text within the field using bold, italics, colors or additional fonts. More complex features, such as lists, tables, URL's and images are also available without programming or the need to understand HTML code.

To add an HTML Area control to a field:

1. Access the page in the WebApp.
2. Right-click the field name.
3. Select **Add > HTML Area**.
4. Click **Save**.

Enable Dynamic Image

Dynamic Image is a Column Property that displays an image uploaded to a record instead of file path. The value returned by the view in this column can be:

- A file path to an image on the web server
- A GUID value representing the ImageID in the System Administration > Images
- A unique image name, referenced from the Image Name in System Administration > Images

To enable Dynamic Image for a field:

1. Access the page in the WebApp.
2. Right-click the field name.
3. Select **Add > Image**.
4. Click **Advanced Properties** tab.
5. Click **Dynamic Image** check box.
6. Click **Save**.

Create a Label

Label is a Control Type that displays a label to distinguish between *Vertical* View sections. The label needs to be added to the SQL View with NULL in the **Column** column and the label title in the **Alias** Column before the label can be created. This manual does not include steps for using SQL.

NOTE: DSP™ adds spaces to words based on the placement of capital letters in field names. Therefore, all multiple-word field names should be concatenated with capital letters that serve as the demarcation point between the words. The column then displays with the appropriate spaces on the page.

To create a label:

1. Access the page in the WebApp.
2. Right-click the field name on the vertical view that should be a Label.
3. Select **Add > Label**.
4. Click **Save**.

Add a Tab

Tab is a Control Type that displays data on different tabs accessible at the top of a *Vertical* View. Labels can display on Tabs to further organize the data. Tabs are created like labels by entering NULL in the column in SQL and alias as the Tab name. The view must be updated before the tab can be added in the DSP™.

NOTE: This document does not provide steps for working in SQL.

To create a tab:

1. Access the page in the WebApp.
2. Right-click the field name on the vertical view that should be a Tab.
3. Select **Add > Tab**.
4. Click **Save**.

Enable Show in Navigation

Show in Navigation sets the values that display when a page title is expanded in the *Navigation* pane. By default it displays the Primary Key from the underlying table.

To show values in the *Navigation* pane:

1. Access the page in the WebApp.
2. Right click the column heading.
3. Select **Add > Text Box**.

NOTE: The Control Type must be **Text Box** to display a field in the *Navigation* pane.

4. Click **Advanced Properties** tab.
5. Click **Show in Navigation** check box to enable.
6. Click **Save**.

Set Spanning Properties

Spanning Properties is a method to reduce horizontal scrolling and present data efficiently. In some cases there may not be enough fields or differences to warrant creating a *Vertical* View. The Spanning Properties Column Properties can create a line break on the Column headings to shift some fields down making the record two lines. For example, if the record has 10 columns, the Spanning Properties Line Break can be set on the 6th column. Columns 1-5 will display on the top line and columns 6-10 will display beneath columns 1 -5. Prioritize important text-based contextual information on the left, and actions on the right.

In some cases the width of the column might need to be adjusted to avoid character wrapping. The Column Span feature determines the width of the column. For example if the Column Span is set to 2, it will span 2 columns.

If the Description is a text area and contains a sentence, it would not make sense to make it wider, it would be better if it was taller. The Row Span feature determines the height of the row. For example, if the Row Span is set to 2, the Row will be 2 rows high.

Column Span can be used in any case; however Row Span should only be used when a Line Break is set.

To set the Spanning Properties:

1. Access the page in the WebApp.
2. Right-click the column heading.
3. Select **Add > Text Box**.
4. Click **Spanning Properties** tab.
5. Click **Line Break** check box to enable.

NOTE: Any columns to the right of this column display on the next line.

6. Click **Save**.

List Boxes and Combo Boxes

List Box and Combo Box are Control Types that display a field as a list box or text box with options from a selected Table or View. Column Properties are specified to control the behavior of the Control Types, what is displayed: and how values are stored in a table.

NOTE: This document does not provide steps for working in SQL.

When creating List Views:

- Use the naming convention **webXXXList** where **XXX** describes what is being listed, e.g., webProductList.
- Include only the columns that apply to the List Value, List Display, List Select or List WHERE clause registrations.
- Include a unique set of data.
- Do not sort in the view. Sorting is done in DSP™ using the List Order By Column Property option. Sorting is not officially supported in later versions of Microsoft SQL Server.

General Combo Box guidelines:

- Only values in the list view can be entered in the field.
- Use a combo box instead of a list box when the list view contains too many records to display suitably as a list box.
- Enable List Filter to allow filter on a combo box.
- Include both a text field and a combined value and text field. Do not display both on the combo box filter page. Create a column property with View Type set to Filter (Control) to hide the combined value and text field. If the list view contains other fields that do not aid in the combo box search, hide them with a Control Status column property.
- Create a column property with View Type set to Control to define the list view properties for fields which have check tables.

To create a list box in DSP™:

1. Access the page in the WebApp.
2. Right-click the column heading,
3. Select **Add > List Box**.
4. Select the list box view from **List Source** list box.
5. Select the field that stores the values in the view from **List Value Field** list box.
6. Select the field that stores the name of the field from **List Display Field** list box.

NOTE: In most cases, List Value Fie**ld** is the primary key stored in the table and List Display Field is the option visible in the list box.

7. Click **Save**.

To create a combo box in DSP™:

1. Access the page in the WebApp.
2. Right-click **the** column heading.
3. Select **Add > Combo Box**.
4. Select the combo box view created in SQL from **List Source** list box.
5. Select the value that stores the field from **List Value Field** list box.
6. Select the field that displays a description of the field from **List Display Field** list box.

NOTE: In most cases, List Value Field is the primary key stored in the table and List Display Field is the option visible in the list box.

7. Click **Save**.

List Boxes, Combo Boxes and Filter Features

The following sections demonstrate the many features that can be added to a basic list box or combo box.

Enable List Unique

List Unique is a Column Property that displays list box values not previously selected in another record on the page. If the record has already been added, the List Unique property removes it from the list box, leaving only the records that have not been selected.

To enable List Unique for a list box:

1. Access the WebApp page.
2. Right-click the list box column heading.
3. Select **Edit (List Box).**
4. Click **Advanced Properties** Tab.
5. Click **List Unique** check box to enable it.
6. Click **Save**.

Enable List Allow Insert

List Allow Insert is a list box and combo box search mechanism if the value stored in the field is known or additional values not available in the list need to be added. If this property is enabled, the value entered displays under the search box for the list box or combo box. Click the value entered to use that value.

Options for List Allow Insert are:

- **No** – Insert does not display and additional options cannot be added or searched.
- **Yes** – Insert displays and **any** additional option can be entered in the field regardless of the options available in the list box. The record is considered valid. The added value does **not** become a future option in the table.
- **Constrained to table** –A user can add a new entry by clicking **Use this Value**. A validation rule automatically runs to verify the entry is valid. This prevents undesirable data being added to the list table.

List Allow Insert Page ID is a column property that works in conjunction with List Allow Insert. This feature permits additional values not available in the list to be added to the associated list. If both are enabled, clicking the icon opens a window containing the page associated with the list. Enter the new options in the fields and the value is available in the list box when adding new records.

NOTE: The current user must have security to add new records to the page associated with the list.

To enable List Allow Insert Page ID:

1. Access the WebApp page.
2. Right-click the column heading, for the list box or combo box.
3. Select **Edit (List Box)/(Combo box)**.
4. Click **Advanced Proper**ties tab.
5. Select **Yes** from **List Allow Insert** list box.

NOTE: Yes must be selected so a new record can be added and validated.

6. Select the page name from **List Allow Insert Page ID** list box.

7. Click **Save**.

Enable List Where Clause

Add a list Where Clause to restrict the values that display in a list box or combo box.

To create a List Where Clause:

1. Access the WebApp page.
2. Right-click the list box or combo box column heading.
3. Select **Edit (List Box)/(Combo Box)**.
4. Click **Advanced Properties** tab.
5. Enter the WHERE clause in **List Where Clause** field.
6. Click **Save**.

NOTE: For nvarchar and nchar database fields, the **N** must be prepended to string literals in Where Clauses so Unicode characters will not be lost and the value must have single quotes on either side. This is unnecessary for numeric fields.

Add a List Selection Field

The List Selection Field is a feature that displays more descriptive text in a list box or a combo box when in add or edit mode. To use this feature, modify the view in SQL and update the List Selection Field in DSP™.

For example, two fields, **ProductID** and **Description** must display in the **ProductID** list box when in add or edit mode on the *Order Details* page. The description should only be visible in display mode. For both fields to display in the **Product ID** list box in edit mode, add a column in the SQL list view to concatenate the two fields.

NOTE: This manual does not include steps for working in SQL.

To choose a more descriptive field for a list box in edit mode:

1. Right-click the list box or combo box column heading where the concatenated fields must display.
2. Select **Edit (List Box)**/(Combo box).
3. Click **Advanced Properties** tab.
4. Select the concatenated or more descriptive field created in the view from **List Selection Field** list box.
5. Click **Save**.

Set the List Order By

By default, like pages, list boxes are sorted based on the primary key. Use List Order By to sort select options in a list box based on another field.

NOTE: DSP™ and SQL Server no longer support the sort in a view in SQL.

To order the list box options:

1. Access the WebApp page.
2. Right-click the list box column heading that requires sorting.
3. Select **Edit (List Box)**.
4. Click **Advanced Properties** tab.
5. Enter **t**he column that should sort n **List Order By** field.
6. Click **Save**.

NOTE: When a field name is entered in **List Order By** field, the default order is ascending. To sort in descending order, enter the field name followed by a space and **desc**, for example, **ProductName desc**. Separate entries by a comma.

Set a Dynamic List Box

Dynamic List Box is a feature that uses a Where Clause to control the values displayed in one list box based on the value selected from another list box. Dynamic List Boxes can be used for list boxes on the same page view.

For example a user could add a list box and create a Where Clause on the State field on the *Customers* page so the country selected determines the values displayed in the States/Regions list box.

To filter the options in one list box based on the value selected from another on the same view:

1. Access the WebApp page.
2. Right-click the column heading for the column that will display the list box that will be updated based on the selection in another list box.
3. Select Add > **List Box**.
4. Select the view from the database that provides the values to the list box from **List Source** list box.
5. Select the name of the column that will be stored from **List Value Field** list box.
6. Select the name of the column that displays a description of the item from **List Display Field** list box.
7. Click **Save**.

To create a WHERE clause that controls the options in a list box based on a value selected from another list box for the same page view:

1. Right-click the list box column heading.
2. Select **Edit** (**List Box)**.
3. Click **Advanced Properties** tab.
4. Enter the WHERE clause in the **List Where Clause** field.

NOTE: For example, enter **CountryID = N'#Country#'. N** must be used on Where Clauses with text so Unicode characters will not be lost and the value must have single quotes on either side to indicate a text field; they are not needed if the field is numeric. The field name to the left of the operator is the field name in the list view. The field name to the right of the operator is being passed in from the page. Surrounding the column name with pound signs (#) indicates it is a dynamic field and the value is coming from the page view.

5. Click **Save**.

Add a Filtered Combo Box

Filters are designed to reduce the number of items displayed in the value field or the descriptive field.

When a filter is applied to a combo box, the Filter icon displays next to the combo box, indicating a filter is enabled.

To add a filter to a combo box:

1. Access the WebApp page.
2. Right-click the combo box column name.
3. Select **Edit (Combo Box)**.
4. Click **Advanced Properties** tab.

5. Click **List Filter** check box.
6. Click **Save**.

NOTE: Fields on the filter can be customized. Any column property can be applied to any filter.

Add a Watermark

Some columns benefit from a watermark, which displays as grey text in the field.

A watermark can be added in two ways: A static text watermark can be added to a column, or a view can be registered that displays the watermark depending on certain conditions.

For example, for list boxes that search for particularly named values/objects a watermark can display the expected formats to ensure that the user understands what's expected. Also, suggested text formats can be useful. If the user is expected to use a namespace convention, indicating it during the add/edit is more useful than providing a warning after the save.

To add a static text watermark to a column:

1. Access the WebApp page.
2. Right-click the text box column name.
3. Select **Edit (Text Box)**.
4. Enter the text in the **Watermark Text** field.
5. Click **Save**.

To use a view to display a watermark:

1. Write a view with a Dwv suffix and alias the column as boawatermark.
2. Access the WebApp page.
3. Right-click the text box column name.
4. Select **Edit (Text Box)**.
5. Click the **Relational** tab.
6. Click **Edit**.
7. Select the view in the **Watermark View** list box.
8. Enter the binding field name(s) in the **Watermark Binding Field Names** text box.

NOTE: Include a comma between names.

9. Click **Save**.

Add a Tooltip

Some columns may benefit from a tooltip, such as images that represent a status. The status name or description could display when a user hovers the cursor over the status icon. For instance, a red light in a column called "Status" could have the tooltip: "X templates failed to process". A tooltip can also be useful when a field is disabled due to lack of prerequisites. If a "Submit" button is disabled because the record is "Inactive", then the tooltip can state "Disabled due to being inactive".

First, write a view with the suffix Dtv for each column that requires a tooltip, alias the column as "boatooltip" in the view.

For example, the Finish button on a page is disabled when the Dependency Complete field is false. To display a short message that says why the button is disabled, the user creates the following view.

```
CREATE VIEW webRequestRoleDtv

AS

SELECT RequestID, RoleID, CASE Validate WHEN 0 THEN 'Dependency not complete for role' ELSE NULL END AS boaTooltip

FROM dbo.webRequestRoleDcv
```

The view uses a CASE statement, and sets the column to NULL when the tooltip should not display.

After the view is written, register it the column.

To register the tooltip view:

1. Right-click the image column name.
2. Select **Edit (Image)**.
3. Click the **Relational** tab.
4. Click **Edit**.
5. Select the view in the **Data Tooltip View** list box.
6. Enter the binding field name(s) in the **Data Tooltip Binding Field Names** text box.

NOTE: Include a comma between names.

7. Click **Save**.

Set Required Fields

Records containing fields that are Hard Required cannot be saved until the field is populated. Records with Soft Required fields can be saved but are not considered valid until the designated field is populated.

Fields can be Hard Required, Soft Required or not required and are differentiated by their background colors. The default color for Hard Required is green, Soft Required is yellow and Not Required is white; however, these colors are configurable in Class Styles under Configuration in System Admin.

Any field in a table that does not allow NULLS automatically displays as a Hard Required field in DSP™. In many cases it will be the Primary Key field. For example, the *Order Detail* page displays the Product ID field as required because it is a Primary Key, the Quantity fields and Discount fields are not automatically required because the table requirement allows NULLS.

Set a Hard Required Field

Only certain controls, such as check boxes or text boxes, allow a field to be set as required.

To make a field Hard Required:

1. Access the WebApp page.

2. Right-click the column heading that must be required.
3. Select **Add** > **[Control Name}**.
4. Select **Yes (hard)** from the **Required** list box.
5. Click **Save**.

Set a Soft Required Field

If a Soft Required field is not populated, the record can be saved, but it is considered invalid until the field is populated.

Applying the Soft Required column property automatically creates a view in SQL. DSP™ then registers the view as a Validation Rule to the page and creates a default message that displays to the end user.

To make a field a Soft Required field:

1. Access the WebApp page.
2. Right-click the column heading that must be required.
3. Select **Add** > **[Control Name}**.
4. Select **Yes (soft)** from the **Required** list box.
5. Click **Save**.

Assign Page Properties

Page Properties are used to control how the page displays and define actions that can be performed on the page, including functions on the Page toolbar.

To add page properties in DSP™ either:

- Log in to System Administration. Navigate to WebApps, locate the WebApp and click **Pages**, click the **Vertical View** for the page.
- Navigate to the page in the WebApp where the page property is to be applied. Click the **Design Page** icon in the Site Toolbar then click **Vertical View**.

Order By

Order By is a Page Property that sort columns in either ascending or descending order based on Field Name on the *Horizontal* View only. Sorting must be done in DSP™ not in the SQL view.

To sort a column in ascending order:

1. Access the page in the WebApp.
2. Click the **Design Page** icon on the Site toolbar.
3. Select **Design** from the Menu.
4. Click **Vertical View** for the page.
5. Click the **Edit** button.
6. Enter the field name in **Order By** field.

NOTE: When a field name is entered in **Order By** field, the default order is ascending. To sort in descending order, enter the field name followed by a space and **desc**, for example, **lastname desc**. Separate entries by a comma.

7. Click **Save**.

Enable Persistent Insert

Persistent Insert is a feature that forces the page to remain in add mode after a record is entered and saved.

To enable Persistent Insert:

1. Access the page in the WebApp.
2. Click the **Design Page** icon on the Site toolbar.
3. Select **Design**.
4. Click the **Vertical View** for the page.
5. Click **Advanced Properties** tab.
6. Click **Support Persistent Insert** check box.

Disable Support Download

Support Download is a Page Property that allows records on the page to download to an Excel spreadsheet, Word document or other formats. By default, Download is enabled for all pages. When Support Download is disabled, the Download is not available.

Refer to *Enable File Upload/Download* for more information.

To disable download capabilities:

1. Access the page in the WebApp.
2. Click the **Design Page** icon on the Site toolbar.
3. Select **Design**.
4. Click the **Vertical View** for the page.
5. Click **Advanced Properties** tab.
6. Click **Support Download** check box to disable.

Copy Page

An existing page in a WebApp can be copied. It can have a different name and can be placed in the same WebApp or another WebApp. The entire page and all dependents, such as Column Properties, Events, Validations, Business Rules and/or WebApp Security Groups can be copied.

NOTE: To copy Validation and/or Business Rules, Events **must** also be copied.

To copy a page:

1. Access the WebApp page.
2. Click the **Design Page** icon on the Site toolbar.
3. Select **Design**.
4. Click **Vertical View** for the page.
5. Click **Advanced Properties** tab.
6. Click **Copy** button.
7. Click the **Edit** button.
8. Enter a description in **Target Description** field.
9. Verify **Target WebApp ID** is correct.
10. Select which items to copy for the page.

11. Click **Save**.
12. Click **Copy** button, again.

NOTE: The **Copy** button must be clicked again to officially copy the page. The new page displays on the *Pages* page.

Enable File Upload/Download

The Files Column Property is used to stage any type of file including graphics, text, Word or Excel files. Files are stored in a specific location on the web server and are enabled for download.

To enable file upload and download:

1. Access the WebApp page.
2. Right-click the field name.
3. Choose **Add > File**.
4. Enter the path name to upload files to and to store files for download in **File Path** field.

NOTE: The files are usually stored in a folder with the same name as the DSP™ WebApp ID (GUID). Use the forward slash / within and at the end of the path. The forward slash at the end of the path denotes a folder. For example, if **/NWEmployees** is used, it is treated as a file name. **/NWEmployees/employee/** is treated as a folder.

The folder **UserArea** is created by DSP™ and should always be the first folder name in the path. The English folder is required and must be added by the user. The WebApp ID sets the path to the WebApp. Other folders or sub folders can be used but the English folder is used as the default.

5. Select **Virtual** from **File Path Type** list box.
6. Select **Upload or Download** from **File Direction** list box.
7. Click **Save**.

Set Security

When a new WebApp is created, the User and PowerUser WebApp Security groups are automatically generated with default assignments. The user that created the WebApp is added to the PowerUser WebApp Security group. New pages added to the WebApp are also assigned to the PowerUser group.

Business processes often require new WebApp Security groups to be registered with more stringent security to the visibility and the control (add, edit, delete) of sensitive data. This section describes how to assign specific access to pages and menus through the Security and Design security group functions, creating a secured business environment. The user creating the secured environment must have access to both Security and Design security groups. Some companies may require two individuals to perform these functions.

NOTE: These steps must be performed by the Security Administrator and Designer.

To add a user and grant access to pages in a WebApp at a high level:

Security Administrator

1. Create WebApp Security Group
2. Add User to WebApp
3. Add User to WebApp Security Group

Designer

4. Add page(s) to WebApp Security Group
5. Add WebApp Home Page to WebApp Security Group
6. Grant Page rights (Allow Select, Allow Insert, Allow Update, and Allow Delete) to page(s)

Refer to *Add Pages and Grant Access to Security Group* for more information.

Effectively Implementing Security Definitions (Role Security)

Security Definitions (commonly referred to as Role Security Configuration) is a feature that enables application developers to:

- Internally restrict data access to particular users (or roles) and
- Provide an external API to other applications to secure their data on the source of the data itself. This allows inter-dependent applications to share a single security model across the platform.

The secured data is then made available to users via direct access or Roles. Good candidates for data that can be secured via these Security Definitions are:

- Low record count tables / values (i.e., only ever be <100 unique values)
- Very high level data sets (not low on the hierarchy, data won't build exponentially)
- Landing page data that's usually the first page(s) the user sees when utilizing the app

NOTE: You do NOT have to create a security definition only using a table's primary keys. The Security Definition keys could be a commonality that data sets share. It could also be user configuration data driven, not predefined by the application designer, as long as the application designer is aware of the likely data sets that will make up the Security Definitions data set.

These data sets are managed by an onsite administrator on a daily basis. The fewer total Security Definition records there are, the easier it is for them to manage.

NOTE: If there is already a role that contains the relevant Web Application access and data access, the user can be directly assigned to the role from the *Roles* page.

NOTE: If the user is not expected to participate in any Roles, directly application access can be grated via the WebApp Security pages.

For example, an application allows users to create a request, which can then be approved or rejected by another user, and then finally completed by a final user.

This application has three roles built into it, "Requester," "Approver," and "Request Completer" where:

- Requester – Can create, edit, and remove requests
- Approver – Can accept or reject requests (but not delete)
- Request Completer – Can close the request by completing it

These roles allow the designer to manipulate application flow and restrict access to an individual user, to show them only what's relevant for the task they are attempting to accomplish.

Creating a Security View Guidelines

In most cases, Security Definition views should be created using the original source of the data that the view will be securing against. This way, when new records are added, the security view will implicitly contain those records.

It's also a good idea to apply the security definition to the page(s) that maintain the source of the data. This way, the "Update User Definition on Save" option can be enabled on those pages, ensuring that the user who creates the data will always have implicit access.

WebApp Security Group Design

When designing an application, the designer should know the different types of users who will access the application. Some applications are only meant to have a single user who has full access to the features, pages, and functionality that will be put into the application. However, more elaborate and partitioned applications may expect different types of users to have access to certain pages.

WebApp Security Groups are a good way for Application Designers to convey expected roles to the On Site Administrator, without requiring "Use Case" documentation.

Pitfalls of Using WebApp Security Groups

- Groups may go unnoticed and under-used. Security Administrators may not know how to add control logic to supplement the WebApp Security Group's functionality.
- Groups are often an after-thought to an application's implementation. Adding them disrupts the original flow of the application's design.

NOTE: User profiling upfront can help prevent these pitfalls, by understanding the usage scenarios and planning for them.

- Out of the box, WebApp Security groups have limited restriction capabilities (Page Access, Insert access, Update access, Delete access). However, WebApp Security Groups can be supplemented by more granular features:
  - boaUserGroup can be used as an Environment variable to restrict control view information
  - System Views boaWebAppGroupUserSel and boaWebAppSel can be used in conjunction with other page-related views

NOTE: Unlike with User Control Views, there is no record-per-group equivalent control view. When attempting to implement boaUserGroup filter in, for example, a Toolbar View, there is the possibility of a user existing in two groups. In this case, a WebApp Design Error results, because Toolbar Views are limited to returning one row.

Benefits of Using WebApp Security Groups

Web App Security Group restriction logic is usually more simplified than trying to compute it at a *User Control View or Data Control View* level. The overhead on this is much less, increasing performance while maintaining relatively simple logic. The overhead also scales with WebApp Security Group count, instead of User count, where applications rarely have more than 10 Groups.

Access manipulation of WebApp Security Groups is performed in System Administration, instead of within the WebApp itself. This reduces the number of configuration pages required within the WebApp and does not require user configuration to happen in both the WebApp and System Administration.

Create WebApp Security Group

WebApp Security Groups are user definable depending on specific business requirements. Any number of groups can be created. Users must then be assigned to a WebApp Security Group.

Only users assigned to the Security and PowerUser WebApp Security Groups in System Administration have authority to create a WebApp Security Group. However, adding a user to PowerUser is not recommended. As new pages are created for a WebApp, they are automatically added to the PowerUser WebApp Security Group. Therefore, members of this group always have access to all pages, unless otherwise configured.

To create a WebApp Security Group:

1. Click **Admin** > **Security** > **WebApp Security** in the *Navigation* pane.
2. Click the **Groups** icon for the WebApp**.**
3. Click **Add**.
4. Enter a name in **GROUP NAME** field.
5. Click **Save**.

Guidelines for Configuring a New User

When creating a new user for DSP™ access, there are many first time setup steps to go through. This document guides an Administrator through these steps.

General guidelines include:

- When adding an account, it's recommend that the Password Last Changed value is set to several years in the past. This ensures that on initial login, the user is forced to change the password from the one given to them.
- If the user is expected to only work in a particular application, or has a predefined work flow that is expected, the Default Page ID can be set to the relevant location.
- If the User's primary language is different than the site's default, the Language ID can be set during creation. This can be changed by the User from the *Settings* page via the Change Setting icon on the Site toolbar.
- If the User requires a High Contrast UI, the Style ID can be set to DSP High Contrast #1 or DSP High Contrast #2. This can be changed by the User from the *Settings* page via the Change Setting icon on the Site toolbar.
- If the Locale of the user differs from that of the Web Servers, the Locale can be set for that user. This can be changed by the User from the *Settings* page via the Change Setting icon on the Site toolbar.

Add User to WebApp

To add a user to a WebApp:

1. Click **Admin** > **Security** > **WebApp Security** in the *Navigation* pane.
2. Click **Users** for the Web App.
3. Click **Add**.
4. Select an ID from **USER ID** list box.
5. Click **Save**.

Add User to WebApp Security Group

New users are automatically added to the User WebApp Security Group of DSP™, not to a specific WebApp. The User WebApp Security Groups only permits **Allow Select** access to the main page. To view and access a WebApp, the user must be assigned to the specific WebApp and added to a WebApp Security Group.

Only users assigned to the Security WebApp Security Group in the System Administration WebApp have authority to assign users to WebApps.

If users are added to more than one WebApp Security Group within a single WebApp, they inherit the combined access of all groups.

The **Expiration Date** allows for temporary security to be granted. This feature is useful for interns, temporary employees, etc. It is also useful when another employee is filling in for a person on vacation. The security automatically expires without an administrator manually changing the security.

To assign users to WebApp Security Groups:

1. Click **Admin** > **Security** > **WebApp Security** in the *Navigation* pane.
2. Click **Groups** for a WebApp.
3. Click the **Users** icon for a group name.
4. Click **Add**.
5. Select the user added to the WebApp from **USER ID** list box.
6. If the user's login is temporary :

   Enter a date in **Expiration Date**, if applicable.

   Or

   Click **Calendar** icon to select a date.
7. Click **Save**.

Add Pages and Grant Access to Security Group

Access rights further define WebApp Security Group page rights, such as Allow Select, Allow Insert, Allow Update and Allow Delete. The following table outlines DSP™ page access rights.

Page rights	Access
Allow Select	Users in the group can view the page and data on the page. Users may download the data, if download is supported in the page properties and WebApp Security Group.
Allow Insert	Users in the group may add data to the page, if insert is supported in the page properties and WebApp Security Group.

Page rights	Access
Allow Update	Users in the group may update the page, if update is supported in the page properties and WebApp Security Group.
Allow Delete	Users in the group may delete the page or rows from the page, if delete is supported in the page properties and WebApp Security Group.

To add a page to a WebApp Security Group:

1. Click **Admin > WebApps** on the *Navigation* pane.
2. Click **Groups** for a WebApp.
3. Click **Pages** for a Group Name.
4. Click **Add**.
5. Select the page from **PAGE ID** list box.
6. Click **Save**, add mode persists.
7. Add another page to the WebApp Security Group or click **Cancel**.

NOTE: For a page to be accessed, security must be granted to the WebApp itself.

Column Encryption

Column-level encryption is used to deny unauthorized users and users with access to the underlying database visibility to information of a sensitive nature.

Terminology

- **FIPS 197 Compliance** – Federal Information Processing Standard 197 is a US Government standard that provides best practices for implementing crypto algorithms, handling key material and data buffers, and working with the operating system.
- **Decrypt/Pass** – When reading data from an encrypted column, the decrypt mechanism is designed to look at the data, find the *magic marker* and attempt to decrypt the data. If the *magic marker* does not exist, there is no attempt to decrypt the data and the data is preserved as is.

Column Encryption Guidelines

When encrypting a column:

- All encryption and decryption is performed at the application level. Data entered into the database outside of the DSP™-specific mechanism will be seen as plain text because this is application-level encryption and decryption.
- Triple DES or AES-256 is used.
- Encryption Keys are stored in the database. Each instance of DSP™ can have one or more encryption keys that can be used to encrypt one or more columns.
- Proprietary keys are protected by a Site Master Password (which is system-generated if one is not specified) that must be identical across all servers for a specific instance. The Site Master Password is stored in web.config. The system-generated Site Master Password is based on the server name and the database; therefore, all host files or DNS aliases must be in place and identical across machines.
- Users only need rights to view the page to view the encrypted data.
- Encryption is available to developers through the plugin API with encrypt and decrypt/pass. To use API, the KeyID must be known (located in the Key table, GUID).

- Encrypted columns must be nullable and be configured with data type nvarchar. The size of the column must be between 128 and 256.
- WebApps with encrypted data must be decrypted before a CTS package is built. Move the package, then re-encrypt.
- Database-level mechanisms cannot be used to read or write encrypted data. These operations must be performed by plugins or through the DSP™ front end.
- Joins, search or indexing are not supported on encrypted columns.
- Once data is encrypted with a specific key, the key cannot be deleted until the data is decrypted.

Encrypt a Column

To encrypt a column:

1. Click **Admin > Data Sources** on the *Navigation* pane.
2. Click **Encryption** for the WebApp.
3. Select the name of the table that contains the column from **Table Name** list box.
4. Select the column name to encrypt from **Column Name** list box.
5. Select **System Administration Passwords** from **Key ID** list box.
6. Click **Save**.
7. Click **Enable** button under the Encryption column to encrypt the values.

Catalogs

DSP™ catalogs provide a means to populate and administer translations, help text and alias values at the WebApp and page levels.

Catalogs exist in the system at these levels:

- **Page-specific Catalogs** – Catalogs that exist for a given page. Translations, help text and alias values in these types of catalog are only applicable for the one page to which the catalog is linked.
- **WebApp Catalogs** – Catalogs that are created generally in System Administration and are applied to one or many WebApps. Translations, help text and alias values found in these catalogs apply to all pages in a WebApp to which it is assigned.

When phrases on a page are translated, DSP™ searches through all catalogs assigned to the page. A translation is attempted to be made at the highest level catalog. If a match is not found, the next level catalog is searched until a phrase translation match is found.

It is recommended to translate phrases at the most generic level appropriate. The higher the level the phrase is translated, the more useful the translation will be. A phrase translated at the Catalog (WebApp) level can be reused by many applications and many pages. A phrase translated at the page level is only good for that single page.

There are cases where a Phrase has a unique meaning on a page, different from its *usual* meaning, and should be translated at the page level. However, page level translation should be the exception to the rule.

Create WebApp Catalog

WebApp Catalog is the most common and useful level for catalog registration so that translations, help text and alias values can be administered at the WebApp level as well as across WebApps.

To create a Catalog for a specific WebApp:

1. Click **Admin** > **Translations** > **Catalogs** on the *Navigation* pane.
2. Click **Add**.
3. Enter the name for the catalog in the WebApp in **NAME** field.
4. Click **Save**.

Register Catalog to WebApp

Once the Catalog is created, assign it to a WebApp. Multiple catalogs can be registered to a single WebApp. Catalogs are searched according to the assigned priority in order to find a phrase translation. Once a phrase translation is found, DSP™ stops attempting to translate the phrase. If, after searching through every catalog assigned to the WebApp, a phrase translation is not found, the field/column name displays.

To assign a WebApp Catalog:

1. Click **Admin** > **WebApps** on the *Navigation* pane.
2. Click **Catalogs** for the WebApp.
3. Enter a number in **PRIORITY** field.
4. Select the catalog name from **CATALOG ID** list box.
5. Click **Save**.

Add Phrase to a Catalog

Once the catalog is added to the WebApp, translations, alias and help text can be added to the catalog. All columns within the WebApp with the same column name will be aliased (i.e., replace the column name with the phrase). Phrase files can be imported or exported at the WebApp catalog level. This feature is useful in the following instances:

- Make multiple phrase changes
- Generate a report of all phrases to analyze or view
- Generate a catalog outside of DSP™ and import

To add a phrase:

1. Click **Admin** > **Translations** > **Catalogs** on the *Navigation* pane.
2. Click **Phrases**.
3. Click **Add**.
4. Enter the text to translate in **PHRASE** field.
5. Enter the text that displays as the translation in **PHRASE OUT** field.
6. Click **Save**.

Add Phrase Out Values to Alias a Column on a Page

Phrases can be changed for a specific page verses for the entire WebApp.

To change a phrase for a specific column:

1. Click **Admin** > **Translations** > **WebApps** on the *Navigation* pane.
2. Click the **Pages** icon for the WebApp.
3. Click **Phrases** for the page catalog.

4. Enter the phrase to be translated across the WebApp in **PHRASE** field.
5. Click **Save**.
6. Click **Translations** for **the phrase**.
7. Click **Add**.
8. Select **(Page)** from **Catalog ID** list box.
9. Enter the text to translate to in **Phrase Out** field.
10. Click **Save**.

Add Column Help Text

Column help text can be added to any column/field on a page or the entire WebApp. To view column help, hover over the column heading. The help displays below the *Navigation* pane in the Information pane. By default the Column name and field name display.

To add column help text (if the column property already exists, skip steps 4-9):

1. Access the WebApp page.
2. Click the **Design Page** icon on the Site toolbar.
3. Click the **Column Properties** icon.
4. Click **Add**.
5. Select the name of the field that requires help from **Column** list box.
6. Select **Text Box** from **Control** list box.
7. Click the **Save** button, the *Vertical* View displays.
8. Click **Save**.
9. Click the **Cancel** button to cancel Insert mode.
10. Click the **Help** icon for the column.
11. If not in Add mode, click the **Add** button.
12. Select **(Page)** from **Catalog ID** list box.
13. Enter **the text that displays as help** in the **Dynamic Help** Editor field.
14. Click **Save**.

Events

Events are actions that contain Validation Rules and/or Business Rules. Validation rules run first and validate data before executing any rules. Business rules run after validation rules complete and perform an action. They can be incorporated into each page or for a specific field triggering an action or event. Validation rules always run in the foreground; however, business rules can be specified to run in the foreground or the background depending on the complexity.

- Foreground Events – Events selected to run in the foreground execute immediately when the status icon is clicked and the page refreshes.
- Background Events – Business rules that take a longer time to run can be forced to run in the background. Background events are placed into a job queue for later execution. The status is queued when the rule is first submitted. When the event is running, the boaStatus becomes Executing (11). It then becomes Execution Failed (12) or Complete (4).

Event rules can be created to run:

- **OnLoad** – Executes prior to the page displaying allowing lazy data population.
- **OnValidate** – Executes when a record is saved or the boaStatus icon is clicked.
- **BeforeDelete** – Executes before a record can officially be deleted.
- **Columns** – Executes at the record level from clicking a specific field.

There are two ways to add events in DSP™:

- Log in to System Administration. Navigate to WebApps, locate the WebApp and click **Pages**. Locate the page and click **Events**.
- Navigate to the page in the WebApp where the column property is to be applied. Click the **Design Page** icon on the Site toolbar, then click **Events**.

Event Design Process

Below is a list of steps that can be followed to ensure an event is fully implemented with all available and relevant features to ensure the highest quality user experience.

NOTE: These options are set on the *Page Events* page (**Admin > WebApps> Pages > Events**).

- Is this a large or long running process best suited for the background, or is it fast or need user session level context? A foreground event will run on the Web Server, which can give a plugin code access to specific user and runtime information from the web session. It can also provide immediate feedback to the user, such as reloading the page and displaying post event messages (a Post Message and any messages sent to the user during execution). Any event running for more than a few seconds should be run in the background. Configure this option using the EVENT PROCESS TYPE ID list box on the *Page Events* page.
- Should the user be warned before executing the event? If so, include a Pre Message, which will prompt the user with an Ok or Cancel option for continuing the execution of the event. Most irreversible or not easily reversible actions should include a Pre Message to ensure the user doesn't accidently disrupt their own (or someone else's) process. Enter the pre message in the Pre Message field on the *Page Events* page's *Vertical* View.
- Should a Post Message be included? Post messages display at different times depending on whether the event is foreground or background. A background event will display the message as a Minor Notification, immediately after the job has been queued. In this case, it's a good idea to use the post message to indicate to the user that the work will be run at a certain time (e.g., "The package will be processed shortly."). Foreground events will display their message after execution. Post Messages for foreground events are useful when executing plugin code that doesn't always write out a message, or when executing non-plugin business rules. This will ensure that the user always receives a confirmation that their action has completed. Enter the Post Message in the Post Message field on the *Page Events* page's *Vertical* View.
- Should the Transaction Method be changed from the default (Disabled)? Disabled Transaction Method means that the business rules stop on exception. For example, if three procedures perform a single record insert, and the second procedure fails, the first will still have inserted its single record. An Enabled Transaction Method will encapsulate the business rules in a database transaction, so that, in the aforementioned scenario, the transaction would roll back causing no new records to be committed. A Serialized Transaction Method will only allow one instance of the page's event to run at a time. For example, if User A and User B are attempting to execute the event

(even against different records), one event has to complete before the other one starts. Set the Transaction Method on the *Page Events* page's *Vertical* View on the Advanced Properties tab.

- What "Event Level" should this event execute at? Page is commonly used for toolbars (more specifically page toolbar view toolbars) which will cause no row-level information to be passed to the event. The default and most commonly used Event Level is row, which will pass the row information to the event. Set the Event Level on the *Page Events* page's *Vertical* View on the Advanced Properties tab.
- Should a post event "Function" be utilized? This modifier allows the page designer to cause the user to be navigated back to their previous page, or "to the page's *Vertical* View. These use cases can be useful for a few scenarios. If the designer selects "Back", the User may be expected to navigate to a second page, action a particular button, and then be sent back to their previous page where that event has affected the previous record. The "Vertical" option can be used when an action on the *Horizontal* View results in information in the *Vertical* View that is always relevant to show the user, perhaps a more verbose message they need to read based on what they actioned. Set the Function on the *Page Events* page's *Vertical* View on the Advanced Properties tab.

OnLoad Events

Use OnLoad Events sparingly and only when absolutely necessary. Executing a SQL Stored Procedure automatically increases page load time by at least 100 milliseconds. Page load time should never exceed a full second for a good user experience, so merely the existence of an OnLoad event can attribute to 10% of that ideal ceiling.

Event Complexity

An event that runs every single page load should perform very little work. If it is an event that is only meant to run once after a certain scenario, then there should be a very fast check to determine whether the condition has been met. For example, a Metrics page may need to be refreshed after data on a Task page has been modified. Instead of re-computing metrics after every update on the Task page, toggle a dirty flag OnValidate of a record on the Task page. Then, the OnLoad event on the Metrics page can determine whether a recompilation is necessary or not. Even though it will add a slight amount of complexity to the Task page, the load time of the Metrics page is only ever affected when it needs to be. If the check only takes 10ms, we save the majority of page's load's 450ms.

Complex events can also be used to reduce the scope of the processing based on, for example, binding or key criteria, distributing that 500ms among several other page navigations. This further distributes the overhead to when it actually needs to be utilized.

Complex Events can also be used for:

- Preparing for rendering static content (like an .aspx page)
- Computing or Recomputing metrics
- Creating a temporary record (commonly used as form input)

OnValidate Events

OnValidate events run at three specific times:

- After an insert
- After an edit
- When the boaStatus cell is actioned.

It's important to determine which of these are actually relevant to the business logic and what can be done to ensure complex logic only runs when necessary.

Compromising Between Implicit Behavior and Excess Overhead

It is easy to pile up computation and configuration logic onto an OnValidate event, because it requires the least amount of effort to ensure the logic runs as often as it needs to. However, this can introduce problems of complexity and run time.

An edge case example is on the *Data Sources* page in System Administration. An OnValidate Business Rule exists which attempts to create system views in the database. If the data source is accidentally configured with invalid connection information, every save will attempt to connect to the data source and prepare to create system views. An invalid configuration results in a timeout error after attempting to connect for 30 seconds. This is frustrating to the end user, because what appears to be a small change takes 30 seconds each time and then results in an error.

In this case, the logic of the event can be altered to check if it is meant to create the system views prior to attempting to access the database. This would eliminate this potential problem.

The alternative to having this event on OnValidate is creating a "Create System Views" or "Check System Views" button, which would result in fast adds/edits/updates. "System Views" are already an opt-in feature, meaning, the User has already made a conscious decision that this database needs System Views. It would be the same number of mouse clicks for the button as there are for the check box.

An extra click once to save 200ms every edit will go a long way.

Minimize Work by using Keys or Drill column values

Never perform "Global" operations using OnValidate. If the logic being run does not change between rows, you've implemented the business logic at the wrong level. Now, there are use cases where a child record will affect a parent record, but it's still utilizing data from the child row for that manipulation. That operation should only affect one record (the parent) by filtering the affecting SQL with column values passed through the navigation through of pages.

Validation Rules

During the design or modification of a WebApp, validations can be incorporated into each page or for a field based on an event. Validation Rules manage correct data entry. If the validation occurs upon save and fails, the user is prompted with a generated message. Validation Rules are based on a view in SQL that is registered to the page in DSP™.

The purpose of this section is to understand how Validation Rules function and how to apply these rules to a page.

NOTE: This document does not provide steps for working in SQL.

Validation Rule Guidelines

- Use the naming convention **webXXX_YYYYVal** where **XXX** is the primary table or page and **YYYY** describes what the validation rule is validating, e.g., **webOrderDetail_QtyGreaterThanOnHandVal**.
- DSP™ automatically generates a Validation Rule view for columns under the following conditions:
  - When a column property is created with the **Required** value set to **Required (soft)**.
  - When a column property is established for a list box or filtered text box with List Allow Insert set to Constrained to table.
  - When image or button column properties are added to a column and not linked to a page.

NOTE: Refer to *Assign Column Properties* for more information.

- Validations only run on the single page for which they are created. They are not connected or linked to additional pages. Inherit Validation enables validations to run for all linked pages up to 30 pages. Refer to *Enable Inherit Validations* for more information.
- Validation Rules are assigned a **Severity** on the *Page Validation Rules* page (**WebApps > Pages > Events > Validation Rules**) to dictate boastatus behavior. Severity types are:
  - **Error** – Marks the status as invalid until the error is corrected and the Validation Rule passes.
  - **Message** – The status is valid but a message displays.
  - **Warning** – Displays a message and the status can be validated or invalidated by the user.
  - **Workflow Only** – Works with Business Rules. If a record is valid the workflow defined as a Business Rule executes. If a record is not valid, the workflow defined as a Business Rule does not execute.
- Ensure the table has a boaStatus column. Reserved columns can be automatically added to tables using the Append Column feature. Refer to *Append BOA Reserved Columns to Tables* for more information.
- Is this validation rule only relevant under certain data conditions? If so, providing a "Conditional Column" in the view will allow for a 0 (disabled) or 1 (enabled) value to determine whether this validation rule should be executed. This can be useful when, for example, ensuring a field has a non-NULL value when another column is enabled.

Register a Validation Rule to a Page

To register the Validation Rule to a page in DSP™:

1. Access the page in the WebApp.
2. Click the **Design Page** icon on Site toolbar.
3. Click **Events** for the page.
4. Select **OnValidate** from **Event** list box.

NOTE: An **OnValidate** event must be added for every page that contains a **boaStatus** field, even if there are no Validation Rules or Business Rules registered to the page. **boaStatus** is not automatically updated by DSP™ if the **OnValidate** event is not created.

5. Click **Save**, the *Vertical* View displays.

NOTE: If populated, the Pre Message displays before the validation rule runs and the Post Message displays after the validation rule runs. Use the messages with consideration of the end user.

6. Click **Save**.
7. Return to the *Horizontal* View.
8. Click **Validation Rules**.
9. Enter a number in **PRIORITY** field.

NOTE: The **Priority** is the order in which the validation rules will run.

10. Select an option from **SEVERITY** list box.
11. Click **Save**, the *Vertical* View displays**.**
12. Select the view that stores the validation rule from **View** list box.
13. Enter text in the **Comment** field.

NOTE: The text in the **Comment** field displays to the end user when the validation rule fails.

14. Click **Save**.

Enable Inherit Validations

Pages in a WebApp are not organized in a hierarchy; however, the Inherit Validation feature uses controls linking to other pages to establish the hierarchy.

For example, if Inherit Validation is enabled on the link between *Orders* and *OrderDetail* pages, the OnValidate event of the *Orders* page also runs the validations from the OnValidate event on the *OrderDetail* page, just as if the validations rules were registered on the *Orders* page.

To enable Inherit Validations between a parent and child page:

1. Access the parent page in the WebApp.
2. Click the **Design Page** icon on the Site toolbar.
3. Click **Column Properties** for the page.
4. Click the link in **Link to Page ID** column that provides the navigation to the child page; the *Page Column Links* page displays.

NOTE: The column property is not applied to the linking column on the *parent* page but rather the link property to the *child* page.

5. Click **Advanced Properties** tab.
6. Click **Inherit Validations** check box.

Business Rules

During the design of a WebApp, business rules can be incorporated into each page or for a specific field triggering an action or event. For example, business rules can be created on the *Customers* page to force a customer's name to display in all upper case letters, or to default **Date Shipped** to be the same as **Order Date**. A page designer can add a button with a business rule that, when clicked, runs the rule that adds one day to the **Date Shipped** date.

The purpose of this section is to understand how Business Rules function and how to apply these rules to a page.

Business Rule Guidelines

- Business Rules are stored procedures which are generally created from a Select view.
- Common naming for stored procedures is webTable_Event_Description#Action where
  - **Table** – Table registered to the page that calls the business rule
  - **Event** – Event that calls the business rule
  - **Description** - Procedure description or field being modified
  - **#** – An optional number to prevent duplication
  - **Action** – Action performed by stored procedure: Upd if update; Ins if insert; Del if delete
- Name Stored procedures that include a select query view the same as the select view, without the Sel suffix.

	Governance Example	Migration Example
Select View	webRequestMasterRecipeStepsOperations_BeforeDelete_OpPhasesDelSel	webTargetSourceReportUpdSel
Stored Procedure	webRequestMasterRecipeStepsOperations_BeforeDelete_OpPhasesDel	webTargetSourceReportUpd
Select View	webRequestPurchasingContractItems_OnValidate_PSTYP_3UpdSel	webTargetSourceRuleInsSel
Stored Procedure	webRequestPurchasingContractItems_OnValidate_PSTYP_3Upd	webTargetSourceRuleIns

- It is recommended to base behavior off of boaStatus rather than record existence. To validate data, it must exist, however, if the application data is "invalid" an exceptions during execution could result. Considering boaStatus during record manipulation can ensure business logic will not fail unexpectedly.
- Determine under which validation conditions the business rule should execute. By default, business rules "Run on Validate", which means they execute if the record is left in a valid state after the Validation Rule's execution. However, there may be conditions where a business rule should ALWAYS execute. For example, a counter on how many times a button has been clicked is generally unconcerned with record validity and checking both "Run on Validate" and "Run on Validate Fail" would enable this. Alternatively, you may want to run an event on "On Validate Fail", to record information or update a parent record.
- Force Foreground Execution can result in slightly awkward behaviors. The intent of the feature is to ensure that the business rule runs against the Web Server, even if the event is set to the background. This may be needed for things like plugin access to the HttpContext. However, depending on chain event calling (like Public Web App Events) it may not always behave as expected. It also results in the business rule being executed twice (Foreground and still run in the background).
- Business rules should NEVER throw Internal Errors. Validation rules exist to ensure that the expected configuration for success exists prior to executing. Any runtime failures that may occur (e.g. database connection failures) should be Try/Catch formatted typed Messages back to the user.

NOTE: The view and stored procedure must have already been created in SQL. This document does not provide steps for working in SQL.

Create a Business Rule for a Field

To create a business rule for a field:

1. Access the WebApp page.
2. Click the **Design Page** icon on the Site toolbar.
3. Click **Events**.
4. Click **Business Rules**.
5. Enter a value in **PRIORITY** field to determine execution order.
6. Verify **ACTIVE** check box is enabled.
7. Verify **Stored Procedure** is selected from **the PROCEDURE TYPE** list box.
8. Click **Save**; the *Vertical* View displays.
9. Select the name of the stored procedure from **Procedure** list box.
10. Enter text describing what the stored procedure does in the **Comment** field.

NOTE: This comment is a description of the event; it does not display to the end user.

11. Click **Advanced Properties** label to expand label set.
12. Verify **Run On Validate** check box is enabled.
13. Click **Save**.

NOTE: Business Rules run only after all validation rules on the page are processed without error.

Reports

Link to a Report from a Page

Images and buttons can be added to columns to be used as links to reports. By default, page links render data using the normal *Horizontal* or *Vertical* View with add/edit/delete capability. **Link to Report** provides a mechanism to report data as read-only without having to modify defaulted page properties.

To link to a report:

1. Access the page in the WebApp to link from.
2. Click the **Design Menu** icon on the Site toolbar**.**
3. Select **Design.**
4. Click **Column Properties**.
5. Click **Vertical View** for the image or button that will link to the report.
6. Click the report name to link to from the selected page in **Link to Page ID** field; the *Page Column Links* page displays.
7. Click **Link to Report** check box.

Report Follows Link

Report Follows Link is enabled by default and generates a report for the current Dynamic page for the *Horizontal* or *Vertical* Views.

A report containing data from multiple pages may need to be generated. By clicking the Page settings gear and selecting Report, the report follows any and all page links associated with the primary page.

To generate a report of a page:

1. Access the WebApp page.
2. Click the **Design Page** icon in the Site toolbar.
3. Select **Design.**
4. Click **Column Properties**.
5. Click **Vertical View** for the image or button that will link to the report.
6. Click report page link in **Link to Page ID** field; the *Page Column Links* page displays.
7. Click **Report Follows Link** check box.

NOTE: Report Follows Link instructs the web spider to include the WebApp page details in the report. This option can be applied to all other links on a page.

Dashboards and Charts

BackOffice Associates® Solutions supports the ability to create charts and dashboards to graphically display data. Individual charts of data are created by using the Chart page type. Dashboards are created using the Dashboard page type to create a collection of charts. For example, a dashboard can be configured to display tables as a bar chart and the package types used for downloads as a pie chart.

Charts can display numeric data against categorical data, dividing the data into two separate categories. The categories are value columns and category columns. Column properties and category column properties can be added to customize the charts. Pages can be linked to charts, charts to pages and charts can be linked to charts.

There are several steps required to create a chart page:

- Create a view in SQL.
- Register the view as a Chart page Type in DSP™.
- Customize the chart page by applying column properties.

All charts in DSP™ are based on a view in the associated database. Views and charting typically follow the pattern of using a category of data and a numeric value. A chart view contains the data to be displayed via the chart page and can reference multiple views and tables.

At least one column in the view must contain a Value field (for X-axis data) and a Category (for Y-axis data) field. There can be more than one Value column and one to two Category columns. If there are two Category columns, there can only be one Value column.

An extra column can be included in the view, which must be manually added to the *Page Columns* page where the Control = Category. The Platform only supports two column properties where Control = Category. Mark each category control as Primary and Secondary via the **Grouping Category** field on the *Vertical* View.

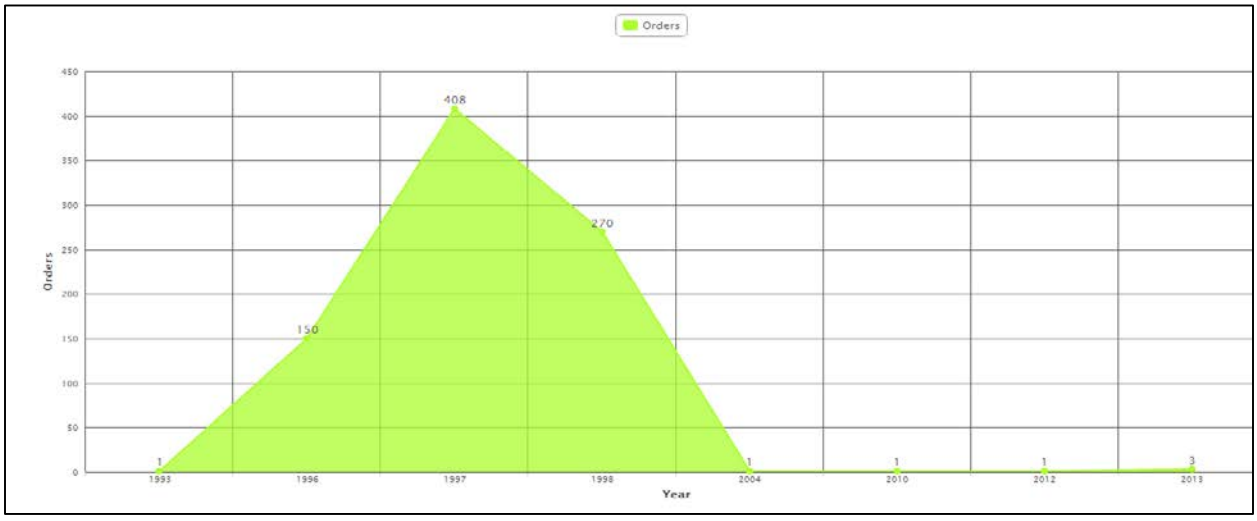
Use the naming convention **webXXXChart**, where **XXX** is a logical name for the chart page.

NOTE: BackOffice Associates® Solutions is delivered with a series of charts and dashboards. Custom charts are maintained during upgrades to a new version of BackOffice Associates® Solutions. Modifications made to delivered charts are not supported; any changes made to delivered charts will be lost once the BackOffice Associates® Solutions is upgraded.

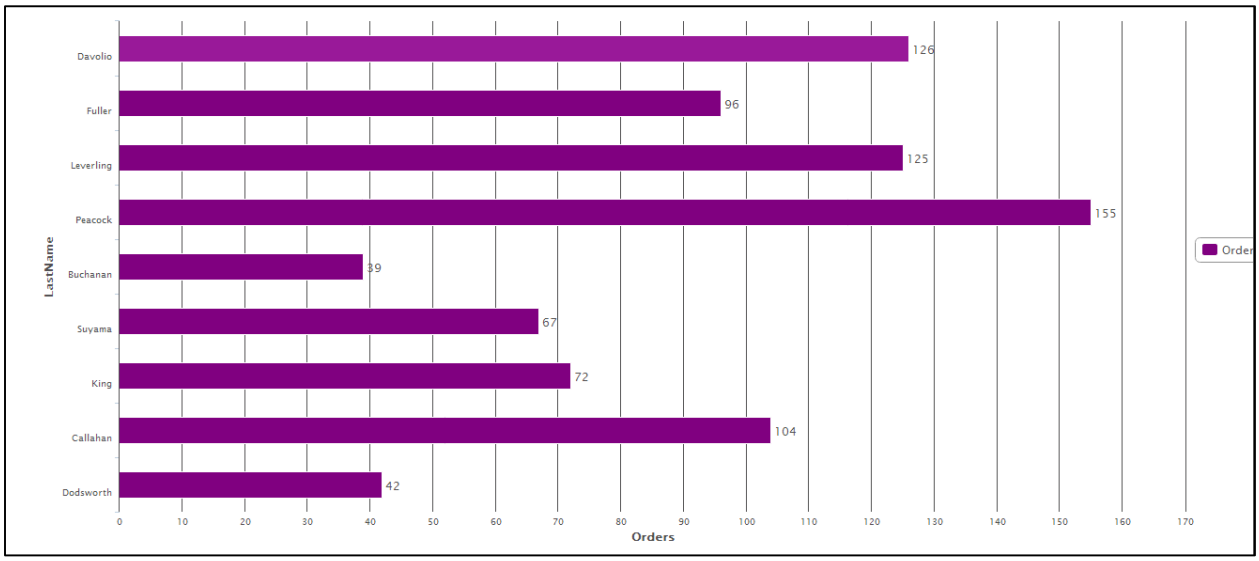
Chart Types

The following chart types are available:

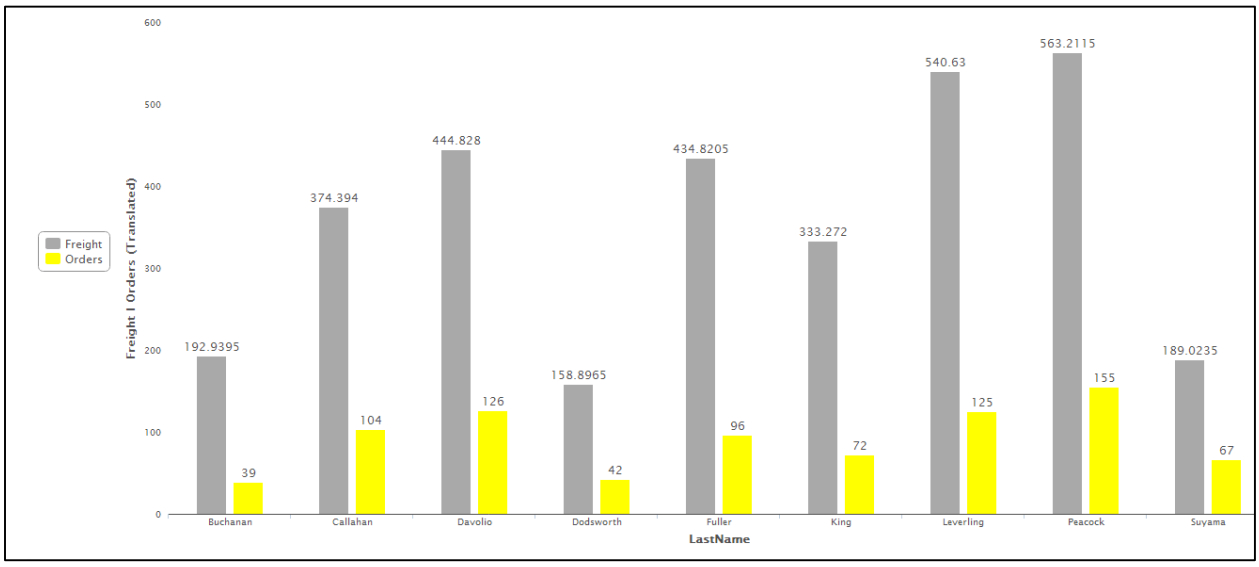
Area



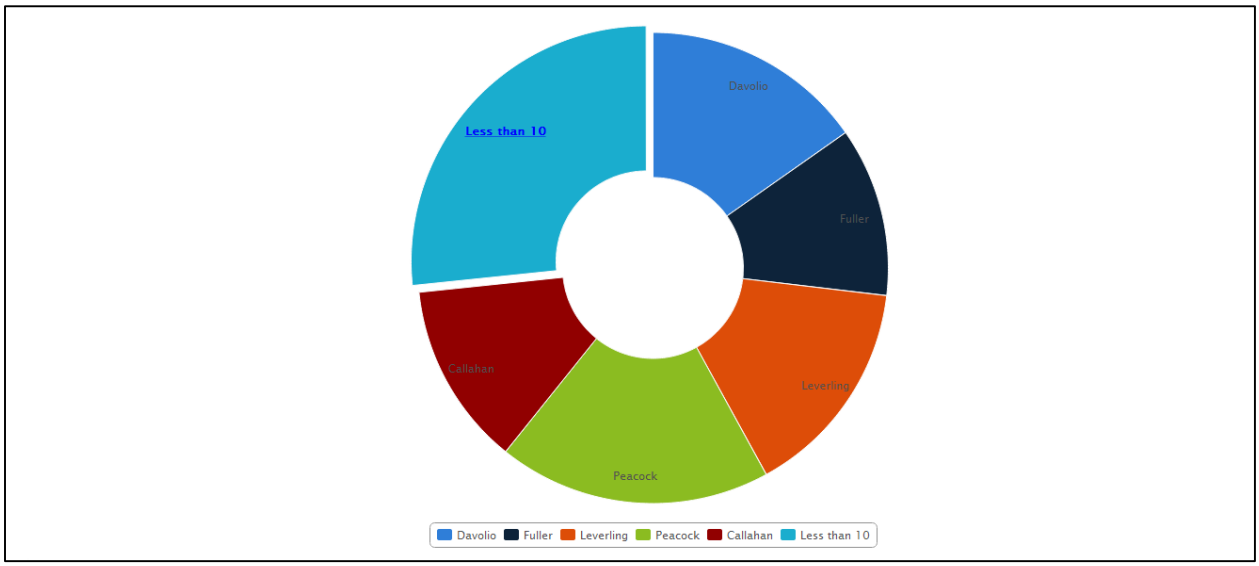
Bar



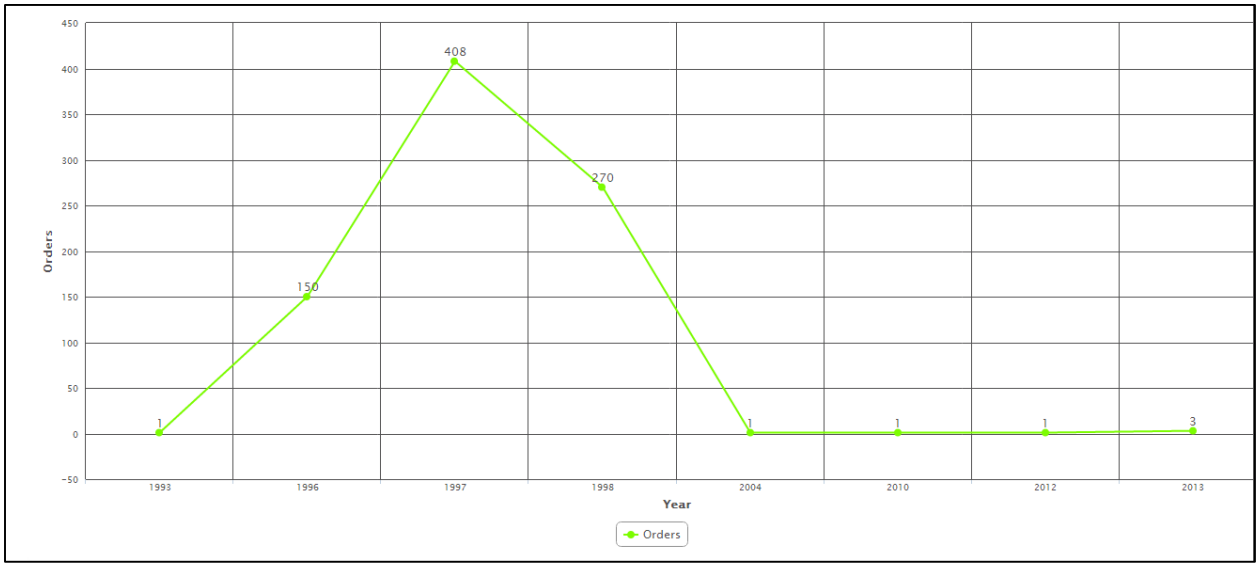
Column



Doughnut



Line



Create a Chart

NOTE: This manual does not contain steps for working in SQL.

NOTE: When making a chart page, DSP™ assumes the very first column in the view is the category and the second column in the view is the value and automatically defines the column properties. It is best practice to have the first column be the category and the second column be the value. Any other columns for binding or description, display or ordering can then be added.

To register the chart in DSP™:

1. Click the **Design Page** icon on the Site toolbar while on any page.
2. Select **Add Page**, the *Pages* page displays in a new window.
3. Enter a description of the chart in **DESCRIPTION** field.
4. Select **Chart** from **PAGE TYPE** list box.

NOTE: It is not necessary to select a table when adding a **Chart** Page Type.

5. Click the **Save** button, the *Vertical* View displays.
6. Select the name of the view created in SQL from **Report View** list box.
7. Select the type of chart from **Chart Type** list box.
8. Click **Save**.

NOTE: The page needs to be registered to the *Configuration* Menu so it can be accessed in the WebApp.

9. Click the **Menu Name** link in the **Horizontal Menu ID.**
10. Click **Add**.
11. Enter a number in **PRIORITY** field.
12. Select the chart page from **Link To Page ID** list box.

13. Click **Save**.

Modify Chart Properties

Chart properties can be modified to display the page in different ways.

Category and Value column properties are required for the chart to render. By default, two columns from the web*Chart view are selected as a Category and a Value column; review these column properties for accuracy.

NOTE: For Tabular chart type, the user has the ability to add a Text Box control to any column as a means to support page links. No other controls are supported.

Category Column Property

The Category column property allows the user to define which column in the web*Chart view corresponds to the X-axis. This column property can be used for most data types; the exception is binary data types. As with list boxes, a non-string key column can be used as the underlying value and a display value can replace what is shown.

There are two types of categories: Primary and Secondary. Each combination of Primary and Secondary Category values must be unique to ensure a proper rendering of the dataset.

A **Primary** Category column corresponds to the field whose data is rendered on the X-axis. If there is only one Category column defined (by definition, it must be Primary), each value row's primary column field is rendered on the X-axis. There must be one Primary Category column per non-tabular chart.

A **Secondary** Category, which is optional, allows for the definition of an implicit number of virtual Value columns for a dataset. If a Secondary column is configured, there must be exactly one Value column. Secondary Category columns allow for multiple data points to be rendered per Primary Category value. For example, if the user can configure *n* number of statuses and wants to create a chart to visualize an Object/Status Count, they could define a view that returns data as outlined in the following table:

Object	Status	Count
A	Status1	1
A	Status2	2
B	Status1	3
B	Status2	4

NOTE: All combinations of Object and Status are unique.

The configuration outlined in the above table ensures that as new Statuses are added, they are automatically taken into account when the chart renders by treating them as a Value column. When this chart is rendered, the X-axis will have two tick marks: A and B. Within those tick marks will be two series: Status1 and Status2. If this was a Bar chart, there would be two bars for A and two bars for B. If a Status is not included, it is assumed to be NULL. If a user wants non-existence to equate to zero, this must be written into the view logic.

Value Column Property

Value column properties are used to specify which field within the web*Chart view corresponds to a Y-axis value. This column is expected to be a numeric type because charts can only display numeric ranges. Multiple Value columns are supported when there is only one Category column defined. When the graph renders, there will be *n* number of series. For example, if a web*Chart view renders ["1/1/1990", 1, 2, 3], the first column would be defined as the Category column and the last three columns would be defined as Value columns. Value columns are recommended over a Secondary column when the amount of supplemental data is static.

When to use Value Columns Instead of a Secondary Column

It may seem intuitive to always use a Secondary category when each record represents a separate value because Secondary categories give the best angle to manipulate data. However, Secondary categories do not allow for column properties that the Value columns support, such as coloring individual lines or bars.

An example of when to use multiple Value columns instead of a Secondary column would be for data within a finite number of possible states. For instance, if there is a record that denotes Pass or Fail, there are only ever going to be two individual series (the Pass series and the Fail series). If a view is created to return the following data, "Green" can be the color for Pass and "Red" for Fail. This kind of manipulation can only be performed at the Value column level.

Modify a Chart

Chart properties can be modified to display the page in different ways, similar to other page types.

To modify the chart specific options:

1. Click the **Design Page** icon on the Site toolbar on the WebApp page that displays the chart.
2. Select **Design.**
3. Click **Vertical View** for the *Chart* page.
4. Click **Chart Properties** button.
5. Click **Edit**.
6. Select an option in the **Zoom Type** list box.

NOTE: This option sets whether the user can zoom in on the chart on either, or both, axis.

7. Select the location where the legend displays in the **Legend Location** list box.
8. Click the **Show Value on Hover** check box to display the numeric value when the user hovers a cursor over the chart.
9. Click **X Show Major Gridline** check box to display vertical grid lines on the chart.
10. Click **Y Show Major Gridline** check box to display horizontal grid lines on the chart.
11. Click **Save**.

Create a Dashboard

A Dashboard page can be created to display one or more charts. Dashboards are associated with a Layout to determine how the charts display. A Layout controls how and which charts display. DSP™ comes delivered with Layout Templates that can be copied but not modified. Custom Layouts can be created.

Charts assigned to a Dashboard page can be configured to drill down to more detailed information displayed on a page or another chart.

NOTE: A Layout that is currently assigned to a page cannot be modified. This is because if it were modified, the changes would cascade to all dashboards with potential undesired results. If a custom Layout needs to be modified, it must be unassigned from the page.

When creating a Dashboard page, users are able to create a layout directly from the *Pages* page, without leaving the page to navigate to the *Layout* page in order to create a new layout.

To create a Dashboard page:

1. Click the **Design Page** icon on the Site toolbar on any page.
2. Select **Add Page**, the *Pages* page displays in a new window.
3. Enter a description of the dashboard in **DESCRIPTION** field.
4. Select **Dashboard** from **PAGE TYPE** list box.

NOTE: It is not necessary to select a Table.

5. Click **Save**.
6. Click the **LayoutID** list box and select the + icon, the *Layout* page displays.
7. Enter a description of the dashboard in the **DESCRIPTION** field.
8. Click **Filter** to select a **Layout Template ID,** the *Layout Template ID* page displays.
9. Click **Save** next to the desired layout.
10. Click **Save** on the *Layouts* page.
11. Click **Layout Frames** for the newly added layout, the *Layout Frames* page displays in a new tab.
12. Click **Edit** for a Frame.
13. Select a chart name from the **DEFAULT PAGE ID** list box.
14. Click **Save**.

NOTE: Repeat the previous three steps and select the chart name to display in each frame of the dashboard.

15. Close the **Layout Frames** tab, and return to the *Pages* page.
16. Click **Edit**.
17. Select the dashboard from the **Layout ID** list box.
18. Click **Save**.
19. Close the *Vertical* View.

NOTE: The Dashboard must be registered to the *Navigation* pane in the WebApp.

20. Locate **Dashboard** on *Pages* page.
21. Click **Vertical View**.
22. To add this dashboard to the menu, click the **WebApp Name: Switchboard** link for **Horizontal Menu ID**.
23. Click Add.
24. Enter **a number** in the **PRIORITY** field.

NOTE: The Priority sets the order that page displays on the *Navigation* pane.

25. Select the created **Dashboard** page from **Link to Page ID** list box.
26. Click **Save**.

Create Chart Drill Down

Charts can be linked together or linked to any page type (chart, dynamic, static, dashboard, header detail, etc.). If a chart is linked to another page, when a chart section is clicked (a pie slice in a pie chart, a bar in a bar chart, etc.), a data page or another chart displays to show additional details.

To create a drill down to link a chart to any page type:

1. Access the chart page.
2. Click the **Design Page** icon in the Site toolbar.
3. Click **Column Properties**.
4. Click **Vertical View** for the column that allows a drill down (i.e., the column on the page that is drilled down to).
5. Click the **Edit** button.
6. Select the name of the page to drill down to from **Link to Page ID** list box.
7. Click **Save**.
8. Close the *Vertical* View.
9. Click the link for the column to drill down from.
10. Click the **Edit** button.

NOTE: Binding Fields are needed when the linking relationship cannot be inferred.

11. Enter the field name to bind to on the drill down in **Binding Field Names** field.
12. Enter the field in the **Shared Fields Names** field.

NOTE: Shared Fields are used to pass additional column values to the linked page.

13. Click **Save**.

Audit Trail & Electronic Signature

Any edits to a record can be tracked through an Audit Trail. When the Audit Trail is enabled on a table, an **Audit** icon displays next to each record. Clicking the icon opens a window that displays the date and time of the change, who changed the record, and the values before and after the change.

When Audit Trail is enabled, three tables are automatically created in the database: **XXXXAudit**, **XXXXAuditColumn** and **XXXXAuditProcedure**, where **XXXX** is the name of the table being audited. It is important to decide if the Audit tables are to reside in the same database as the application tables or in a different database. If using a separate database, register the database in the System Administration WebApp as a data source.

An electronic signature is a digital signature of a token representing the changes and is used in conjunction with the Audit Trail feature enabled. The token is computed by concatenating the user ID and the process token. The string is hashed using the MD4 algorithm. The resultant binary value is the signature. The value is stored in the **#Audit** table using Base 64 encoding.

This method provides a fast and simple digital signature implementation. It is important to note that the authenticity of an electronic signature depends on restricting access to the audit tables. If an unauthorized user gains write access to the audit tables, the signature can be forged or the audit trail could be altered.

Enable Audit Trail on a Page

There are two options to select when adding a new table:

- **Enable Auditing** – If checked, any additions or changes made to data in the table are maintained in the audit tables.
- **Audit Procedures** – If checked, a record of every stored procedure that runs against a record as part of a DSP Page Event is maintained in the audit tables.

To enable Audit Trail on a page:

1. Click **Admin** on the *Navigation* pane.
2. Select **Configuration > Data Sources** in the *Navigation* pane.
3. Click **Audit** for the WebApp's data source.
4. Click **Edit**.
5. Select the data source from **Audit Data Source ID** list box.

NOTE: The **Audit Data Source ID** is the database where the three audit tables are stored once the tables are built. This can be the same as the data source being audited; however, using a secondary database and data source can provide benefits in data management.

6. Click **Save**.

NOTE: The tables to be audited must be added.

7. Click **Tables**.
8. Select the table from **Table Name** list box.
9. Verify **Enable Auditing** check box is enabled.
10. Click **Save**.

These buttons become enabled on the *Horizontal* View.

- **Build Audit Tables button** – Creates the audit tables in the specified Data Source. Once the tables are built, the **Snapshot Data** button is enabled.
- **Snapshot Data button** – Creates a copy of the tables when the snapshot is taken. When a record is edited, the audit trail records both the before and the after values. However, the trail only shows values that are edited. Snapshot Data can be viewed as an insert for existing records. If the audit is enabled after the table has values in it, the trail has no way of telling where the data came from, so the snapshot is a way of verifying that some data existed prior to auditing.
- **Check Columns button** – Reports any differences between the columns in the table and the audit table. When the audit tables are built, all the columns from the table that are being audited are included. However, it is possible to have a case where the columns in the table and the audit tables do not match. There can be two reasons for the misalignment: 1) the designer deliberately removed some columns from the audit table because those values should not be audited or 2) the designer added columns to the table after the audit tables were build and forgot to manually update the audit tables.

11. Click **Build Audit Tables** button, a validation message displays.
12. Click the **OK** button.
13. Click **Snapshot** button to take a snapshot of the data in the table if necessary.

14. A validation message displays, click the **OK** button.

Enable Electronic Signature

When users save changes to an existing record, they may be prompted to sign the changes. This authorization occurs only if the users have changed a value in a column for which electronic signature is enabled.

To enable Electronic Signature:

1. Access the WebApp page.
2. Right-click the column heading requiring the electronic signature.
3. Select **Edit (Control Name)**.
4. Click **Advanced Properties** tab.
5. Click **Require Signature** check box.
6. Click **Save**.

Drop Audit Trail

Two users are required to drop the Audit Trail. The first user requests the drop and within 24 hours, the second user drops the audit tables.

To drop the Audit Trail:

1. Click **Admin** > **Data Sources** on the *Navigation* pane.
2. Click **Audit** for the WebApp's data source.
3. Click **Tables**
4. Click **Vertical View**

The first user clicks the **Request Drop** button. A message displays, confirming a request has been submitted. Both the **Request Drop** and the **Drop Audit Tables** buttons are inactive.

If the second user does not click the **Drop Audit Tables** button within 24 hours, the button becomes inactive and the process needs to be initiated again. Once the button is clicked, the audit is dropped. When Audit is dropped, the audit tables remain in the database. Once the Audit is dropped, the record must be deleted on the *Audit Tables* page.

NOTE: Dropping an Audit table is **not** workflow enabled; verbal communication between the two users involved is required.

Create a Workflow

Workflow is an email notification method called from within a WebApp where an email or multiple emails are sent to one or more recipients upon the execution of an event.

If a workflow requires a user to log in to DSP™ to perform a function, a link can be included in an email to navigate directly to the relevant page in the WebApp. The recipient of the workflow email must have DSP™ security to the page in the WebApp. If the user does not have security, a permissions error displays. If the user's password is expired, the settings page displays requiring a new password. DSP™ security overrides workflow events. If the site license, WebApp License or User ID expires or if the user is locked out, the link redirects to the login page.

To send workflows through DSP™, a mail server must be set up and the server must be registered in System Administration on the *Parameters* page. Email attachment size and the default for when a workflow link expires are also set up on the *Parameters* page. A workflow is used in situations where groups, users and fields outside the context of a particular page are needed, or when the decision to manually send an email are particularly complex.

The only two required workflow fields are Email To and Email Subject. Additional aspects of the email, such as the Email Body, Email From and Attachment should be included. Carbon Copy and Blind Carbon Copy can also be configured. Once the view is created, a workflow event must be created and the view registered in DSP™. All workflow view names must be prefixed with *web* and suffixed with *Flow*, by default.

Types of workflows are:

- **Standard** allows the email to navigate through the standard DSP™ authentication process which could be anonymous, basic or integrated authentication, depending on how the site is configured.
- **Semi-anonymous** allows DSP™ to look up the user's email address in the user's account in order to send the workflow email. When the recipient clicks the workflow link in the email, they are automatically logged in as the specified user.
- **Anonymous** allows the email to be sent to the email address specified in the "**To"** field of the workflow view. Upon clicking the link in the email, the user is automatically logged in as the anonymous user selected as long as a Workflow UserID is set up when setting up the event. Another approach is if a workflow is sent to an email address and the site supports Anonymous, the user is automatically logged in anonymously.

Workflow methods affect auditing. If the action required by the user is recorded by *boaAudit*, the following should be considered.

- If the workflow UserID is anonymous, the audit displays anonymous as the user.
- If the Site supports Anonymous but the workflow is sent to an email address, the audit displays that the email address is a user.
- If the Semi-Anonymous method is used or a method forcing the user to log in, audit displays the user ID.

To create a workflow:

1. Create a workflow page table and a view based on the table.
2. Create a page in DSP™ based on the Workflow view created in the database. A page must be created to manage all workflow content generated by the WebApp. The content of the email can be dynamic based on the columns on the page where the workflow is enabled.
3. Create a boaUser view to populate users in the **Email to** and **Email from** list boxes.
4. Add column properties to the workflow message page and create the workflow message.
5. Create the workflow Business Rule view.
6. Add a **Send Email** column to the *Horizontal* View of the page in SQL and add a button on the Page in DSP™.
7. Register the workflow Business Rule in DSP™ on an Event for the page.
8. Configure the workflow link

NOTE: Some applications have their own workflow management and configuration pages. Refer to the online help for those applications on how to configure the workflows (e.g. The "Configure Workflow Messages Overview" section of the help for dspGovern™).

Control Views

Run Time Control Views (RTCV) provide a way to define column control at run time based on the data. This view type defines standard DSP™ control statuses for columns on a page. These columns are based on data rather than a column property.

Control statuses are:

- **0** – Disabled
- **1** – Enabled
- **2** – Hidden

In addition, select BOA™ Reserved Columns can be defined in a RTCV, for example, boaAdd, boaEdit and boaDelete.

Types of RTCVs are:

- **Data Control View (DCV)** – Defines control based on what is known about the data on the page.
- **Page Control View (PCV)** – Defines control based on what is known about the page as a whole, including the parent page.
- **User Control View (UCV)** – Defines control based on what is known about the user.

When designing the control view hierarchy for a page, design the views in a way to minimize duplicating logic in multiple views.

Also, put control views at the appropriate level (e.g., a Data Control View conforms to the BOA reserved word functionality thereby allowing a boaUserID column to be added to the view). Using the boaUserID filter in a DCV, it could emulate the same functionality as a UCV. This strategy should not be implemented; rather, put user-centric logic in the UCV and control logic based on data in the DCV.

When a page loads, DSP™ uses the following approach to determine page and column control:

- WebApp Security Group
- Page Properties
- Page Column Properties
- UCV
- PCV
- DCV

In this hierarchy, the most restrictive prevails. Once the control status of a column has been limited, it cannot be made less strict. For example, if one of the controls on a page sets a column to hidden (2), the column is hidden even if another RTCV or property sets the column to enabled (1).

In terms of DSP™:

- Hidden always wins
- Disabled beats enabled
- If nothing hides or disables the column, the column is enabled.

Data Control Views

Data Control Views are created in SQL and define control based on what is known about the data on the page. For example, a DCV could be used to determine if a record can be edited or deleted or if buttons are disabled based on the state of the record.

Once created in SQL, the view must be registered to the page in DSP™ (**Admin > WebApps > Pages > Vertical View > Control Views tab**).

The following guidelines should be used when creating Data Control Views:

- Use the naming convention **webXXXDcv**, where **XXX** is the name of the table registered to the page.
- Include all key and criteria columns on the page to control.
- Include all column names for the technical names of the columns on the page that contain the control status values (0, 1 or 2).
- Prefix the column containing the control status with **boaCtl** when assigning a control status to a key column.

As an example, a user could disable the **Product History** and **Order ID** icons on the *Customers* page where the count on the linked pages is zero.

NOTE: This document does not provide steps for working in SQL.

Page Control View

Business requirements may require edit and delete capabilities to be disabled, depending on the state of the page as a whole. **Page Control Views** (PCV) are created in SQL to determine if records can be added, edited and deleted based on the state of the page.

Once created in SQL, the view needs to be registered to the page in DSP™ (**Admin > WebApps > Pages > Vertical View > Control Views tab**).

Guidelines for Creating Page Control Views

When creating Page Control Views:

- Use the naming convention **webXXXPcv**, where **XXX** is the table name on the calling page.
- Use a PCV when the control status determinations are to be made based on what is known of the page as a whole (e.g. binding criteria, shared criteria or boaPageID).
- Include all key and criteria columns on the page to control.
- Include all column names for the technical names of the columns on the page that contains the control status values (0, 1 or 2).
- Prefix the column containing the control status with **boaCtl** when assigning a control status to a key column.

User Control Views

Business rules may require specific functions to be disabled based on the logged in user. A *User Control* page can be created to manage the criteria for a User Control View. **User Control Views** (UCV) are created in SQL and determine if records can be added, edited, deleted or even viewed by a specified user. Once the views are created in SQL, they need to be registered to the page in DSP™ and the control criteria must be defined. A single User Control View can control multiple pages in a WebApp if designed correctly and registered to multiple pages.

NOTE: This document does not contain steps for working in SQL.

Guidelines for Creating User Control Views

When creating User Control Views (UCV):

- Use the naming convention **webXXXUcv**, where **XXX** is the name of the table assigned to the page minus the prefix.
  - If using one UCV across multiple tables, **XXX** is a meaningful name.

NOTE: When controlling multiple pages with a single User Control View, Enforce Strict Naming might need to be turned off depending on the naming convention on the view.

- Use a UCV when the control status determinations are to be made based on what is known about the User.
- Include the following columns:
  - **boaUserID** – Contains DSP™ User ID
  - **boaColumn** – Technical name of column on the page
  - **boaControlStatus** – Control Status: disabled (0), enabled (1) and hidden (2)

NOTE: Besides the DSP™ Reserved Words, no other columns are considered when applying the UCV control status settings to a page.

Setting up user control views involves the following steps:

- Create a User Control View in Management Studio.
- Register the view in DSP™.
- Define the control criteria on the new page.

The following steps are optional to allow management of the control view's functionality:

- Create a User Control table.
- Create a User Control Horizontal View to manage the criteria.
- Register the Page in DSP™ and assign to the Configuration Menu.
- Create list views for the Column Properties.
- Assign Column Properties.

Dynamic Views for Multiple Control Fields

This section contains an example of an alternative option to using a Data Control View to manage the way the fields display on a *Vertical* View. This method is to only be used when there are many control field options.

The steps below provide an example of creating Dynamic Views on a page where a selection in a list box entered on the *Horizontal* view controls the fields displayed on the *Vertical* View.

Dynamic views are used most effectively on vertical views. The best way to demonstrate their usefulness is through example.

The Orders table contains the following fields:

- Fields: **ID** for record ID
- **Submitted** as date of order

- **Type** being *Online* or *InStore*
- **StoreID** as ID for the store where the order was placed
- **ShipAddress** for address for shipping online orders

The four views would be:

webOrdersHor horizontal view	webOrders#Type#Ver registered vertical view	webOrders#Online#Ver substituted view	webOrders#InStore#Ver substituted view
ID	ID	ID	ID
Submitted	Submitted	Submitted	Submitted
Type	Type		
	StoreID		StoreID
	ShipAddress	ShipAddress	

When the user chooses the **Type** of *Online* on the horizontal view insert, and then they navigate to the vertical view, the DSP will look at the view webOrders#Type#Ver and replace the 'Type' between the two # marks with the value from the table. It will locate webOrders#*Online*#Ver and use that to determine the fields to display, allowing the user to enter the **ShipAddress** for the Order.

This enables one registered page to enable a few different vertical views tailored to the content that needs to be displayed.

Control Status Manipulation Overhead Minimization

Complex logic to determine whether to enable, disable, or hide page elements can often have the biggest negative impact on page performance. Since there are several different sets of control status manipulators, it can be difficult to pick the right one based on the context of the page.

Refer to *Control Views* for more information.

Control Status Fields

This feature has been deprecated due to the complexity it can add to views which are laden in control status fields, but in very simple cases, it can be utilized without bloating the data views. Control Status field is a field within the pages view (Horizontal, Vertical, Toolbar, etc.) which acts the same way as a Data Control View.

When used sparingly, it can result in pages that perform well and are still easy to understand.

Precedence

The simplest optimization that can be made to most pages is control view record reduction. UCVs and DCVs are usually the biggest offenders of unnecessarily large record sets.

The majority of the time a DCV is created, it selects the entire record set and does something like:

```
SELECT CASE WHEN [Field] = 1
THEN 0 --Disabled
ELSE 1 -- Enabled
END as [Enabled]
FROM [Table] WHERE [Field]
```

However, DCV's do not require every record to be present. If it's expected that more than often records will be DISABLED as opposed to ENABLED the DCV should change to:

```
SELECT 1 as [Disabled] FROM [Table] WHERE [Field] <> 1
```

where the Control Status is set to Enabled by default within the pages configuration.

This does several things:

- Reduces the record set site, improving performance
- Reduces the complexity of the view, improving performance
- Simplifies the view, making it easier to understand

Consolidation

Another problem is making sure you choose the most appropriate control mechanism to achieve the end goal. Most individual mechanisms can implement the same behavior in one way or the other. However, some may require much less computational logic.

An example of this is that a DCV can contain a boaUserID, which can allow for a cell to be controlled exclusively off of the boaUserID. For instance:

```
SELECT boaUserSel.[UserID] as boaUserID,
[Table].[KeyField],
CASE WHEN boaUserSel.[UserID] like 'a%' THEN 1 ELSE 0 END as [ControlledField]
FROM [Table] CROSS JOIN boaUserSel
Instead of a DCV, this should be a UCV, which would reduce the logic to
SELECT boaUserSel.[UserID] as boaUserID,
'ControlledField' as boaColumn,
CASE WHEN boaUserSel.[UserID] like 'a%' THEN 1 ELSE 0 END as [boaControlStatus]
FROM boaUserSel
```

This would result in the pages performance scaling linearly against the applications User count, as opposed to a multiplication of Users * Table Data.

Simplification

In the previous example, the following view was demonstrated

```
SELECT boaUserSel.[UserID] as boaUserID, 'ControlledField' as boaColumn, CASE
WHEN boaUserSel.[UserID] like 'a%' THEN 1 ELSE 0 END as [boaControlStatus] FROM
boaUserSel
```

Could be:

```
SELECT boaUserSel.[UserID] as boaUserID, 'ControlledField' as boaColumn, 1 as
[boaControlStatus] FROM boaUserSel where boaUserSel.[UserID] like 'a%'
```

The biggest performance gains most often come from SQL simplifications. Becoming a SQL expert will be the best way to ensure complex pages will perform at their best.

Last Updated on March 23, 2016