WSIA Customization Subcommittee

Usage Scenarios

1 Memory Configurator

1.1 SKU Display

A configurator needs to display the reseller's SKU numbers rather than the manufacturer's part numbers.

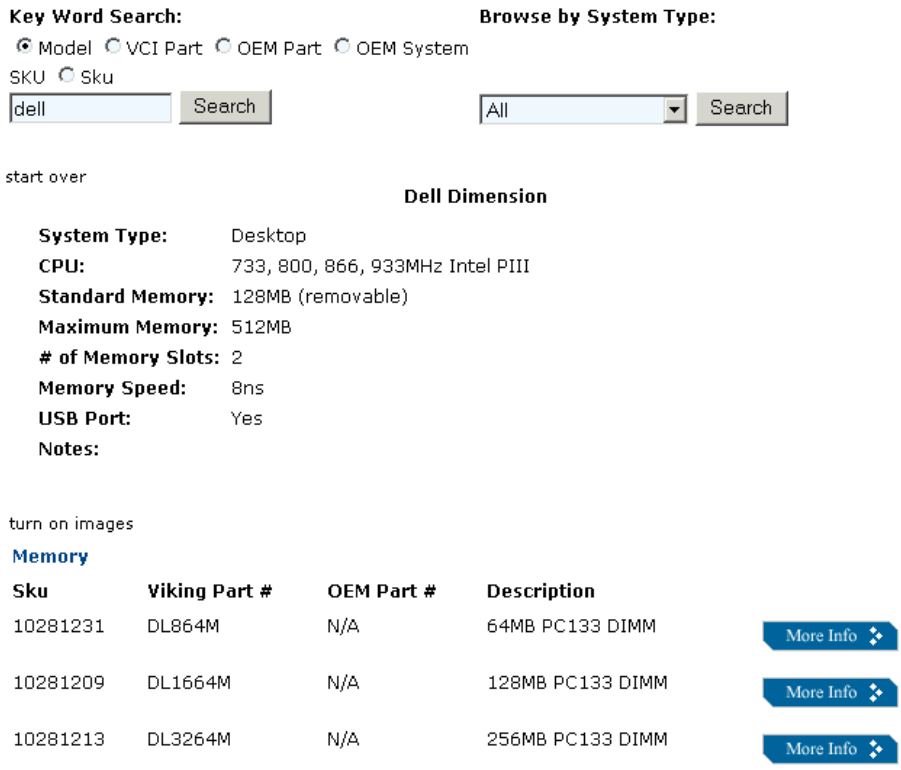
End User Experience Alternatives:

- The Configurator is presented at the Reseller with the Reseller SKU numbers
- The Configurator is presented at the Reseller site with both the Manufacturer and the Reseller SKU numbers.

Configuration Alternatives:

- Reseller and Manufacturer SKU numbers are available in a mapping table at the Reseller.
- Reseller and Manufacturer SKU numbers are available in a mapping table at the Manufacturer.

Below is an example of Viking’s configurator within Buy.com. The left column (SKU) is Buy.com's SKU number, dynamically added. The manufacturer part number is also displayed. Price is not displayed. In this particular case, SKUs are stored on Viking's site to facilitate search by SKU. In this example, all the items are ordered neatly in an HTML table.



URLs:

generic http://config.vikingcomponents.com/buydotcom/detail.asp?ConfigID=18008&model=4100

customized http://config.vikingcomponents.com/detail.asp?ConfigID=18008&model=4100

1.2 Availability

The configurator only needs to display items available for sale at the reseller site.

End User Experience Alternatives:

- The Configurator is presented at the Reseller, displaying only products that are available for sale at the Reseller.
- The Configurator is presented at the Reseller with all products, but those not available are marked as such.
- The Configurator is presented at the Reseller with all products, but when a user selects one, the user is provided with a notification that the product is not available.

Configuration Alternatives:

- Product availability requires on a real-time check, and on the individual user.
- Product availability requires on a real-time check, but not on the individual user.
- Product availability is available in a static table at the Reseller.
- Product availability is available in a static table at the Manufacturer.

A reference to an example of a corresponding application can be found at: http://configure.us.dell.com/dellstore/config.asp?customer_id=19&keycode=6V777&order_code=8TWORL&view=1

In this example, the items are not nicely arranged in an HTML table, but are rather grouped into option groups.

The example in sub-scenario 1.1 also demonstrates this need - the USB options are removed (curiously, the USB title remains).

1.3 Add to Cart

The configurator should provide an "Add to Cart" button that "returns" the product id to the reseller.

Configuration Alternatives:

- Returned product id is the Manufacturer's.
- Returned product id is the Reseller's.

WSIA Questions: how does the Reseller know that the user has completed the interaction with the Configurator? How does it receive the information back? What if the information is more complex (e.g., quantity)? What if the information is even more complex (a configured product)? (Didn't have time to bring a concrete example)

2 Health Insurance Enrollment

This is an application produced by an insurance provider and consumed in a participating company's intranet.

2.1 Personal Profile Entry Automation

The enrollment application has a screen for personal profile information. There is an opportunity for the consuming application to provide this data and skip this screen, enhancing the user-experience.

End User Experience Alternatives:

- The enrollment application is provided this information from the consumer and the application skips it associated screens.
- The enrollment application is provided this information from the user.

Configuration/Runtime Alternatives:

- The consumer stores profile information and provides the information through a runtime adaptation.
- The consumer stores profile information and provides the information through a runtime screen scraping and http automated post to the profile form.
- The enrollment application stores profile information or it is "synched".
- No profile information is stored and the user enters it.

It seems to me like this is more a pure data adaptation, with the enrollment application altering its workflow of its screen based on data already collected (this may be borderline with some of

the other use cases). I think that this type of scenario also applies to the Traveler's Cheques scenario, with the consuming application adapting the method of payment or "last amount ordered" on the provider side.

2.2 Restricting plan enrollment choices Part I

The insurance provider's application is produced for enrollment in all plans it offers (PPO, HMO, etc.). The consuming company though has a finite number of plans it offers to its employees (Only PPO).

2.3 Restricting plan enrollment choices Part II

The insurance provider's application is produced for enrollment in all plans it offers (PPO, HMO, etc.). The consuming company though has a finite number of plans it offers to its employees that are also based some employees attributes (Geographic region, Department).

3 Sports Portal

An integrated multimedia portal for Indy Car racing (for the purposes of this example I will refer to this application as LiveIndy).

3.1 Context-sharing for selected indy car driver

The consumer of LiveIndy is (for example) ESPN that adds history, statistics, and other value-added information around the live feed and data from LiveIndy. ESPN gives a consolidated view of the driver and all aggregated information relating to him in one application (including the consumed LiveIndy). ESPN functionality allows "switch driver" from a dropdown box which refreshes the application to have the point of view of the selected driver; this includes all historical, etc. information from ESPN as well as the live video feeds, selected drivers on the clickable map, etc. from LiveIndy.

End User Experience Alternatives:

- The LiveIndy portion of ESPN is autonomous and is refreshed or given driver context by directly manipulating it.
- The LiveIndy portion of ESPN is aware of driver context from the consuming ESPN application and refreshes accordingly.

3.2 (Reverse) Context-sharing for selected indy car driver

Not sure if this is relevant to this use case, but LiveIndy has a clickable map, that I assume behaves more like the embedded use case in that the click actions on the map are opaque to the consumer. This is not the case here, when you click and select a driver on the map to (opaquely) adapt the driver context of LiveIndy, this opaque adaptation becomes interesting to the consumer. ESPN would like to also "share" in this adaptation and refresh its application also.

This scenario can also be applied to something simpler that live multimedia feeds, like the stock-plot application… The analogies would be that the stock chart from FAME allows for plotting multiple stocks on one chart and being able to switch stock context by clicking on the associated plot line on the chart. This could also cause the surrounding divine portal to refresh with the new

stock context. (Also the first sub-scenario which would cause the chart to refresh when the divine application is manipulated).

Like Eilon said, I also have questions on where the lines are drawn here… Is this functionality covered in the Embedded case though an invokeAction() method where actionName=”selectDriver”, or is there some property on the LiveIndy application like “selectedDriver” that is altered through adaptations.

4 Model instance customizations

Producer/Consumer interactions around the Producer's model instance values. Consumer writes, reads Producer instance values either at endpoints of lifecycle or throughout the lifecycle of user interaction.

4.1 Sub-scenario: pre-filling Producer data

A Consumer needs to initialize a Producer form with data gathered previously in user interaction. For example, once the user has logged in, the Consumer determines his profile including bill-to/ship-to information that is then pre-filled into an order fulfillment form.

User experience alternatives:

- None - the goal is to display the Producer form pre-filled with the user's data

Configuration alternatives:

- Non-delegated: Consumer provides data to Producer which then includes it when rendering the form
- Delegated: Producer generates output along with metadata indicating where the Consumer should insert values. Note that the output generated by the Producer may be view only or be delegated to the Consumer with model and view separate but with binding expressions -- this option may allow for easier Consumer-based insertion of data into the output.

4.2 Sub-scenario: returning Producer data to the Consumer

The Consumer needs to determine the data values resulting from end-user interaction with the Producer. For example, the user navigates a catalog to a particular product for which the Consumer wants to provide reviews from an independent source. The Consumer needs to be able to obtain the product information for the currently selected item in order to pass that data to the review Producer.

Configuration alternatives:

- Data returned only at completion of Producer's execution:
- Data returned along with output at each cycle of end-user interaction

5 Model type and validation customizations

Producer/Consumer interactions around the Producer's field validations - necessarily involves notifying the Consumer during the lifecycle of interactions not just at the end. Requires a processing model for passing values between Producer/Consumer perhaps with an initial step as

they flow first through the Consumer, i.e. Consumer->Producer->Consumer…need to define a processing model for what happens at each step.

5.1 Sub-scenario: involving the Consumer in data validation -- case where Consumer narrows field validation (type) constraints on a Producer field

The Consumer needs to constrain the type of data which can be entered into a Producer's field given the state of interaction between the Consumer and the end-user. For example, a "delivery month" field in an order entry form must be constrained to accept values only in the first quarter - January, February, or March. The Consumer needs to "narrow" the acceptable values for that field without specifying which month in particular is to be set (i.e. the Consumer is not setting the instance value but adding a constraint to its type).

Configuration alternatives:

- The Producer generates its output without additional information from the Consumer, but then returns the user-specified value to the Consumer which then further validates its value. Some type of processing model between Producer and Consumer could be defined for a multi-stage data validation cycle.
- The Consumer passes a type constraint to the Producer in advance of generating the output. The Producer can use this constraint to be smarter in the output that it generates, e.g. including only specific pull-down options in a menu or generating script for client-side field validation taking into account the Consumer's constraints (Jan-Mar only for months).

5.2 Sub-scenario: involving the Consumer in data validation/computation --setting a Producer's field value given the values of other Producer data elements

The Consumer needs to add constraints specifying how the Producer's data elements are to be computed. For example, given the interaction's context, the Producer can determine that the interest rate on a loan will always be 2% above prime.

Configuration alternatives:

- Delegated from Consumer to Producer (reverse of normal delegation) - the Consumer passes a constraint expression to the Producer relating the two fields
- Non-delegated - the Consumer computes the correlated field value when informed of changes to the initial field.

5.3 Sub-scenario: involving the Consumer in data validation -- case where Consumer adds options to a Producer's field, is this valid/possible?

The Consumer wants to specify additional values for a field beyond those anticipated by the Producer - for example, colors for a clothing item or mortgage types for a loan.

Configuration alternatives:

- Define a broad type for the field such as string and then allow the Consumer to narrow that type as in the sub-scenario above. This option still allows the Producer to participate in field validation.
- Define an opaque set of enumerated values and manage them by delegating all field validation to the Consumer as in the validation sub-scenario above.

6 View customizations

Producer/Consumer interactions around construction of the Producer's form/frame initially -- Consumer setting preferences on how the Producer's output will be generated, adding/deleting elements, setting display labels etc.

6.1 Sub-scenario: deleting a field from the Producer

The Consumer needs to remove portions of the Producer's output which are not relevant to the interaction with the current end-user. For example, in the stock plotting scenario, if that Producer were included in a wealth-management application the Consumer might need to remove the pull-down options for selecting the stock symbols to be plotted, and set them programmatically instead.

Configuration alternatives:

- Non-delegated: the Consumer sets preference options on the Producer controlling the visibility, labeling, and other display-oriented preferences before output generation
- Delegated: The Producer generates output along with metadata describing how the Consumer can optionally turn off selected portions of the page. Note that even in the delegated case, the Producer may need to provide the Consumer with logic (style sheets, JSPs etc) for regenerating the page given the Consumer's preference settings.

6.2 Sub-scenario: adding a field (e.g. form field, table column) to the Producer's output

The Consumer needs to insert elements in the Producer's output to present or gather additional data from the end-user. The Producer integrates this markup with its output. The Producer also determines the acceptable markup restrictions and publishes those to the Consumer.

A further use of this customization is to allow the Consumer to control detailed formatting within Producer-defined cells, such as the individual days of the month in a calendar. In this case, the Consumer would provide the markup to be replicated in the display for each calendar's day, drawing data elements (day number, name, appointments) from the existing Producer data model.

Configuration alternatives:

- Bound to existing Producer data - the output elements added by the Consumer contain data binding expressions allowing them to render existing elements of the Producer's data model
- Not bound or bound to the Consumer's data elements - the output elements added by the Consumer refer to its own data elements (perhaps shadowed by the Producer).

6.3 Sub-scenario: selecting/ordering/naming columns to display in a Producer table

- The Consumer needs to be able to control the visibility, ordering, naming, and other display preferences of Producer's output elements.
- Configuration alternatives:
- The Producer exposes a set of properties to be used by the Consumer in customizing the Producer's display.

7 Use Cases

The scenarios described above can be decomposed into a set of abstract functional patterns, or use cases, described below.

7.1 Data initialization

Presentation data instances are initialized by the Consumer. This may imply alteration of the view if required to render the initial values.

7.2 Interaction automation

The Producer obtains data instance values directly from the Consumer. Views corresponding to the collection steps for these values are skipped.

7.3 Data addition

The presentation data model is augmented by the Consumer. This implies alteration of the view, in order to render/collect the additional data to/from the user.

7.4 Static view alteration

The view of the data model is modified for a specific Consumer.

7.5 Dynamic view alteration

The view of the data model is modified for a specfic Consumer, user, or external configuration (e.g., inventory status, time of day, etc.).

7.6 Out-of-band communication

The Producer sends additional information alongside the presentation markup but intended for the Consumer’s internal use rather than for display to the user (i.e., meta-data).

7.7 Event notification

The Producer asynchronously sends information to the Consumer about a user’s actions.

7.8 Summary

The table below summarizes the mapping of customization scenarios to use cases. Each row represents the union of all scenario alternatives (i.e., not all checked use cases may be valid for all instances of a scenario); some use cases for a scenario may be valid when the customization is Consumer-applied and others when it is Producer-applied.

	Data initialization	Interaction automation	Data addition	Static view alteration	Dynamic view alteration	Out-of-band data communication	Event notification
Memory Configurator							
SKU display			✓				
Availability				✓	✓		
Add To Cart						✓	
Health Enrollment							
Profile automation	✓	✓					
Global restriction				✓			
User-specific restriction					✓		
Sports Portal							
Consumer-driven context		✓					
Producer-driven context							✓
Model Instance Customizations							
Pre-filling data	✓						
Producer sends data to Consumer						✓	
Model Type & Validation Customizations							
Consumer restricts type				✓			✓
Consumer calculates value	✓						✓
Consumer augments type			✓				✓
View Customizations							
Consumer removes fields					✓	✓	
Consumer adds fields			✓			✓	

Consumer adjusts fields (ordering, etc).					✓	✓	

8 Revisions

Date	Author	Description
20 May 2002	TNJ	initial draft