PeopleSoft

EnterpriseOne JDE5 Designing Web Applications PeopleBook

May 2002

EnterpriseOne JDE5
Designing Web Applications PeopleBook
SKU JDE5EDWA0502

Table of Contents

Overview

This document describes how to design and produce J.D. Edwards software applications for use on an HTML client with the J.D. Edwards Toolset. It provides information on similarities and differences in designing for HTML clients and Windows clients. It also provides valuable best practices guidelines, tools, and techniques that will help you design more usable J.D. Edwards Web applications. This guide is intended for use by application developers.

The guidelines in this document are suitable for use by J.D. Edwards employees, J.D. Edwards business partners, and J.D. Edwards clients.

See Also

- *J.D. Edwards Development Tools*
- *J.D. Edwards Development Standards: Application Design*
- *J.D. Edwards Portal*

User-Centered Design Guidelines

To create usable designs, you need to follow a user-centered design (UCD) technique. User-centered design means that you need to get early and frequent user interaction with the real user community to get feedback and input into the design of the application. Before beginning application development, you must clearly document and validate the goals and business objectives with users through conceptual and cognitive design reviews. All team members should understand who the application users are and what goals the users expect to achieve with the application.

User Analysis Checklist

- ❑ Define the users
- ❑ Define user and business goals
- ❑ Define user tasks
- ❑ Create use cases
- ❑ Perform usability evaluations

Define the Users

You must understand your audience before you can design an application that will enhance the user experience. Therefore, you must develop a user profile for each user type (end user, power user, and so forth) that you expect to use the application. You can gather information for a user profile from various sources such as Web surveys, interviews, task analysis, contextual inquires, focus groups, and market and competitive analysis. The following questions are the key questions that you should ask:

- Who are the targeted users? (For example, B2B e-collaborative suppliers and buyers)
- What are the targeted users' job titles? (For example, e-Procurement Managers, Buyers)
- What are the targeted users' levels of expertise? (For example, novice Web users, experts in the supply chain business)
- What are their unique needs? (For example, do they need to use this application remotely, when they are mobile, and so forth)
- What types of computers do the targeted users use? (For example, desktops, handheld devices, screen resolutions and sizes, Web browsers)
- What is the targeted users' work environment like? (For example, shop floor, cubicle, office)

Define User and Business Goals

You must understand how users move from goals to tasks to actions. Some user goals may be as simple as "doing time entry quickly and accurately." The business' goals might be to increase revenue or decrease the cost of providing support. Successful products are designed by understanding both user and business goals. You can define user goals through contextual inquiries, user interviews, and observations.

The following questions are the key questions that you should ask:

- What is the user's personal goal in using the applications? For example: Doing data entry as quickly as possible to meet incentive measures.
- What are the key goals for the targeted users to accomplish?
- What are the overall business goals? For example: To increase Daily Sales Order Entry.

Solutions for Different Users

In any given enterprise, only a fraction of the employees have access to the enterprise application suite. Most enterprises would like to achieve a better return on the technology investment and reap the benefits of consistent communication across the organization. However, they would like to meet these goals without equipping every desktop in the organization with a fully loaded desktop machine. Corporate intranets have achieved widespread popularity due, in part, to the low-cost, streamlined distribution of information that they provide. Outside of the constraints of traditional paper-based publishing, companies find they can dramatically improve the information flow within the organization.

However, most intranet-based information is relatively static. In contrast, information within the enterprise system tends to change with each new transaction. Companies would like to marry the ease of distribution afforded by the corporate Intranet with the real-time accuracy and processing capabilities of enterprise applications.

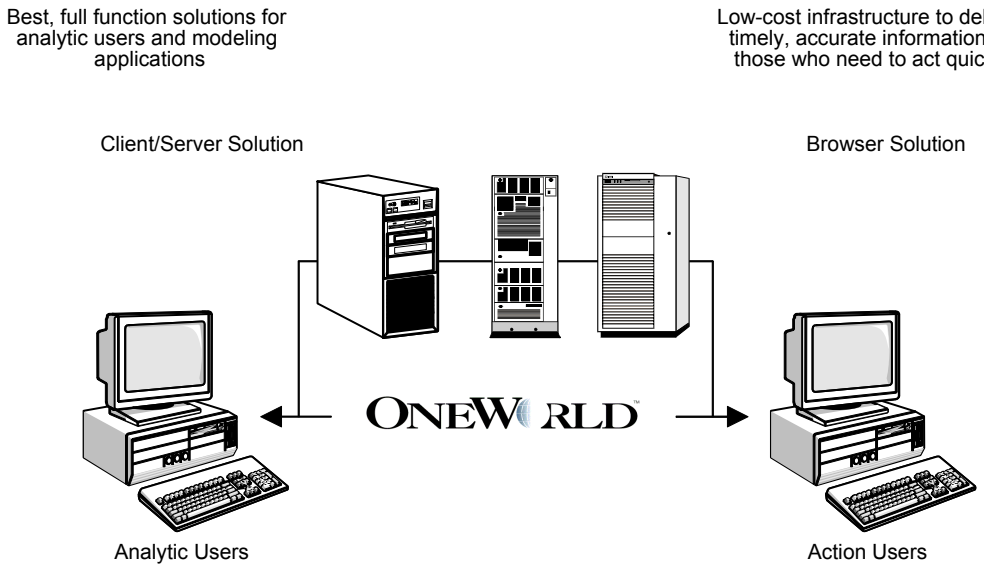
J.D. Edwards software allows businesses to leverage the corporate intranet to increase access to enterprise applications. Any browser-equipped device can provide a real-time window into the enterprise's information resources. Because of the low technology overhead, businesses can include more users in the information flow, thus achieving the following:

- Quicker dissemination of information to a broader corporate audience
- The knitting together of disparate business operations and distributed sites
- Better communication of goals, priorities, and strategies
- Improved decision making through the increased availability of information at every organizational level

The User Spectrum

Not every user in an organization uses enterprise applications in the same way. Some need continual access and the full suite of capabilities and desktop tools afforded by a robust client environment. Others require only the ability to review statuses and enter straightforward transactions. The spectrum of needs presupposes different technology requirements throughout the enterprise, depending on the user's skills and job requirements.

Users' requirements from enterprise applications vary depending on their role in the organization. Analytic users leverage multiple desktop tools to interpret and package enterprise information. Action users rely on quickly available, easily accessed information. The following graphic depicts the roles of the analytic and action users.



Analytic Users

At one end of the spectrum are users who gather, analyze, repackage, and distribute information to the rest of the enterprise—the knowledge workers of the organization. These users rely on a range of desktop tools, including enterprise applications, spreadsheets, and publishing tools, to bring together the various islands of information in the organization and integrate them into a meaningful whole.

J.D. Edwards client and server modes provide these users with the OLE-based, fully integrated desktop they need to maintain the information flow within the organization. The Windows client unites the processing tools of personal productivity applications and enterprise applications, as well as their respective data resources. Action users can then integrate this information, analyze it from various departmental perspectives, and repackage it in a context relevant to multiple functional areas, for example, requirements planning, executive decision making, marketing, purchasing, and so on.

Action Users

At the other end of the spectrum are those users who review information so that they can take a particular action. This class of users crosses organizational levels and, for example, might include order entry clerks, shop floor personnel, and executives. Action users often use the system to check order status and review item availability, for example, but they do not repackage and publish information for subsequent distribution throughout the enterprise. In browser mode, J.D. Edwards software offers action users the access they need with a point-and-click interface that minimizes training. At the same time, the low-overhead client enables the business to extend access to those action users previously out of the information flow due to the cost of equipping them with a fully loaded client workstation. The enterprise can push applications out to these users over a standard TCP/IP network to any browser-equipped device. Because no J.D. Edwards code resides on the client, the business also gains the benefits of centralized software maintenance and upgrades for an entire class of users.

Enterprise Applications Considerations

Like users, enterprise applications tend to fit better with either client/server or browser mode depending on the role the application plays in the information flow. Applications that provide the tools for analyzing and manipulating information from a variety of sources are well served by client/server implementations. J.D. Edwards client/server modes furnish the needed interactivity and graphical support, as well as the advantage of dedicated use of the client's processor. Specific examples include:

- Modeling and prototyping
- Budgeting and forecasting
- High-volume transactions requiring custom interfaces by transaction or customer type

Applications intended to communicate and distribute shared information and support standard transaction entries work well in browser mode. The types of applications listed below showcase information without tying up the client's resources, making them ideal candidates for J.D. Edwards browser mode:

- Information gathering and presentation
- Inquiry-based self-service applications
- Repetitive standard transactions

Even with the complementary fit between computing mode and application environment, business need often dictates that both modes be available to adapt to real-time changes in the business. In the J.D. Edwards environment, both modes are inherent in the architecture, to be deployed when and as needed. The enterprise can employ a mix of both modes, maintaining consistency in business data and processes. By offering both client/server and Web-based access to enterprise applications, J.D. Edwards software can meet the full spectrum of user needs within the enterprise.

Because client/server and browser modes exist in a single software solution, the business can implement J.D. Edwards software to match user needs or the information requirements of a given business process. With more users accessing the enterprise's information resources, the business realizes significant benefits:

Tighter integration of distributed business units	More users throughout the organization have access to a single, consistent source of information.
Streamlined processes	Traditional paper-based processes are more easily automated.
More efficient decision cycles	Decisions are not always pushed to the limited number of users who have access to enterprise applications.

Despite the differences between the two modes, enterprises are moving to combine client/server and browser solutions in a single computing solution, such as that provided with J.D. Edwards software. Where and how the enterprise deploys each mode depends on the business need and the built-in flexibility of the solution.

In considering the combined use of Internet and client/server technology in the enterprise, businesses face two fundamental challenges:

- They must identify those areas of the business best served by Web-based solutions and those areas best served by client/server solutions.
- They must implement solutions that accommodate both client/server and browser modes.

Define User Tasks

A user task analysis defines all the tasks that the users perform with the system to meet their goals. The following are some of the benefits of performing a task analysis:

- Generate ideas for new products
- Identify key features to include in products

- Design the user interface for products that are already identified and scoped
- Improve the usability of products already in production

To drive a user-centered design approach rather than a function-driven design approach requires an understanding of the user's tasks and the context in which those tasks will be performed. As with goals, a user task analysis can be gathered using contextual inquiries, interviews, and observations. The following questions are the key questions that you should ask:

- What tasks do the users need to perform using the system?
- What are the critical and important aspects of their tasks?
- In what sequence do they perform those tasks? In other words, what is the workflow of the task?
- What are their current environmental constraints and issues?
- How can their current work processes be improved by using the system?

Task Analysis Example

This example uses the E-procurement self-service application as a model for creating a task-based, user-centered design. The following is an example of the task analysis:

- Customer accesses the application to see customer alerts on changing market needs.
- Supplier responds to the alerts in a timely manner.
- Supplier receives "just in time" critical data on buyer/seller market—for example, shortages.
- Supplier searches for specific order request.
- Supplier browses current schedule for shipment tracking.

The example below lists all of the tasks a user might perform in the system as a customer or a supplier.

#	Customer Task	Supplier Task
1.	Change user profile	Change user profile
2.	Request for new user or password	Request for new user id or password
3.	Monitor daily alerts from enterprise (first)	Monitor daily alerts from enterprise
4.	View prioritized schedules and work	Views prioritized requests – for example, rejected orders
5.	View daily and delayed delivery schedules	Notify of daily and delayed shipment schedules
6.	Send request for quotes to supplier	View request for quotes from buyers
7.	View responses from supplier	Respond to request for quotes from buyers
8.	Search status of a specific order or item ordered from supplier	Search for status for a specific order or item for buyer
9.	Track outstanding shipment statuses	Track deliveries
10.	Request unfulfilled order shipment	Respond to shipment request

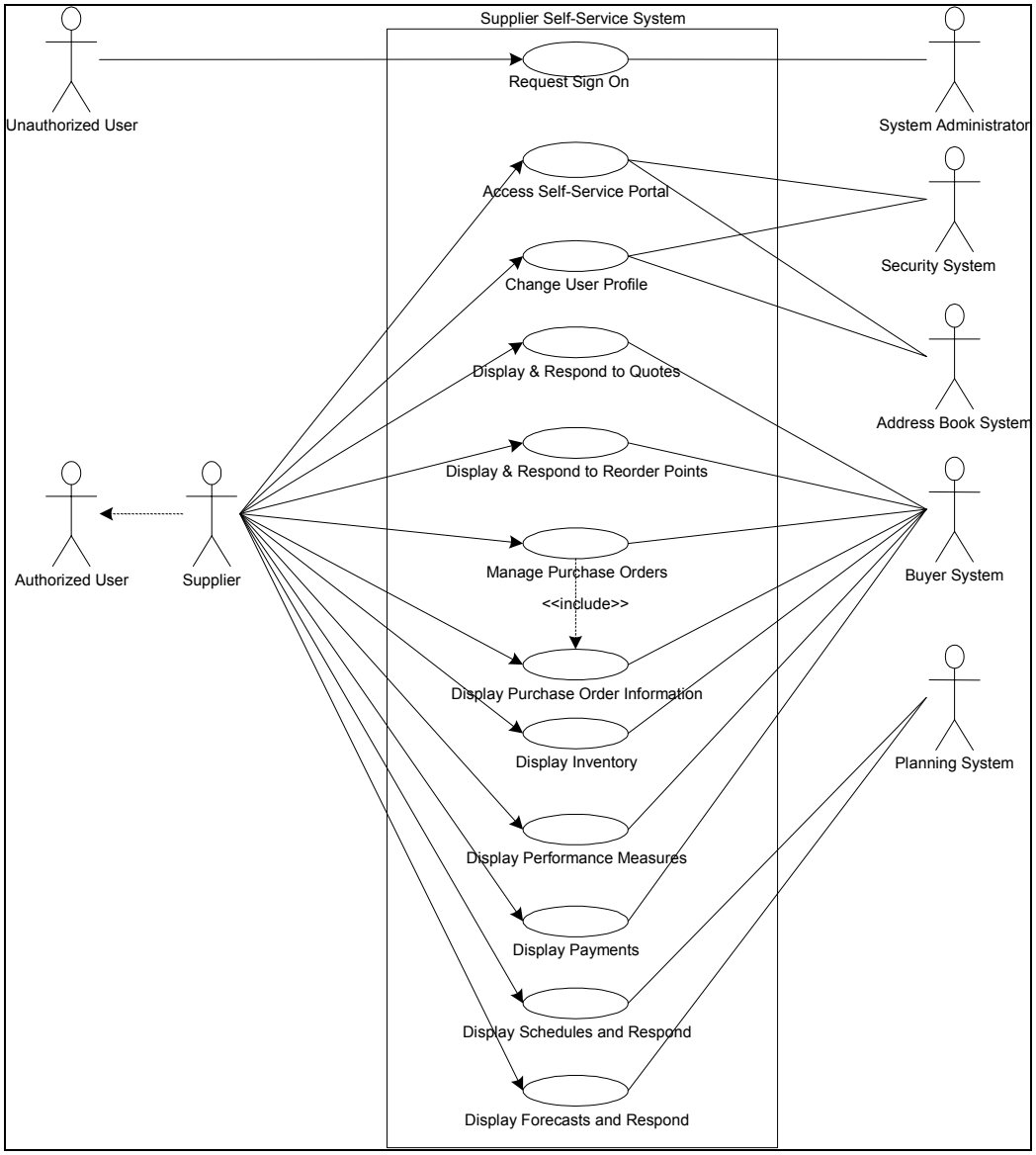
11.	Check outstanding invoices to be paid	Check overdue invoices
12.	Create sales orders	View sales orders
13.	View sales order status	Respond to sales order status
14.	Check supplier inventory	Check buyer inventory

Create Use Cases

A use case is a method for modeling user tasks. The purpose of use case design is to model user tasks in flowcharts to understand the navigational structure behind the Web application design. A use case describes the tasks the user wants the system to do, such as querying the status of an existing order. A use case approach helps define the boundaries of the system and avoids scope creep that can often happen without a clear model of use cases.

In the process of identifying and defining the actors and use cases, the designers define the application scope, such as what can be done and what cannot be done within the application.

Example of a Use Case



The following list describes some of the benefits of use cases:

- Provide an easily understood communication mechanism
- Make it difficult for requirements to fall through the cracks (when requirements are traced)
- Provide a concise summary of what the system should do at an abstract (low modification cost) level
- Use the language of the customer

Usability Engineering

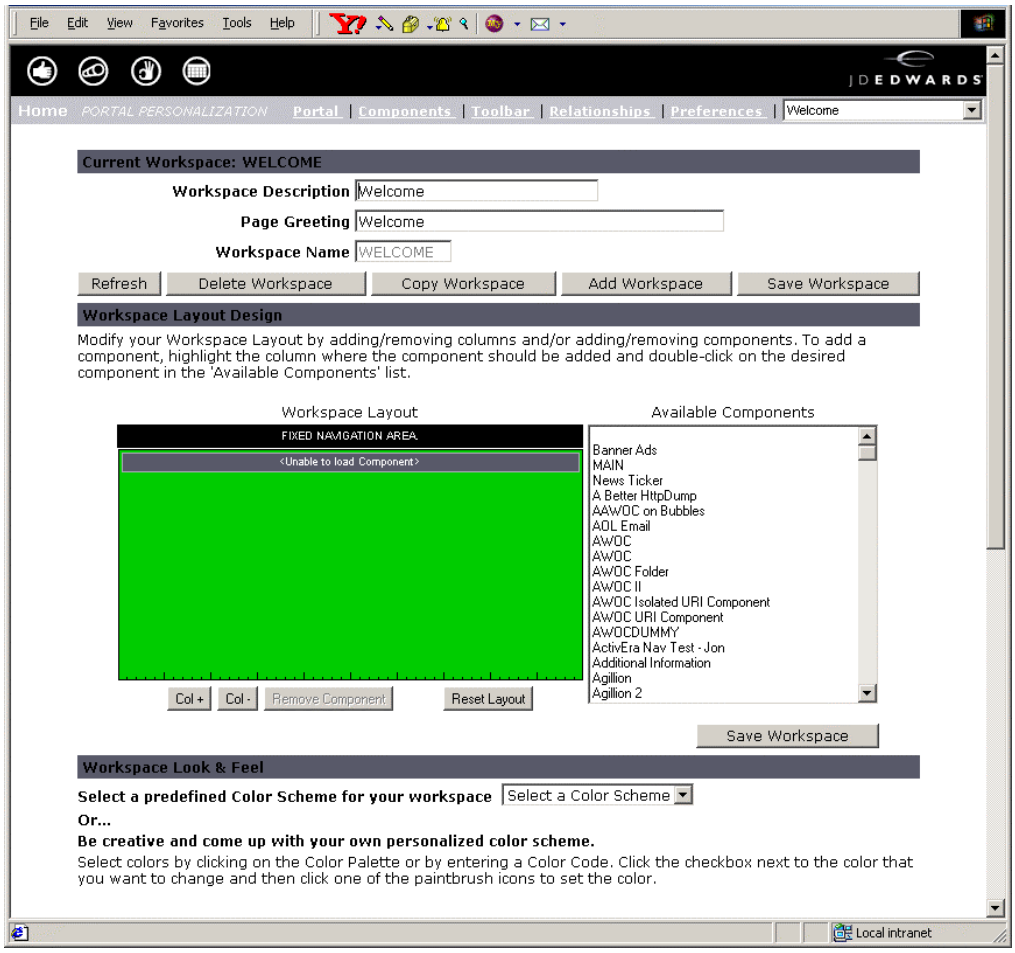
Designing J.D. Edwards software applications using user-centered design principles is not enough in and of itself. As application development progresses, you must validate and verify the designs with the end-users. Usability engineering includes conducting iterative usability

evaluations of the designs during the design cycle. The following list describes the benefits of incorporating user-centered design and usability engineering into the development cycle:

- Increase user efficiency and performance
- Increase operational efficiencies and productivity
- Improve user experience
- Improve customer satisfaction
- Increase sales and build customer loyalty
- Reduce overall development costs
- Reduce training and support costs

The case study shows how and why designers changed a form after performing usability evaluations.

The following illustration shows the form as the developers originally designed it.

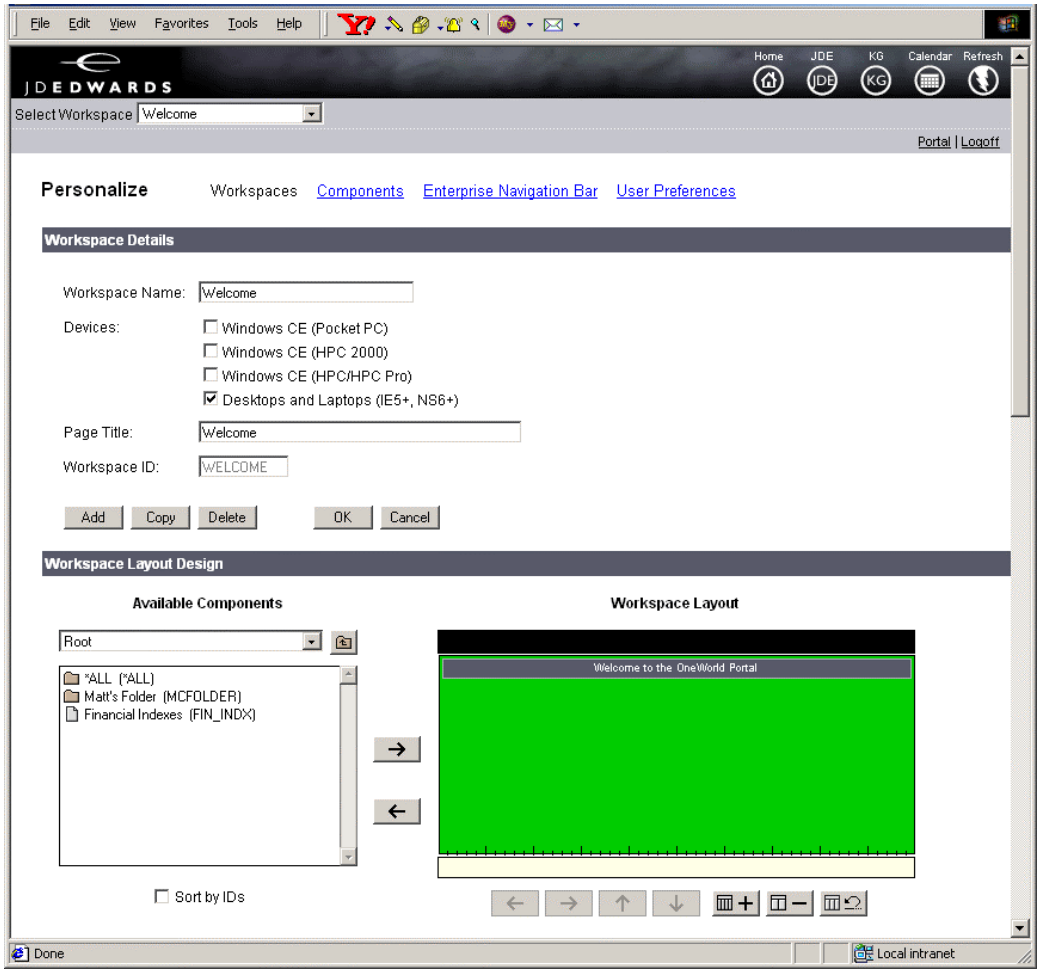


In response to the evaluation, designers sought to redesign the form to make it more usable. They decided to make the following changes:

- Emphasize the J.D. Edwards branding
- Use graphical icons that are inoffensive to international audiences
- Incorporate more meaningful labels and headings

- Allow more user-defined layout of screen elements based upon user task flow
- Reduce the amount of on-screen text by incorporating clear labels and headings
- Reduce clutter and visual noise
- Incorporate a better grouping of information

The following illustration shows their changes to the interface.



Information Architecture

The presentation of most of the J.D. Edwards Web applications is created using Form Design Aid (FDA).

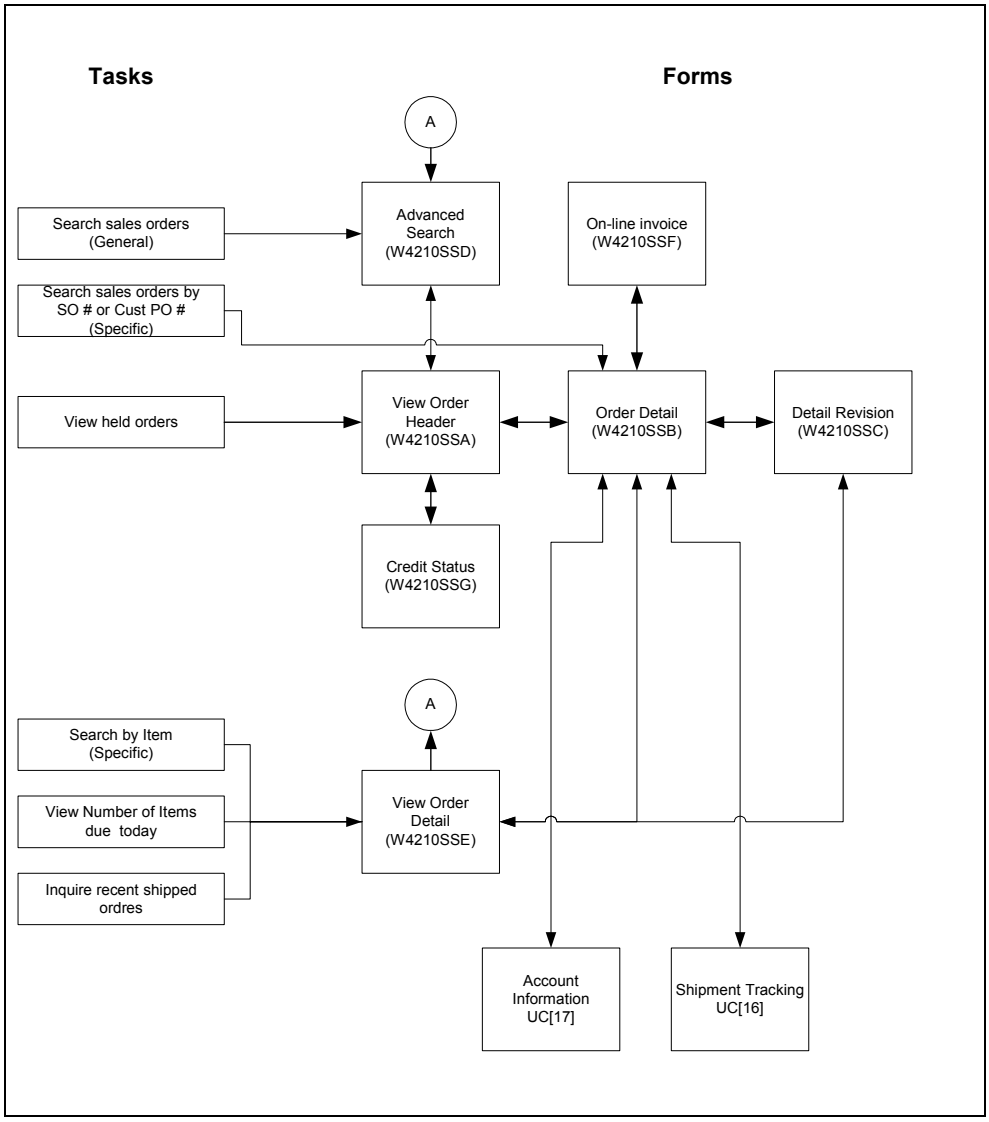
Create Wire Frames

One technique used in Web application design is the process of creating wire-frame diagrams to describe the initial Web screen design mock up. A wire-frame diagram provides a placeholder for your screen content and serves as a high-level architectural blueprint for the Web application. Incorporating wire frames into the development cycle helps speed up the process of creating the Web application.

The following list provides some of the benefits of use cases:

- Conceptualize the navigational flow from one screen to another
- Identify the contents of the Web application
- Provide an overview of the Web application
- Develop a conceptual prototype of the design
- Reveal problems in how tasks are distributed over the interaction spaces
- Provide a rough overview of how complex the system will be for users (overly long chains of transitions invite review for possible consolidation and simplification)
- Serve as a powerful tool for understanding the overall organization of software
- Reveal lurking problems in the existing designs and can help clarify possible solutions

Example of a Wire-frame Diagram



Basic Design Guidelines

Use the following guidelines to design Web applications while working within the constraints of the FDA tool:

- Organize the information based on the user's task needs. The layout must mirror the user's task flow.
- Display critical and frequently used information first.
- Avoid horizontal scrolling – users dislike scrolling horizontally.
- Use subheadings to group information into sections.
- Avoid using long, wordy sentences or phrases in instructions or labels. Instead, use action verbs (active voice) to describe the actions needed. The user interface should be self-evident and require minimal instructions.

Define the Navigation

Based on use cases and wire-frame diagrams, you define the navigation model of the Web application. Currently, the J.D. Edwards navigation scheme is sequential: users can traverse only in one direction by closing a form and returning to the previous form. However, Web applications require greater flexibility in designing the navigation scheme because they include the Back button. If the users cannot achieve their goals in the Web application, they are likely to click the Back button on the browser to exit an application.

Basic Navigation Guidelines

The following is a list of the 10 principles for designing a successful navigation for Web applications and Web sites. Navigation:

- Should be easily learned
- Should remain consistent across the application
- Should provide feedback to users
- Should appear in the user's context (for example, error messages)
- Should offer alternatives
- Should require economy of actions and time, meaning it should be efficient for the user
- Should provide clear visual messages
- Should provide clear and understandable labels
- Should be appropriate for the application
- Should support the user's goals and behavior

Navigation Schemes

One key challenge for Web designers is how to organize information in the Web forms. The navigation scheme must provide users quick and easy access to the information they need. Depending on the purpose of your design, you should use one of the following types of navigation schemes:

- Task-Based

  Navigation is based upon the user's task flow. This scheme is the most effective for J.D. Edwards Web applications. The J.D. Edwards Task Explorer is an example of a task-based navigation. Task-based navigation follows a browse path, but should not be too deep. Users should not have to drill down through four to six levels of hierarchy to reach their specific tasks.

  J.D. Edwards Self-Service applications use a task-based navigation model. Users navigate through the Enterprise Portal to access the workspaces where they can then navigate to specific tasks.

- Sequential

  The navigation is based upon one step at a time and is linear in structure. Use this type of navigation for a dialog-driven, tutorial style design.

- Informational

  Nonlinear design by which users can jump and skip pages using a hypertext model and information is organized by category. The user does not have a specific path to follow and requires more flexibility in browsing different information. This type of navigation scheme is used in sites such as Yahoo!. The J.D. Edwards Portal workspace also uses an informational navigation scheme.

Links

If you are designing an application with an HTML look-and-feel such as a self-service application with lots of customized HTML code, then you should use links as your navigation mechanism. If you are designing a Windows-based J.D. Edwards application with grids, tabs, and other J.D. Edwards controls, then it makes sense to use J.D. Edwards standard navigation schemes such as the J.D. Edwards toolbar and menu bar.

The J.D. Edwards software menu items such as Find, Select, OK, Cancel, Close are familiar to the end user and offer standard runtime processing. You should use these standard menu items instead of customized buttons whenever possible.

Use the following guidelines when designing navigational links in J.D. Edwards Web applications:

- Use active verbs to describe the task. For example, View Account Information.
- Avoid detouring the users to a different path when completing a task. For example, Browse Catalog.
- Provide a clearly marked and easy to see Close button to allow users to save and exit the application.
- Use hyperlinks to connect users to additional information. In a J.D. Edwards software grid, use hyperlinks instead of the two-step Select action.
- Make hyperlinks easy to understand. Avoid leaving the user guessing about what information will appear. For example, avoid providing a Travel hyperlink that displays information about travel agencies.
- Use a single word or a short phrase for links. At the most, use two to three words.
- Avoid making long sentences or paragraphs links.
- Avoid using an underline for labels that are not links. Make sure users can distinguish between a link and other words that are only being emphasized.
- Show **unvisited** links in blue with an underline. Exception: For the Portal navigation bar where there may be a conflict of color, use a contrasting color link. This contrasting color should be calculated so that the link does not become invisible.

  Show **visited** links in purple with an underline.

  Show **currently selected** links in red where the cursor is positioned.

- Provide an e-mail link in blue and underlined with correct e-mail labels. For example, E-mail helpdesk.

- Avoid using “click here” for more information. Place the link on the most relevant word in the sentence.

Correct Open Sales Orders

Incorrect Open Sales Orders: Click here!

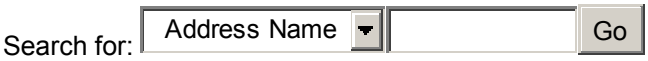
Define the Search Interface

For Web users, you must make the search function simple and fast to perform. Search is one of the most important user interface elements for Web applications. Usability studies indicate that only 64 percent of users find what they need on the Web because of poorly designed search applications. Use the guidelines in the following topics to design searches.

Simple Search

A simple search is required for the novice user who knows what she or he is looking for. It must be easy to use. Typically, simple search functions include minimal filtering options. In J.D. Edwards Web applications, a simple search function should be provided in the Portal home page or in each workspace. Use the following guidelines to design a simple search:

- The area to search should be clear. For example, J.D. Edwards Database or the World Wide Web.
- Search should be available from the home page or from the Portal.
- A simple search should not take the users to another page using a hyperlink.
- Provide a link to the Advanced Search function from Simple Search.



Advanced Search

If the component or screen space is limited, move the words “Search for:” above the control. For example:

Search for:

Advanced Search

- Use the Go button label when space is limited, such as in a Portal component. Otherwise, use the Search button label. See *Labels for Buttons, Fields, and Links* for more information.
- Map the Go button to the <Enter> key.

Advanced Search

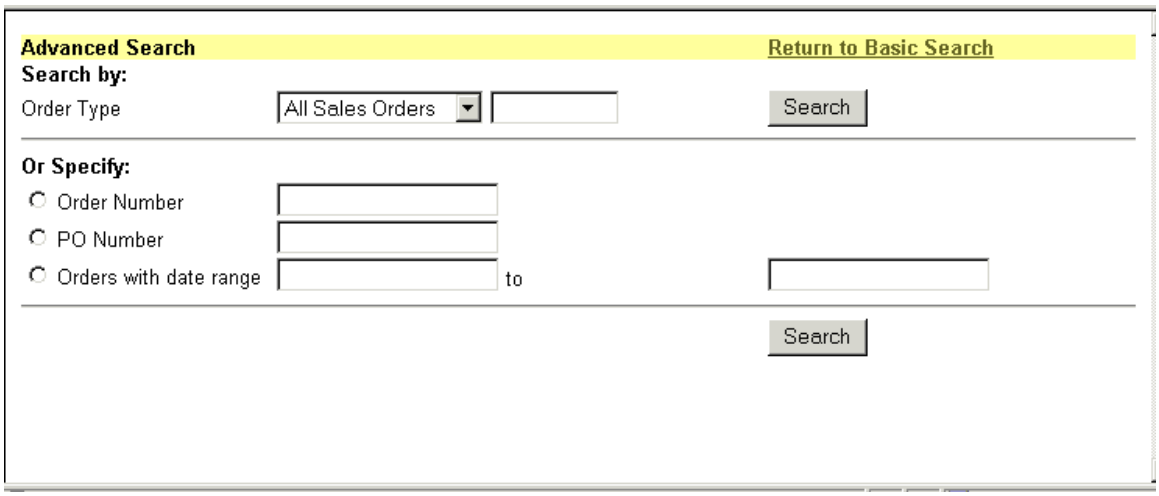
Advanced Search is used to find specific information. Users can use the advanced search to narrow the scope of their search by specifying one or more fields of information. Use the following guidelines to design an advanced search:

- For expert users, provide an advanced search option that replaces the current Query by Example (QBE) search function in J.D. Edwards software.
- Provide a hyperlink for the Advanced Search function on main forms and on Simple Search.
- Use the search default <contains> to replace the QBE function.

  Search for: Address Name | contains | Go

- Provide users with a way to navigate to the simple search from the Advanced Search page.
- Logically group the search criteria and clearly indicate the different options for searching.
- Allow users to refine the search on the same page that displays the results.
- Use Arial as the font for the text entry text box as it is a narrow font and allows users to enter more characters.

The following figure is an example of an advanced search form that meets these guidelines:



Displaying Search Results

Use the following guidelines to display search results:

- Display search results on the same page where the search was performed.
- Allow users to refine the search further if required; add a Refresh button for a refined search.
- The title of the search results section should be clearly labeled **Results** in bold.
- The search results should be shown in batches of 10 records.
- Search should allow users to jump back to a specific set of results section.
- Provide “breadcrumb” navigation back to the visited pages or links. For example, 1 2 3 4 .
- Provide <Next> and <Previous> links rather than <Forward> and <Back>.
- If no records are found, display a clearly visible “No Records found” message; for example, in the top left corner.

Search Error Recovery

Use the following guidelines to facilitate the user's ability to refine searches:

- Display the error message above the same form where the user entered the search so that the user can refine the search or correct the search parameters.
- Make all error messages clear, constructive, and specific.
- Allow users to begin a new search if no records are found.
- Offer the user tips to refine the search further.

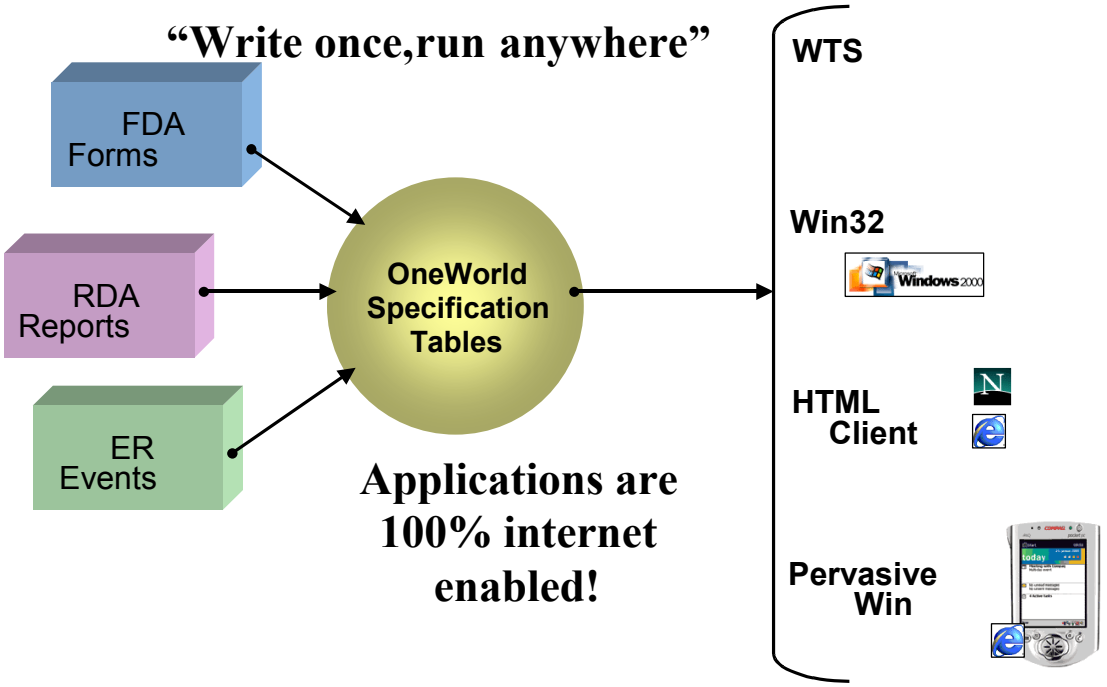
See *Error Handling and User Feedback* for more information about writing error messages.

Functional Differences between HTML and Windows

J.D. Edwards software offers three distinct interactive client experiences:

- Microsoft Windows Full Client
- Windows Terminal Server Client
- HTML Client

You can mix and match these clients in a J.D. Edwards software implementation to support a full range of enterprise-wide solutions. The common component that is used by all client solutions is the J.D. Edwards software specification. Because all client solutions access the same J.D. Edwards software specifications, developers can employ the Write Once, Run Everywhere strategy for all J.D. Edwards applications and reports.



Nevertheless, an HTML client functions somewhat differently from a Windows client. Anyone using the Form Design tool to produce HTML applications alone or in addition to Windows applications must understand these differences so that the same applications can be designed to function efficiently in both environments.

To create Web applications, you use the J.D. Edwards Development Toolset in the same manner as you would for Windows applications. To generate the application for the Web:

1. Save the application in Form Design Aid.
2. Generate the application to the Web server.
3. Attach the application to a menu and designate in which mode the application should run.
4. Execute the application.

Currently, the process of generating Web applications is relatively manual. J.D. Edwards intends to automate parts of this process in the future.

Appearance Differences

The Windows client runs as a Windows application. It uses windows, frames, dialog boxes, menus, toolbars, and other GUI components provided by the windows platform.

On the other hand, the HTML client runs in a Web browser. It uses the common controls provided by the HTML standards implemented by the browser. These common controls have a different look and feel than the windows GUI components.

In the future, HTML rendering will use absolute positioning most of the time. Consequently, the system will render a control at the exact position as it is defined in Form Design Aid (FDA) and with the exact size as it is designed in FDA. Because the Browser and Microsoft Windows Application display fonts differently, the same font might appear bigger or smaller in one platform than in the other. It is the application developer's responsibility to ensure that a control is big enough for both Windows and HTML clients.

The exception to the absolute positioning rule is the rendering of grid controls. To better use the browser's real estate, the HTML client usually renders the grid to be as wide as the form. In another words, the right boundary of the grid is stretched to the right boundary of the form. Controls to the right of the grid are rendered below the grid. Controls to the left side of the grid are rendered to the left.

J.D. Edwards recommends you do not put controls to the left or right of the grid control because the grid is usually the center of form operation.

Behavior Differences

The Windows client is field-based; for example, when the user tabs out of a field, the data validation routine is executed immediately. The user immediately sees the associated description and the error status of the data field.

On the other hand, an HTML client is page-based; for example, when the user tabs out of a field, that piece of data typically is not processed immediately. The events for that field are not launched immediately, either. Rather, these events are queued. When the whole page is sent to the JAS server, all the queued events are executed on the server and all the controls are processed in a fashion similar to the Windows client control processing. The process of sending the page to the server is called a post.

After the JAS server receives the page with user inputs, it processes all inputs in the sequence they were entered. During this process, the enterprise server is used to execute business functions, carry out database queries, and so forth. In our example, the data validation routines will be executed and the associated description will be loaded from the enterprise server. Then the processed page is sent back to the browser. This whole process,

from the post to the return to the browser, is called a round trip. At this point, the user can view the associated descriptions of all the data fields and the error status of each.

How the JAS Server Processes a Post

The HTML Client runtime engine maintains virtual images of the open forms in the system. When the JAS server receives a new page posted by the browser, it first finds the correct virtual client image for that user session. Then it processes all data on the form in the following order. The following sequence does not necessarily include all server processing. Instead, it outlines the sequence of events.

1. Execute the Control is entered event.
2. If the control has changed, copy data sent from the browser into form controls of the virtual client that is running on the JAS server.
3. Execute the following events if a control value has been changed:
    - Control is entered
    - Control is exited
    - Control is exited and changed inline
    - Control is exited and changed asynchronously
4. Validate and format the control.
5. Repeat steps 1 to 4 for all controls that have been changed.
6. Execute the following events and runtime logic if a QBE or grid cell value has been changed:
    - Column is entered
    - Column is exited
    - Column is exited and changed inline
    - Column is exited and changed asynchronously
    - The grid cell is validated and formatted
    - Row is entered
    - Row is exited
    - Row is exited and changed inline
    - Row is exited and changed asynchronously
7. Repeat steps 6 to 8 for all grid cells and rows that have been changed.
8. Execute Button Clicked events such as OK Button Clicked that caused this page to be posted.

Actions that Trigger a Post and Input Processing

The following user actions will trigger the browser to post the Web server:

- Clicking a toolbar button such as OK or Cancel
- Clicking a form exit, row exit, or tools exit

- Clicking a radio button or check box that has event rules in the Selection Changed event
- Clicking a push button or bitmap
- Clicking a hyperlink such as clickable static text, a clickable grid cell, or clickable text block control
- Clicking visual assist for a text field or a grid cell
- Clicking the binder clip image of a grid row
- Switching tab pages
- Clicking Previous/Next link for the grid
- Clicking edit button on single line grid
- Clicking on an editable grid for the first time
- Operations on tree control or the tree in parent/child control such as:
    - Expanding a tree node
    - Collapsing a tree node
    - Dragging and dropping a tree node
    - Double-clicking a leaf node if it has event rules in the Double-Click Leaf Node event.
- Clicking the Tools/Refresh menu

The input processor handles inputs that trigger a post. These input processor events are separate from ER events; in fact, input process events do not necessarily require that ER events be attached. The input processor resolves posted inputs in a specific order, and it looks for specific cues to determine what changes occurred. The input processor executes events as it detects them; consequently, some programmatic changes to the grid can be interpreted as if the user had keyed changes. The following list shows the order in which the input processor handles events, how it detects changes, and what actions it triggers:

1. If the Shortcut or Back button was clicked, perform that action and exit (perform no further processing).
2. Process QBE events, if appropriate.
3. With the exception of the grid, PC events, and the tree, process the control events that the user visited or keyed in to. The system uses JavaScript to detect these conditions.
    - Text fields – The processor sends "entered" and "exited" events to all controls that the user visited. It also sends "changed" events to the controls the user keyed into.
    - Check box – The processor looks for a changed state and sends the "changed" event if appropriate.
    - Radio buttons – The processor sends a “clicked” event only (the control itself detects changes).
    - Combo box – The processor checks the selection and sends an event if appropriate.
4. Process grid events or parent-child events.
    - Grid events – JavaScript does not indicate where the user has visited, so it sends the entire contents of the grid. The processor compares the previous values in

the grid to the current values. If any have changed, it sends a "cell edit" event to the grid. This event launches grid focus, row entered, column entered, and later column exited/change and row exited/changed events. When executed, these events occur in sequence, top-to-bottom and left-to-right. The system uses this same logic for silent posts except that only a few rows are sent at a time.

The processor also selects (check box) rows that the user has checked.

- Parent Child events – The system can send one of these mutually exclusive events for the PC control:
  - Cancel drag drop (started to drag, but next action was not drop)
  - Row selections
  - Next/Prev page
  - Expand/Collapse node
  - Drag
  - Move
  - Copy

5. Process tree events – The system can send one of these mutually exclusive events for the tree control:
   - Row selections
   - Next/Prev page
   - Expand/Collapse node
   - Double-click node
6. Process click events – The system can send one of these mutually exclusive events:
   - Customize Grid
   - Export/Import
   - Refresh
   - Load Form (opening from link or menu)
   - Grid Prev/Next page
   - Form/Row/Report/View Menu Exit
   - Clicked Hyper Link column on grid
   - Tab Page Clicked
   - Grid Tab Clicked
   - Grid Next Line (single line edit)
   - QBE/Cell Visual Assist
   - Row double-clicked (select)
   - Row Header double-clicked (MO)
   - Embedded MO event
   - Check attachments
   - Textblock hyperlink clicked
7. Button clicked – These events include all the standard buttons.

Actions that Do Not Trigger a Post

The following events will not trigger a post automatically and, therefore, will not result in a round trip:

- Tabbing out of a text field
- Tabbing out of a grid cell
- Tabbing out of a QBE cell
- Selecting a tree node in a tree

Handling Round Trips

A round trip has an adverse impact on performance because it uses network bandwidth. Too many round trips reduce the scalability and performance of the system. A round trip also impacts user experience. When a round trip occurs, the user must wait for the server to process the page and return it to the browser. When the page does come back, it flashes on the browser. Although round trips can negatively impact performance, they are acceptable (even expected) when data accuracy is critical.

Because of these impacts, a key design principle for Web-based application is to avoid unnecessary round trips. However, after carefully considering the performance impact of the post and eliminating alternative solutions, if an application still requires some events to occur immediately, you can use the following process to do so.

Set the event property HTML Post for the events that you want to trigger a post. This property is available for the following events:

- Tabbing out of text editor: Control Is Exited, Control Is Exited and Changed Inline, Control Is Exited and Changed Asynch
- Tabbing out of grid cell editor: Grid Column is Exited, Grid Column Is Exited and Changed Inline, Grid Column is Exited and Changed Asynch
- Selecting a tree node: Tree Node Is Selected, Tree Node Level Is Changed

The above solution is the recommended solution for J.D. Edwards applications. In Xe, if you have already set the event property on desired events, you should also uncheck the form property HTML Auto Refresh so that the generator does not need to scan your event rules.

Grid Selection

In a Windows client, the user selects a grid row by clicking on the row. The client highlights the selected row in response. In a multiple-selection grid, when the user selects a different row, the previously selected row is automatically unselected unless the user holds down the SHIFT or CTRL key.

In an HTML client, when a grid row is selected it is not highlighted. For single-selection grids, the HTML client shows a radio button beside each grid row. The user clicks on the radio buttons to select different grid rows. For multiple-selection grids, the HTML client shows a check box beside each grid row instead.

Note

The HTML client does not automatically unselect the previous grid row if a new row is selected in multiple-selection grids.

Grid Row Attachments

In a Windows client, the user can double-click the row header to bring up the attachment window for the grid row. In an HTML client, the user clicks the binder clip image on the row header instead.

In Windows, by default the grid row does not display the binder clip image, even if it has attachments. When the user hovers the mouse over the row header, the runtime engine checks to see if an attachment exists for that row. If an attachment exists, the engine displays the binder clip. In an HTML client, a plus sign is displayed for all grid rows initially, regardless of whether the row has an attachment. When the user clicks the binder clip image in the row header, the runtime engine checks to see if an attachment exists for that row. If an attachment exists for a given row, the system changes the plus sign to a binder clip image.

Tab Sequence

The HTML client supports tab sequences, with slight differences from the Windows client. Because the HTML client is hosted within a browser, you cannot negate the browser's tabbing sequence. The browser starts tabbing in the page, but after you reach the end of the tab sequence it does not wrap back to the first control in the J.D. Edwards form. Instead, the browser makes a stop in its own URL Address box outside of the J.D. Edwards form.

Grid Functions

The Windows client supports Zoom, Maximize/Restore, Charting, and Print on the grid. These functions are unavailable in the HTML client.

Type-Ahead Edit

The HTML client uses the type-ahead feature provided by the browser. Generally, the type-ahead features of most browsers are not as advanced as the type-ahead feature in the Windows client.

Client-Only Business Functions

Client-only business functions are not supported on the Web. All business functions are mapped to the server. A client/server business function or server-only business function should not call client-only business functions.

There is one exception. If an NER is marked client-only solely because it has a form interconnect (for example, it does not access the local hard disk or make Windows-only calls), the NER will function properly on the Web. You should keep these NERs as client-only. When these NERs are generated, J.D. Edwards generates Java classes instead of C code for them. In this context, these types of NERs are called Interpretive NERs.

In summary, J.D. Edwards automatically uses Interpretive NERs when a client-only NER is executed on a server. NERs that access local hard disks or make windows calls will not work; however, NERs that make form interconnects will. Consider the following guidelines for Interpretive NERs:

- Interpretive NER can only be called from application ER or from another Interpretive NER.

- Interpretive NER should not be called from UBEs, client/server NERs, or server-only NERs.
- Interpretive NER should not be called from asynchronous events.
- The NER must be designated client-only if it includes form interconnects. J.D. Edwards uses the client-only designation to determine if Interpretive NER should be used.

Modeless Processing

The HTML client does not currently support modeless processing because the Web browser is by nature a modal mode; that is, you can have only one active window at a time. All modeless form interconnects are executed as modal interconnects.

Modeless processing in the HTML client is planned for a future release of the tool.

Asynchronous Events and Asynchronous Business Functions

In OneWorld Xe, the system administrator can enable or disable asynchronous event processing by modifying settings in HTMLClient.ini. By default, asynchronous event processing is disabled, which means that Control Is Exited And Changed-Asynch, Column Is Exited And Changed-Asynch, and Row Is Exited And Changed-Asynch are executed as inline events. They are executed after Control Is Exited And Changed-Inline, Column Is Exited and Changed-Inline, and Row Is Exited And Changed-Inline.

With the same file, the system administrator can also enable or disable asynchronous business functions. The default setting is on, which means that all asynchronous business functions are executed asynchronously.

Multi-Line Edit and Silent Post

By default, the HTML client only displays one editable grid line. You can choose to display multiple edit lines for specific applications. This feature is recommended for high-volume data entry applications. The multi-line edit property is a form level property. You can change it through Form Design.

When a form is set for multi-line edit, the J.D. Edwards runtime engine also provides a silent post feature. As the user enters data in the grid, the first two lines that are entered are posted automatically to the JAS server to be processed. The server executes events such as *Grid Row is Exited and Changed-Inline* and *Grid Row is Exited and Changed-Asynch*. Data calculated by the Web server is returned to the Web browser automatically. For example, the associated descriptions are updated to the first two grid lines. At the same time, the user can continue to enter data lines. This feature greatly improves the performance of data entry applications.

Note

The Row is Exited And Changed-Inline event is not executed before the user enters the next row.

Platform Compatibility

When you design Web applications, you should be aware of Web browser compatibility issues. Every browser interprets HTML tags a little differently. Table, form, link, and alignment tags work differently in each browser. Use the following guidelines when designing your Web applications:

- Design for the appropriate Web browser version. J.D. Edwards Web applications currently run on Internet Explorer 5.5 and up and Netscape 6.2 and up.
- Avoid using tags supported by only one browser.
- Test designs on multiple browsers and platforms before release.

You should select the screen resolution appropriate to the needs of your target user populations. For OneWorld Xe, the default form guide size in Form Design Aid is 640 x 480 pixels. Most users with 15-inch monitors are using 800 x 600 screen resolution, and users with 17-inch monitors are using 1024 x 768 screen resolution. B2B users in the manufacturing and distribution industries often use 15-inch monitors with 800 x 600 screen resolution in their facilities.

If you are designing application for Windows CE or pocket PC, you should select the 240 x 320 pixel option in the Forms Guide menu.

Web Form Design

The input form in Web applications is referred to as a Web form. The organization of information in a Web form must conform to the logical task sequences defined in the use cases to ensure that users achieve their goals when using the Web form. Use the following guidelines to design Web forms.

Visual Grouping

To create groupings on a Web page, use:

- Section headings (SectionHeading class)
- Subheadings (SubHeading class)
- Light horizontal rules (HR1 class)
- White space

Group form elements as described below:

- Avoid creating long user input forms. Do not make a form longer than two vertical screens (assume the user is using a 14-inch monitor).
- Place important information first.
- Provide clear groupings of sections broken down into tasks or logical groupings; for example, Dates.
- Group the action buttons, which control actions in the form header, near the form header (top right corner or within the form header).

- Group the action buttons, which control the results or grid section, closer to the grid (top right or below the grid).
- Provide clear headings that are task based at the top left of the section.

Form Controls

Use the following guidelines when designing form controls:

- Avoid pull-down lists of only one or two items. Display these items using radio buttons instead.
- Use the combo box for discrete choices of up to 12 items that are predefined and can be translated. For lists of greater than 12 choices, use the Visual Assist function instead.
- Avoid using radio buttons for m
- Avoid using group box controls; use horizontal rules above and below the information instead.
- If you are designing a new application that is dramatically different from traditional J.D. Edwards applications, then do not use the J.D. Edwards tool bar. An example of an application that is different from traditional J.D. Edwards applications is the employee self-service application. That application does not contain a grid, it does not use any default J.D. Edwards commands, and it does not use any default runtime processing for functions such as Add and Find.

Note

J.D. Edwards recommends that application developers use the Java API com.jdedwards.base.webgui, which determines how controls will appear: in a Pocket PC, desktop, or other device.

Text Justification

Use the following standards when designing text blocks:

- Left align all text blocks.

  Center- and right-aligned text blocks are harder to read. The ragged margins created by center- and right-aligned text make eye scanning from left to right difficult.

- Left align the titles and he
- Left align text labels, text entry fields, and field labels above columns.

Button Placement

Use the following standards when placing buttons on a form:

- Do not create custom buttons that simply wrap J.D. Edwards standard buttons. For example, if you want to provide a delete function, use the default Delete button under the File menu. Do not create your own Delete Item button that simply calls the Default button.

- Place action buttons at the bottom of the form below all entry fields.
- Group similar action buttons together. For example, place the <<Previous and Next>> buttons together on the bottom right side of the form, and place the Add Contact and Cancel buttons on the bottom left side of the form. The Cancel button must always appear to the right of the action buttons.

Correct

Incorrect

- For forms that contain the OK button, place the OK and Cancel buttons in the following order:

  [OK] [Cancel]

Labels for Buttons, Fields, and Links

Use the following guidelines when labeling objects on a form. See the *Typography* section for more information about button and field labels.

- Use a consistent style for naming field labels, buttons, links, and so on. Avoid using a mixture of Order #, Order Num, Order Number.
- Avoid abbreviating labels in grid columns or in form areas. If you must abbreviate a label, refer to the accepted list of abbreviations found in the *J.D. Edwards Development Standards: Application Design Guide*.
- Use a label to describe the button's action and aim for the shortest possible label; one word is best. If you must use more than one word, use an action word (or verb) as the first word, such as "View Record."
- Use initial caps in button labels. Avoid using all caps.
- Use standard button labels, but consider your application style first. J.D. Edwards standard hypermenu items are most familiar to existing customers. The following table lists naming conventions for the most common button labels and explains each one.

Button Label	Meaning
Back	W
Go	Use in conjunction with the search function. Use Go as an alternative to the Search label when space is limited, such as in a component. See *Search Function Guidelines.*

Next	Sends users to the next sequential page in a Web site. Use the Next label to control the user's navigation through the application.
Previous	Sends users to the previous sequential page in a Web site. Use the Previous label to control the user's navigation through the application.
Refresh	Clears the screen so the user can enter new parameters. Use the Refresh label to allow users to perform a new search.

- Avoid using colons, semicolons, commas, dashes, slashes, and backslashes in labels and headings.
- Use the user's terminology for labeling information, not what is familiar to Development.

  Correct Update or Submit

  Incorrect [Submit to Database for Updates]

- Keep field label terminology consistent.

  Correct Quantity Ordered, Quantity Open, Quantity Received

  Incorrect Ordered Quantity, Open Qty, Quantity Received

- Label the action buttons clearly.

  Correct Login

  Incorrect Click Here to Log Into Member Services

- Always include text labels with icons unless the icon is easily recognizable, such as the Mail icon.
- Strive to accommodate international users when labeling fields, headings, and buttons.
  - Static text and text block controls have text defined in the properties and are translatable. Text strings in Event Rules are NOT translatable; therefore, use static text and text blocks whenever possible.
  - Use the 80/20 rule for text spacing. In other words, no text item should exceed 80 percent of the space allotted to it. Ensure that all field sizes allow for text expansion of up to 20 percent.
  - Avoid hard coding Text Block Control and Static Text Control text in Event Rules. Instead, create a text variable called "Hard-Coded Text." For example:

    ```
    Set Control Text(FC_st_AN8, TV Hard-Coded Text)
    ```

- Left align text labels, text entry fields, and field labels above columns.

Form Titles and Headings

Use the following guidelines when naming forms:

- Each Web form or Web page must have a clear and meaningful title that defines the task. For example, Check Inventory or Edit Shipment Address.

  Correct Edit Shipment Address, Edit Order Headers

  Incorrect Work With Shipment Address, Work With Order Headers

- Titles and Headings above the left-aligned body of a form should also be left aligned.

Typography

Typography affects a variety of usability issues such as screen resolution, platform support, browser differences, and online readability. Good typography depends on the visual contrast between the use of different typefaces and fonts of section headings, titles, and the surrounding white space. Use the following guidelines when creating forms:

- Use sans serif typeface and use the Font-Family tag to ensure that it is supported in multiple browsers.
- Arial typeface is the recommended typeface. Use typeface alternatives within the style sheet to support other platforms. For example:

  ```
  Body {Font-Family: Arial, Verdana, Sans-Serif;}
  ```

  The Sans-Serif typeface specification allows the default class of typeface for sans-serif to be displayed in the event the browser does not support Arial or Verdana.

- Use bold style only for section headings and the title of the form or page. Do not use bold for field labels because it reduces the emphasis of other important headings. In addition, if everything is bold, nothing stands out “boldly.”
- Use initial caps for main headings, section headings, and subheadings only. Avoid using initial caps in field labels.
- Avoid using ALL CAPS in field labels. All caps can be used for small headings, but should be avoided for field labels.
- Avoid using a font size smaller than 8 points.
- Avoid using underlined text for emphasis. Users associate underlined text with hyperlinks.
- Avoid using color in text unless you want to create good contrast and use it as a means to distinguish section headings. Choose dark shades or white, depending on the background color.

Using the Text Block Control in FDA

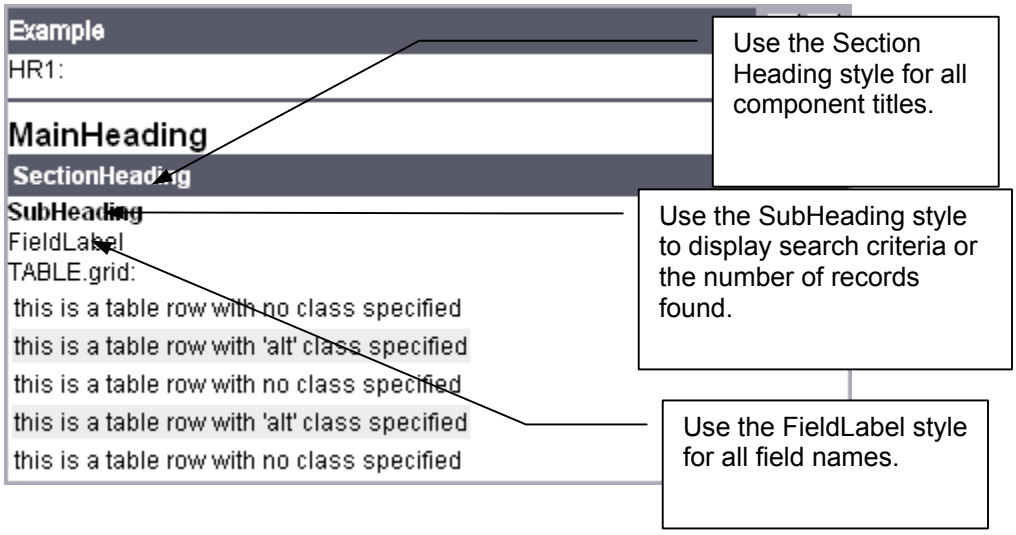
The text block control is a tool that allows you to extend the HTML look and feel of an application. View this control as a custom HTML control for your form. Any HTML text you place in this control will be streamed into the control’s location on the form.

The text block control in FDA does not yet automatically follow the cascading style sheet conventions defined by the Tools team. However, you can apply the style sheet when adding

content to the text block control by adding the appropriate class attributes to the HTML tags of the content. To make OneWorld Xe applications consistent, you must set the text block controls in FDA to the following specifications:

- The main heading segment in the text block control must use the MainHeading class.
- The section heading segment in the text block control must use the SectionHeading class.
- The subheading segment in the text block control must use the SubHeading class.
- The field label segment in the text block control must use the FieldLabel class.
- All grids must use the Table.Grid style. To add shading to every other row, use the "alt" class within the Table.Grid style.
- Set the default color for all the text in the text block controls to black. Avoid using color unless you need to draw attention to something, for example, error Messages.

The following example shows how each segment should look in J.D. Edwards software:



Note

J.D. Edwards recommends that you do not use the font override feature in the text block control. If you use font override, then the text in the control will look different from the rest of the form if the cascading style sheets change.

Colors

Use colors judiciously in your design. Colors are powerful for conveying information, but their overuse can be detrimental for users.

The J.D. Edwards Portal interface provides users a choice of color schemes for J.D. Edwards applications. Users can choose from a predefined scheme or create a new one based on their corporate positioning and branding standards. Each scheme is grouped according to tone and is listed in the following table:

Tone	J.D. Edwards software Schemes
Bright	Counting, Rush
Dark	Circuit, Default
Neutral	Earth Tones
Soft	Innovation Blue, Partnership Purple, Springtime, Techno
Warm	Cityscape, Solar

The following guidelines will help you make better and more usable color selections for your workspaces and J.D. Edwards applications:

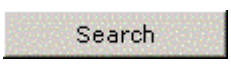
- Always use colors as a secondary cue. Do not rely only on colors to convey information or you will overlook color deficient and colorblind users (five to eight percent of the world's male population is color deficient).
- Make sure the design can stand on its own without colors. View your site in a grayscale scheme to check the result.
- Avoid highly saturated (bright) colors. They are good for catching one's eye, but will cause eye fatigue when viewed for a long period.
- Use soft or pastel tones for workspaces in which users might work all day. These color tones are easier to look at for a long period.
- Use Web-safe colors. If you use custom colors you need to be aware that they will be dithered on 256 color screens.

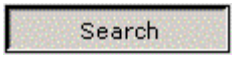
Icons

Icons can contribute to the aesthetics and visual attractiveness of an interface and, if clear and easily recognizable, can help clarify the interface. Substituting icons for text when you cannot convey the meaning clearly through words is a valid use of icons. However, if your icons require explanations (hover text), you have failed to convey the intent. Additionally, cultural differences exist that must be taken into consideration when you choose icon images. Consistency, clarity, simplicity, and familiarity are key attributes of icons.

Use the following guidelines when you include icons on your forms:

- If it takes you more than a few minutes to invent an icon, then the icon's meaning probably will not be clear to the user.
- Objects or concrete nouns are the easiest to represent as icons. Depict them in their simplest or most abstract form.
- Icons must look clickable. Show a visual cue, such as a three-dimensional background or shadowing, to indicate that it is clickable. Also change the icon's appearance after it is clicked.

Unclicked 

Clicked 

- Avoid using a graphic as a button. For example:

- Use only icons that are easily recognizable. Icons have a much quicker recognition rate when the user knows what they mean.
- Provide a parallel mechanism that supports both text and icons. Provide text for novice users of the site and icons for power users. Text must appear below the icon and it should be one word in length or be abbreviated.
- Provide a text equivalent for every icon and graphic by using an “alt” tag and “longdesc” in the element content. “Alt” tags allow users with disabilities to use screen readers to scan text.

Icons for International Use

Use the following guidelines to ensure that your icons will be understandable to international users:

- Avoid using icons that depict animals or body parts.
- Use androgynous, stick-like figures so as not to offend cultures that do not tolerate graphical representations of people.
- When showing an object as an icon, choose the most widely recognized version of the object.
- If no international version of an object exists, choose the most common (that is, the most familiar to the largest number of users) version.
- Size the label area of the icon to fit the largest label in English and multiply this size by the amount the text expands as it translates to other languages. A good rule to follow is to allow for a 30 percent expansion of the text.

Note

Always check with the Translation team if you are unsure about the international implication of an icon.

Graphics

Use the following guidelines when you include graphics on your forms:

- Use graphics to convey meaning. Aesthetics alone is not a good reason for using graphics.
- Avoid using background images—they reduce the legibility of your Web page or Web form. In addition, background images and graphics take a longer time to load in browsers.
- Provide a text equivalent for every graphic by using an “alt” tag and “longdesc” in the element content. “Alt” tags allow users with disabilities to use screen readers to scan text.
- Avoid using animated gifs because they are rarely meaningful, can be distracting to users, and can affect the performance of the application.

- Use colored backgrounds only if the colors are muted and are low in overall color saturation. Pastels, light grays, and light earth tones are best.
- Avoid using flashing graphics that cause the screen to flicker. Flickering or flashing images can cause seizures in users with photosensitive epilepsy. Flashing in the range of 4 to 59 flashes per second (Hertz) with peak sensitivity at 20 flashes per second can cause these problems.
- Graphics should not look clickable.

Designing J.D. Edwards Applications for Web Use

When using J.D. Edwards's Form Design Aid (FDA) tool to develop interactive applications targeted for the Web (HTML), you must consider several design strategies during the design and coding process. This topic discusses several of those considerations.

HTML Web applications often have a different look and feel compared to Windows applications. The design requirements dictate how different the look and feel will be and what the differences are. For example, in Employee, Customer, and Supplier Self-Service applications, one requirement was that non-J.D. Edwards users must be able to navigate between applications and forms. This requirement culminated in the removal of all Menu/Toolbar Exits, which were considered a J.D. Edwards-specific navigation mechanism. To replace Menu/Toolbar Exits, the design allows users to navigate between applications and forms using hyperlinks, which are commonly found in HTML-based applications.

When designing an application for use on both the Windows platform and the HTML platform, you might want to use form modes if you want them look different. Refer to the *Development Tools* guide for a detailed explanation of Form Modes.

When creating menus in J.D. Edwards software to invoke your new applications, you can call the form in a specific Form Mode. If your application is Web only and if you did not use Form Modes in FDA, you can generate the application (HTML serialized objects) in any of these modes. However, if your Web-only application provides form interconnects to other applications that need to be executed in a specific mode, then you must design and generate your new application in that required mode. This requirement is because of the inheritance of execution modes when a form interconnect is processed. In other words, the mode of the child form is forced to be identical to the mode of the calling form.

Note

All forms called from a menu entry point must be generated in the same mode.

Just as when you are creating a Windows application, choosing the appropriate type of form to use is also a key design consideration. Using the wrong form type causes inefficiencies during runtime and user frustration. For example, if you use a form type that supports table updates (for example, headerless detail or header detail) for display-only purposes, the form will refresh (that is, a round trip to the Web server occurs) when the user clicks into each grid row.

Performance is also a key consideration in designing applications for Web use. In addition to the types of performance issues that a Windows environment presents, a Web environment presents additional performance issues to be considered before you perform any coding:

- Reduce unnecessary round trips as much as you can by not using event rules with logic that requires immediate processing.

- Separate business logic from user interface logic. For example, you should only use event rules directly from your application to control the user interface on the form. However, you should create business logic components (business functions) that encapsulate business processes such as calculations, database access, and other types of business logic, and call these components from the application. Doing so makes your application easier to maintain. Reducing the logic in the event rules also means your application will be less processor-dependent, thus improving the performance of your application.
- Consider database access performance. **Always set up correct indexes for your tables. Use indexes as much as possible for database access.** Good design practices for three-tier architecture systems advocate separating the logic for database access into individual components (business functions) that are called from your application.
- Beware of designs that suppress the display of grid lines in the Grid Record is Fetched or the Write Grid Line – Before events. **If the form is suppressing most or almost all of the grid records that it fetches based on the search criterion, then you should devise a more focused search criterion to avoid having to suppress most grid records.**
- Do not turn off "Page At a Time" processing. If you need the total number of records fetched, you can do so by making a JDB call from business functions. "Page at a time" processing offers great performance benefits.
- Do not worry about the round trip from the Web server to the enterprise server. The link between the Web server and enterprise server is usually fast. Additionally, both servers perform smart caching.

Using Form Design Aid to Design Web Applications

Here are some tips and techniques when using the J.D. Edwards Form Design Aid (FDA) to create Web applications.

See Also

- *Generating J.D. Edwards Serialized Objects* in the *J.D. Edwards Java Server Installation* guide for your platform for information about generating your forms after you have designed them

Designing Forms Using Multiple Modes

You can use control modes to develop an application with multiple interfaces, which reduces the need to maintain several different versions of the same application. You can create one base application and use modes to modify the application for different interfaces. You can enable or hide controls on forms for each mode. Only visibility and enable/disable properties for controls, columns, and menu exits are different for different modes. If you show hidden fields, they appear only for the current mode. All other properties are the same and are common for all modes. All fields are enabled and appear all forms.

The Windows runtime engine does not recognize control modes; only the HTML runtime engine recognizes them. Mode 1 is the default mode. You attach an application to a menu to run. This menu allows you to run an application in different modes. When you run an application over the Web, the application runs in mode 1 by default and another mode, if you specify one. If you attach an application to a Windows menu, the Windows runtime engine ignores any modes that you specified and runs the application in mode 1. Use modes consistently throughout your applications. To create Web-enabled versions of your forms, you

generate them in Java and HTML using the Java & HTML Generator. The generator allows you to generate forms simultaneously for one or more modes.

Hiding Menu/Toolbar Exits

If you want to hide form and row exits and all other types of menu/toolbar exits (select, cancel, OK, or delete) from your application, you can do so by selecting the hidden option while you are in mode 2 or 3. If you choose the hidden option in mode 2, then the exit is hidden only in mode 2. In modes 1 and 3, the exit still appears.

► To hide menu/toolbar exits

1. In Form Design and while in mode 2 or 3, on the form with which you are working choose Menu/Toolbar Exits from the Form menu.
2. Choose the exit you want to hide and click Select.
3. In the State group box, click Hidden, and then click OK.

   When generated in mode 2 or 3 (whichever you were in), the exit will not appear on the form.

Enabling In-Your-Face-Errors

The In-Your-Face-Errors property is a form-level property that is available only for the HTML platform. Typically, the system indicates an application error by highlighting the Errors and Warnings hyperlink in the upper right-hand area of the application. When enabled, however, the In-Your-Face-Errors property causes application errors to appear on the Web page.

► To enable In-Your-Face-Errors

1. In Form Design, do one of the following tasks:
   - Create a new form.
   - Choose an existing form and choose Form Properties from the Form menu.
2. On Form Properties, click Enable In-Your-Face-Errors Display, and then click OK.

Sizing Forms for Screen Sizes

Sometimes, you want to size the form according to your target user's browser size. For example, most users of Customer Self-Service applications run in 640x480 screen resolutions only.

To create guides for sizing appropriately, you can choose to set the Form Guide value on the Form Properties dialog. Doing so adjusts the blue line guide in FDA appropriately. FDA does not enforce the size, but merely guides the designer to keep objects within the blue line guides.

► To size forms for screen sizes

1. In Form Design, do one of the following tasks:
   - Create a new form.
   - Choose an existing form and choose Form Properties from the Form menu.

2. On Find/Browse Form Properties, choose the screen size you want to use from the Form Guide dropdown menu, and then click OK.

Hiding the Grid Row Selector

The Hide Grid Row Selector property is a grid-level property that is available for the HTML platform. If you decide not to show grid row selectors on your grids, select the Hide Row Selector option on the grid properties dialog box.

► To hide the grid row selector

1. In Form Design, on the form with which you are working, click in the grid and then choose Item Properties from the Edit menu.
2. On Grid Properties, in the HTML Properties section, click Hide Row Selector, and then click OK.

Showing an Alternate Grid Row Format

The Use Alternate Grid Row Format property is a grid-level property that is available for the HTML platform. The property allows you to control the appearance of your grid.

To use this option, you must provide specific HTML tags by clicking on the Row Format button. Follow the instructions in the HTML Alternative Grid Row Format dialog when creating your HTML tags for the grid rows. These HTML tags control the appearance of each grid row. You may also choose to output a Media Object Image or Text type object for the grid record and hyperlinks.

This option is supported only in non-updateable grids.

► To show an alternate grid row format

1. In Form Design, on the form with which you are working, click in the grid and then choose Item Properties from the Edit menu.
2. On Grid Properties, in the HTML Properties section, click Use Alternate Grid Row Format, and then click OK.
3. On HTML Alternative Grid Row Format, enter a string with the HTML tags required to control the appearance of each grid row as required.
4. Click OK.
5. On Grid Properties, click OK.

Using Multi-Line Edit to Control Page Refresh

The Allow Multi-line Edit property is a grid-level property that is available for the HTML platform. This property is applicable only to grids that allow users to edit the grid records. Checking this option causes the runtime engine for HTML to show grid rows that contain Editable Text Box controls in each cell (all rows) instead of only in the active grid row. It also prevents the system from refreshing the page every time the user exits a grid row. Instead, the system delays the execution of the Grid Row is Exited events and only executes them in groups of 3 to 5 rows using a silent post.

► To use multi-line edit to control page refresh

1. In Form Design, on the form with which you are working, click in the grid and then choose Item Properties from the Edit menu.
2. On Grid Properties, in the HTML Properties section, click Allow Multi-Line Edit, and then click OK.

Controlling the Number of Grid Rows for Each Page of Grid Records

The Grid Row Count property is a grid-level property that is available for the HTML platform. A non-zero value for this option causes the runtime engine for HTML to show the number of grid rows specified. By default (a value of 0), the runtime engine displays 10 records in the grid per page.

► To control the number of grid rows for each page of grid records

1. In Form Design, on the form with which you are working, click in the grid and then choose Item Properties from the Edit menu.
2. On Grid Properties, in the HTML Properties section, enter the number of grid rows that you want to appear per page in the Grid Row Count field, and then click OK.

Showing Check Boxes in Grid Cells

The Check Box property is a grid column-level property that is available for the HTML platform. It displays check boxes in grid cells for a specific column that provides on/off state information.

To use this option, you must provide specific values that specify the Checked and Un-Checked state value. During runtime, the system detects the value of the GC variable for the column and cross-references the value with the settings you specified to render a checked or unchecked check box.

► To show check boxes in grid cells

1. In Form Design, on the form with which you are working, click in the grid and then choose Item Properties from the Edit menu.
2. On Grid Properties, in the Grid Columns section choose a grid column, and then click Grid Column Properties.
3. On Grid Column Properties, in the Display Style section, click Check Box.
4. Click Values.
5. On Grid Column CheckBox Values, enter numerical values in the Checked and Un-Checked fields, and then click OK.

   The values you enter should correspond to the values you set in the GC variable.
6. On Grid Column Properties, click OK.
7. On Grid Properties, click OK.

Showing Hyperlinks in Grid Cells

The Clickable property is a grid column-level property that is available for the HTML and Windows platforms. It allows you to display text in grid cells for a specific column as

hyperlinks (clickable text). Any non-blank value in the grid cell for that column appears as a hyperlink.

Enter the logic to execute when the hyperlink is clicked in the Grid Column Clicked event. GC values are available for the grid row that was clicked.

► To show hyperlinks in grid cells

1. In Form Design, on the form with which you are working, click in the grid and then choose Item Properties from the Edit menu.
2. On Grid Properties, in the Grid Columns section, choose a grid column and then click Grid Column Properties.
3. On Grid Column Properties, in the Attributes section, click Clickable and then click OK.

   Use the Grid Column Clicked event to define what should occur when the user clicks on a hyperlink in this grid column.

4. On Grid Properties, click OK.

Showing Hyperlinks in the Form, Group Box, or Tab Control

To create hyperlinks that are placed in the form, group box, or tab control, you can use either the text block control or static text control. Usually, the text block control is used whenever more control over the text format for the hyperlink is needed; otherwise, the static text control is used due to less overhead.

To use a text block control to display its text contents as hyperlinks, you add a segment to hold the text of your hyperlink, and then select the Clickable option in the text block control properties dialog. The logic to execute when the hyperlink is clicked should be entered in the Text Clicked event.

To use a static text control to display its text contents as hyperlinks, you assign the hyperlink text that you wish to show, and then select the Clickable option in the Static Text properties dialog. The logic to execute when the hyperlink is clicked should be entered in the Text Clicked event. You cannot override the font and color for clickable text segments.

► To use the text block control to show a hyperlink

1. In Form Design, on the form with which you are working, select the text box control you want to affect, and then choose Item Properties from the Edit menu.
2. On Text Block Control Properties, click Add Segment, and enter the text that you want to use as your hyperlink.

   Use the Text Clicked event to define what should occur when the user clicks on this hyperlink.

3. In the Segment Information section, click Clickable, and then click OK.

► To use the static text control to show a hyperlink

1. In Form Design, on the form with which you are working, select the static text control you want to affect, and then choose Item Properties from the Edit menu.
2. On Static Text Properties, enter the text in the Static Text field.

Use the Text Clicked event to define what should occur when the user clicks on this hyperlink.

3. In the Attributes section, click Clickable, and then click OK.

Inserting Custom HTML Tags into a Form

You might want to insert your own HTML into the form to produce a more customized HTML look and feel. To do this, use Text Block Control.

The text block control can be used to extend the functionality already provided by the J.D. Edwards HTML platform. The HTML platform streams any text contained in the platform into the HTML of the form as it is generated. In addition, you can add, delete, and update the text within the control at runtime. Taken together, this control can greatly extend the functionality of the HTML client.

You can enter text in the text block control either by defining text segments in the text block control in FDA or by inserting text segments at runtime via system functions.

Inserting Custom HTML with FDA

Place the Text Control Block on the form, and insert segments as needed to hold the tags. It is advisable to segregate segments that are data-driven and segments that contain static HTML tags. Segments that are data-driven can be set during Dialog is Initialized, Grid Record is Fetched, or other appropriate events by using the Update Segment() system function call.

See Also

- *Using the Text Block Control in FDA*

► To insert custom HTML with FDA

1. In Form Design, choose Text Control Block from the Insert menu.
2. Click on the form in which to place the control.
3. Click on the new control, and then choose Item Properties from the Edit menu.
4. Click Add Segment and enter the HTML text you want to add to the form.
5. To make the segment clickable by the user, click Clickable.

   This option causes the system to generate the segment as a hyperlink that executes a Text Clicked event when the user clicks it.

6. To override the cascading style sheet (CSS) setting for the segment, perform the following steps:

   a. Deselect Use Default Font and Color.

   b. Click Font and Color.

   c. On Font, indicate the typeface, font characteristics, and color you want to use, and then click OK.

   J.D. Edwards recommends that you use this option sparingly.

7. Click OK.

Inserting Custom HTML at Runtime

You can use the following system functions to manipulate the text block control from ER:

Function	Parameters	Comments
Add Segment	Text Control Text Font Clickable (true/false) SegmentId – Returned unique segment ID for the added segment	
Get Last Clicked Segment	Text Control SegmentId – Returned segment ID of the segment that was last clicked by the user	Use this on the Text Clicked event of the text control to determine what segment was clicked.
Get Segment Information	Text Control Text – Returned text of the passed in segment ID Clickable – Returned clickable flag of the passed in segment ID SegmentId – Segment ID of the segment being inquired about	
Remove Segment	Text Control SegmentId – Segment ID of the segment to be deleted	
Update Segment	Text Control Text Font Clickable SegmentId – Segment ID of the segment to be updated	

The parameters, Clickable and SegmentId, expect variables of Integer data type.

Advanced Functionality

Except for applying font and color (if specified) and adding the necessary tags for executing the clickable event, the text entered into this control is streamed unfiltered into the HTML of the form as it is generated. Therefore, you must be familiar with the CSS (Cascading Style Sheet) scheme that J.D. Edwards uses to generate its HTML. By using the appropriate class names in the HTML tags in the text block control, the text block control will have the same look and feel as the form, and the look and feel will change with the rest of the form if the customer changes the controlling CSS forms. This also means that the developer should never define the font and color of text segments unless necessary. The font and color for a text segment ignores the CSS definition of the form and, thus, will always look different from the rest of the form.

Note

The system will not format the text block control correctly in FDA because FDA currently does not communicate with the Web server to determine the correct CSS settings. To see the control correctly formatted, generate the form and then view it on the HTML platform.

The following table lists the common J.D. Edwards CSS tags:

Class Name	Apply to…	Comment
Padded	TABLE	Creates a padded table the width of the page.
Border	Generic	
NoBorder	Generic	
WideTable	Generic	Width = 100%
TallTable	Generic	Height = 100%
TallAndWideTable	Generic	Width = Height = 100%
Grid	TABLE	
MainHeading	Generic	
SectionHeading	Generic	
GroupHeading	Generic	
SubHeading	Generic	
FieldLabel	Generic	
RaisedBorders	Generic	
BlackBorders	Generic	
ClearBorders	Generic	
QBECell	Generic	
GridHeaderCell	Generic	
GridCell	Generic	
InYourFaceError	Generic	
InYourFaceWarning	Generic	
ToolbarText	Generic	
GroupBox	Generic	
GroupBoxHeader	Generic	
FormLabel	Generic	
FormAboveGrid	Generic	

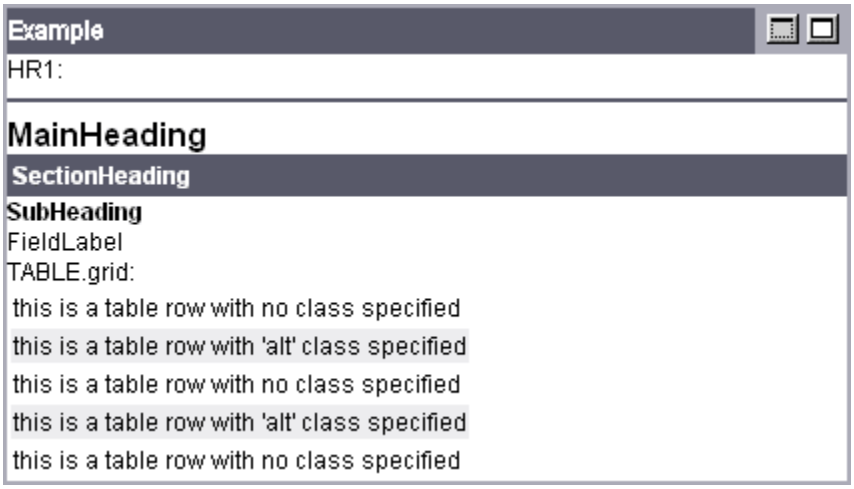
The following HTML tags have been specified with custom style tags so they can be used with the assurance that their text will be formatted correctly:

Tag	Comment
HR1	
HR2	
BODY	Non-formatted text without any enclosing tags, classes, or both entered by the text block control will have a base style to rely on.
INPUT	
SELECT	
A	
TABLE	
TABLE TD	

The following sample HTML code produces the figure that immediately follows it:

```
Text in text block control :
<HR1>
HR1:
</HR1>
<HR class=MainHeading>
MainHeading
</HR>
<HR class=SectionHeading>
SectionHeading
</HR>
<HR class=SubHeading>
SubHeading
</HR>
<span class=FieldLabel>
SubHeading
TABLE.grid:
</span>
<TABLE class=Grid>
<TR><TD >
this is a table row with no class specified
</TD></TR>
<TR class=alt><TD >
this is a table row with no class specified
</TD></TR>
<TR><TD >
this is a table row with no class specified
</TD></TR>
```

```
<TR class=alt><TD >
this is a table row with no class specified
</TD></TR>
<TR><TD >
this is a table row with no class specified
</TD></TR>
```



Inserting Images into a Form

You can place a variety of images on a form with the Bitmap Control. The image can be static, or you can make it clickable. Use the Button Clicked event to define what should occur when the user clicks the image.

► To insert images into a form

1. In Form Design, choose Bitmap from the Insert menu.
2. Click on the form in which to place the control.
3. On Bitmap Properties, click Find to find the file you wish to use for your bitmap control.

   The Bitmap Properties form displays the bitmap you have chosen.

4. Complete the following field:
   - Tool Tip Text

     This text is like Hover Helps appears if the user makes the mouse pointer hover over the bitmap.

5. Click one or more of the following attributes:
   - Clickable

     If you turn on the Clickable option, a Button Clicked event is enabled for the bitmap.

   - Maintain Aspect Ratio

If you turn on the Maintain Aspect Ratio option, the bitmap will maintain its dimensions if you resize it. This ensures that your image does not become distorted if you resize it.

Inserting Drop-Down Lists into a Form

To display drop-down lists in a form, you should place the combo box control on the form. Then, drag and drop a data dictionary item right on the combo box control. When correctly done, your Combo Box Properties dialog will state the data dictionary item to which it is connected.

The content of the drop-down list is derived from (and only from) the UDC values and descriptions with which the data dictionary item is associated. The list is based on the edit rule properties of the data dictionary item. Allowing only a UDC to populate a combo box ensures that the all combo box values can be translated correctly.

► To insert drop-down lists into a form

1. In Form Design, choose Combo Box from the Insert menu.
2. Click on the form in which to place the control.
3. From the View menu, choose Data Dictionary Browser.
4. In Data Dictionary Browser, find the data dictionary item you want to use for the combo box control.
5. Select the combo box control, and then drag the data dictionary item from the Data Dictionary Browser and drop it directly on top of the combo box control.
6. Select the combo box control, and then choose Item Properties from the Edit menu.
7. Verify that the data dictionary item has been associated with the combo box and click OK.

Performing Custom Selects and Sorts for the Grid

Within the Find menu exit's Button Clicked event, you may customize your grid selection criterion by using the Clear Select() system function and then performing the Set Selection() system function. You may also choose to perform Clear Sequencing() and then the Set Sequencing() system function to customize the grid sequencing settings of the grid.

By using Set Selection() or Set Sequencing() system functions, you do not have to use hidden filter fields on the form to perform custom grid selection criterion. Furthermore, the Set Selection() function can support "Or" and "And" statements in the resulting SQL statements.

Lastly, the Set Selection() system function does not depend on fields that exist in the business view of the form. Consequently, you can actually perform selects on fields that are outside of the business view, so long as they belong to the same table as the business view.

J.D. Edwards Software Events on an HTML Client

Two J.D. Edwards software events do not work well in an HTML environment because of posting. J.D. Edwards recommends that you do not use the Control Is Exited event to hide or show controls or to enable or disable grids. If you do use Hide/Show or Enable/Disable Grid system functions in the Control Is Exited event, the system automatically generates JavaScript to refresh the page so that the event can be processed immediately. Note that this solution causes a round trip.

The following table shows how an HTML client responds to all J.D. Edwards software events, including which ones trigger a post:

Object Type	Event	Triggers Post	Delayed
Bitmap	Button Clicked	Yes	No
Checkbox	Selection Changed	No, but will trigger post if ER is present.	—
Combo Box	Control is Exited	No	Yes
ComboBox	Control is Entered	No	Yes; will run against state of form prior to post.
Document Object	Media Objects — Form def	— (Triggered on server during form initialization.)	—
Document Object	Media Objects — Row def	— (Triggered on server during Find button processing.)	—
Edit	Control Exited/Changed—Asynch	No, but will trigger post if the event flag, HTML Post, is turned on.	Yes
Edit	Control Exited/Changed—Inline	No, but will trigger post if the event flag, HTML Post, is turned on.	Yes
Edit	Control is Entered	No	Yes; will run against state of form prior to post.
Edit	Control is Exited	No, but will trigger post if the event flag, HTML Post, is turned on.	Yes
Edit	Post Visual Assist Clicked	— (Triggered on server immediately after the event, Visual Assist Button Clicked, is finished.)	—
Edit	Visual Assist Button Clicked	Yes	No
Form	Add Record to DB — After	— (Triggered on server during OK button processing.)	—
Form	Add Record to DB – Before	— (Triggered on server during OK button processing.)	—
Form	Clear Screen After Add	— (Triggered on server during form initialization.)	—
Form	Clear Screen Before Add	— (Triggered on server during form initialization.)	—
Form	Dialog is Initialized	— (Triggered on server during form initialization.)	—
Form	Director Back Button Clicked	N/A (Not valid on the HTML client.)	N/A
Form	Director Cancel Button Clicked	N/A (Not valid on the HTML client.)	N/A
Form	Director Next Button Clicked	N/A (Not valid on the HTML client.)	N/A
Form	End Dialog	— (Triggered on server during form closing.)	—
Form	Grid Record is Fetched	— (Triggered on server during Find button processing.)	—

Object Type	Event	Triggers Post	Delayed
Form	Last Grid Record Has Been Read	— (Triggered on server during Find button processing.)	—
Form	Post Commit	— (Triggered on server during Find button processing.)	—
Form	Post Dialog is Initialized	— (Triggered on server during form initialization.)	—
Form	Update Record to DB — After	— (Triggered on server during OK button processing.)	—
Form	Update Record to DB — Before	— (Triggered on server during OK button processing.)	—
Form	Write Grid Line—After	— (Triggered on server during OK button processing.)	—
Form	Write Grid Line—Before	— (Triggered on server during OK button processing.)	—
Grid	Add Grid Rec to DB — After	— (Triggered on server during OK button processing.)	—
Grid	Add Grid Rec to DB — Before	— (Triggered on server during OK button processing.)	—
Grid	Add Last Entry Row to Grid	— (Triggered on server during Find button processing.)	—
Grid	All Grid Recs Added to DB	— (Triggered on server during OK button processing.)	—
Grid	All Grid Recs Deleted from DB	— (Triggered on server during OK button processing.)	—
Grid	All Grid Recs Updated to DB	— (Triggered on server during OK button processing.)	—
Grid	Delete Grid Rec from DB—After	— (Triggered on server during OK button processing.)	—
Grid	Delete Grid Rec from DB—Before	— (Triggered on server during OK button processing.)	—
Grid	Delete Grid Rec Verify—After	— (Triggered on server during Delete button processing.)	—
Grid	Delete Grid Rec Verify—Before	— (Triggered on server during Delete button processing.)	—
Grid	Double Click on Row Header	Yes	No
Grid	Get Custom Grid Row	— (Triggered on server during Find button processing.)	—
Grid	Kill Focus on Grid	No	Yes
Grid	Row Exit & Changed — Asynch	Yes. If the grid has the flag, Allow Multi-line Edit, turned on, the post occurs in the background without refreshing the form, making it appear that the post did not happen when it actually did.	No

Object Type	Event	Triggers Post	Delayed
Grid	Row Exit & Changed – Inline	Yes. If the grid has the flag, Allow Multi-line Edit, turned on, the post occurs in the background without refreshing the form, making it appear that the post did not happen when it actually did.	No
Grid	Row is Entered	No	Yes; will run against state of form prior to post.
Grid	Row is Exited	Yes. If the grid has the flag, Allow Multi-line Edit, turned on, the post occurs in the background without refreshing the form, making it appear that the post did not happen when it actually did.	No
Grid	Set Focus on Grid	Sometimes. A post occurs if the user clicks a cell in an editable grid. The event is delayed if the user tabs into the editable grid.	Sometimes. A post occurs if the user clicks a cell in an editable grid. The event is delayed if the user tabs into the editable grid.
Grid	Update Grid Rec to DB—After	— (Triggered on server during OK button processing.)	—
Grid	Update Grid Rec to DB—Before	— (Triggered on server during OK button processing.)	—
Grid Column	Col Exited & Changed – Asynch	No, but will trigger post if the event flat, HTML Post, is turned on.	Yes
Grid Column	Col Exited & Changed — Inline	No, but will trigger post if the event flat, HTML Post, is turned on.	Yes
Grid Column	Col is Exited	No, but will trigger post if the event flat, HTML Post, is turned on.	Yes
Grid Column	Grid Column Clicked	Yes	No
Grid Column	Post Visual Assist Clicked	— (Triggered on server immediately after the event, Visual Assist Button Clicked, is finished.)	—
Grid Column	Visual Assist Button Clicked	Yes	No
Hyper Item	Button Clicked	Yes	No
Hyper Item	Post Button Click — Asynch	—— (Triggered on server immediately after the hyper item's event, Button Clicked, occurs.)	—
Hyper Item	Post Button Clicked	— (Triggered on server immediately after the hyper item's event, Button Clicked, occurs.)	—
Parent/Child	Kill Focus On Control	No	Yes
Parent/Child	Set Focus on Control	No	Yes
Parent/Child	Tree — Begin Drag Operation	Yes	No
Parent/Child	Tree — Cancel Drag Drop	Yes	No
Parent/Child	Tree — Drag Over Node	Yes	No

Object Type	Event	Triggers Post	Delayed
Parent/Child	Tree — End Drag Drop Operation	Yes	No
Parent/Child	Tree Node Is Collapsing	Yes	No
Parent/Child	Tree Node Is Deleted	Yes	No
Parent/Child	Tree Node Is Expanding	Yes	No
Parent/Child	Tree Node Selection Change	No, but will trigger post if the event flat, HTML Post, is turned on.	Yes
Parent/Child	Tree—Node Level Changed	No, but will trigger post if the event flat, HTML Post, is turned on.	Yes
Parent/Child — Grid	All Grid Recs Deleted from DB	— (Triggered on server during OK button processing.)	—
Parent/Child — Grid	Delete Grid Rec from DB—After	— (Triggered on server during OK button processing.)	—
Parent/Child — Grid	Delete Grid Rec from DB—Before	— (Triggered on server during OK button processing.)	—
Parent/Child — Grid	Delete Grid Rec Verify—After	— (Triggered on server during Delete button processing.)	—
Parent/Child — Grid	Delete Grid Rec Verify—Before	— (Triggered on server during Delete button processing.)	—
Parent/Child — Grid	Double Click on Row Header	Yes	No
Parent/Child — Grid	Get Custom Tree Node	— (Triggered on server during Find button processing.)	—
Pushbutton	Button Clicked	Yes	No
Pushbutton	Post Button Clicked	— (Triggered on server immediately after the push button's event, Button Clicked, occurs.)	—
Radio button	Selection Changed	No, but will trigger post if ER is present.	—
Static Control	Text Clicked	Yes	No
Tab Page	Tab Page is Initialized	— (Triggered on server before the event, Tab Page is Selected, is processed, provided that the tab page is selected for the first time.)	—
Tab Page	Tab Page is Selected	Yes	No
Text Block	Text Clicked	Yes	No
Tree Control	Double Click on Leaf Node	No, but will trigger post if ER is present.	—
Tree Control	Get Custom Tree Node	— (Triggered on server during the tree control's event, Tree Node Is Expanding.)	—
Tree Control	Kill Focus On Tree	No	Yes

Object Type	Event	Triggers Post	Delayed
Tree Control	Set Focus On Tree	No	Yes
Tree Control	Tree Node Is Collapsing	Yes	No
Tree Control	Tree Node Is Deleted	Yes	No
Tree Control	Tree Node Is Expanding	Yes	No
Tree Control	Tree Node Selected	No, but will trigger post if the event flag, HTML Post, is turned on.	Yes

Designing J.D. Edwards Applications for Mobile Use

From SP15 onwards, the J.D. Edwards HTML client provides full support for mobile devices based on the Windows CE platform, which is highly compatible with the HTML client architecture. Currently, all of the Window CE devices have a good support for HTML through various versions of Internet Explorer (IE) browser.

J.D. Edwards Application support for mobile devices includes the following:

- Windows CE 3.0 devices running IE 4.x Browser
  These devices include HP Jornada 720, NEC Mobile Pro™ 790, and so forth. Version 3.0 represents the latest version of Windows CE, and the devices based on it are recommended over Windows CE 2.11 because Version 3.0 provides support for the HTML4 specification by the IE 4.x Browser. IE 4.x is the richest browser currently available on mobile devices and includes good support for DTML and JavaScript.
- Windows CE 2.11 SP1 and above running IE 3.01
  J.D. Edwards HTML Client supports these devices. However, these devices are not recommended because they represent the older technology that has been superceded by Windows CE 3.0.
- Windows CE 3.0 running Pocket PC 2000
  These devices include Compaq IPAQ, HP Jornada 540, Casio Cassiopeia, and so forth. These devices also provide HTML access based on the IE3.01 browser. The J.D. Edwards HTML Client fully supports these devices.

One of the design goals for the Mobile Device support in the HTML client was that we leverage the existing architecture and require absolutely no application development changes. The J.D. Edwards Mobile Device architecture supports all J.D. Edwards interactive applications run on the supported platforms. It leverages the existing architecture and requires that you make no application development changes to your applications. Currently J.D. Edwards Mobile support is available for connected users only; that is, users who are not connected to the network via a browser are not supported. Support for the Palm Computing platform is also unavailable because of its lack of good browser support.

Mobile Device Runtime Architecture

The architecture providing support for mobile devices in the J.D. Edwards HTML client is the same as the one for desktop. The output generation for the HTML4-compliant browsers remains essentially the same. However, output is generated for Windows CE browsers running IE3.01.

When the user first signs into J.D. Edwards JAS server, a session is established between the client browser and the Web server. The session then holds information about the client platform and the browser. The system generates HTML output based on this information.

Mobile Device Design Strategies

When you develop applications for mobile devices, use the same design strategies as for regular Web-based applications. However, remember that the form factors are limiting on these devices. The form factor on Window CE 3.0 devices is limited to 640x480, while the form factor on Pocket PC is limited to 240x320. The following are additional design strategies to consider:

- In browser-based applications, vertical scrolling is generally more acceptable than horizontal scrolling.
- Try stacking controls top-to-bottom for the Pocket PC interface to make the applications more useful. You can limit the number of grid columns to prevent the user from scrolling too much.
- From SP16.1 forward, J.D. Edwards HTML supports grid tabs on HTML4 compliant browsers. However, grid tabs are not supported on Pocket PCs. Grid tabs support a feature called Default for Pervasive Device. You can use this feature to limit the number of grid columns displayed on Pocket PC for existing applications.

► To use a specific grid format for mobile devices

1. On any form in HTML showing an editable grid, click Customize Grid.
2. On Customize Grid, choose a format from the Available Formats list.
3. To make a grid format the default format for mobile devices, click Default for Pervasive Devise.
4. Click Close.

Functional Differences between HTML and Mobile Devices

Event Handling

The J.D. Edwards HTML Client and Mobile Device handle form events similarly. However, on the Pocket PC, the Control Is Exited event is never processed for a control until the user changes the data in the associated field. In HTML 4-compliant browsers such as IE 5.x on a desktop and IE 4.x on Windows CE 3.0, JavaScript is used to keep track of all the controls that the user has tabbed out of. This information is posted to the Web server and used to run the Control Is Exited event. However, on the IE 3.01 browser on Pocket PC, the virtual client keeps a virtual image of the Form, and this image is used to compare the fields that are changed. Therefore, Control Is Exited and its associated events (such as Control is Entered, Control is Exited, and Changed) are processed only if a field is modified. The logic of your application should not depend on Control Is Exited to be processed even if a particular field is not modified.

Usability

The look and feel of applications running on mobile devices is not as rich as the one on Desktop IE 5.x browser because the DHTML and JavaScript support on the browsers for these devices is limited. The following is a list of the most important differences:

- Grid scrollbar support does not exist on any mobile device.
- The HTML Client on a Pocket PC has limited keyboard support.

- Multi-line Grid Editing functionality does not exist on the Pocket PC. Therefore, these devices are not suitable for high-volume data entry.
- Support for Media Object RTF Editing and OLE Objects does not exist.
- No Export/Import functionality is supported on any mobile device.
- Support for text block controls is limited by the mobile device. If an application has a text block control that relies heavily on DHTML/JavaScript, it may not work correctly on Pocket PC platforms.
- Support for Viewing PDF files for submitted reports does not exist.