

Multi-Select List

The Multi-Select List (MSL) control provides a convenient set of tools to create, maintain, and display a list values. Depending on the dataset associated with the MSL, the values may be non-object attributes configured by the administrator or business objects (other than User and User Group objects) that are created by the users during the operation of the system.

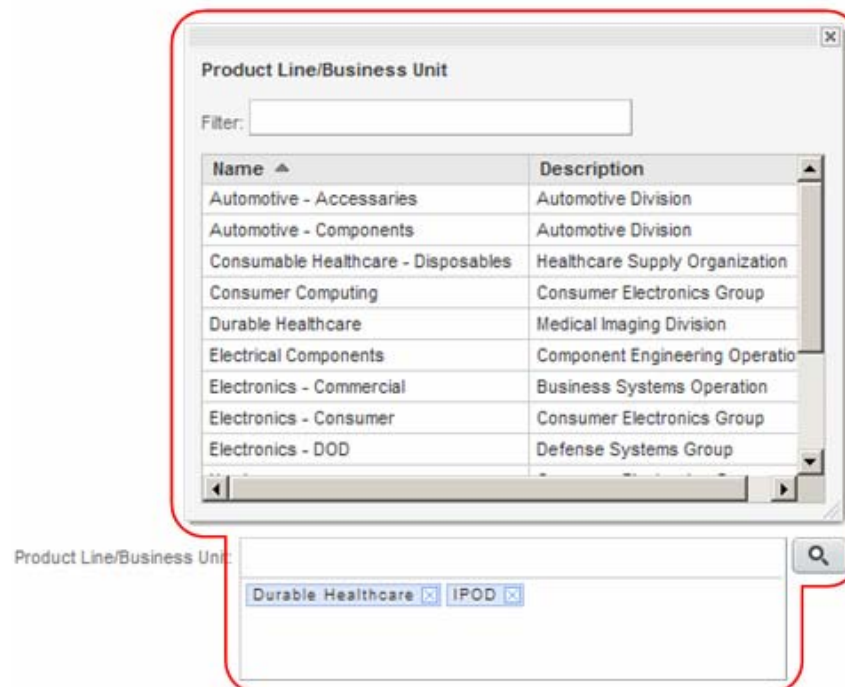
In display mode, object values will be displayed as links which the user can click on to navigate to the page for that object.

In edit mode, all values (object or not) will be displayed as Pills.

The MSL supports Type-Ahead with Auto-Complete targeted at power users who are familiar with the data and who seldom lift their fingers from the keyboard. An easily accessible [Filter Palette](#) provides expanded filtering capabilities to isolate the desired data and is available to all users.

These tools replace the sometimes jarring and invasive use of pop-up windows required to edit lists in prior versions of the Agile user interface.

Element Specification



Multi-Select List (MSL) at a Glance

In the Agile PLM application, attributes (i.e. individual data elements) get their values from lists – either built-in lists of business objects based on Agile or customer defined classes or administratively defined data structures.

A series of administrative decisions during the configuration of the system lead to the selection of the appropriate list control to manage each attribute.

The first of these is the classification of the attribute type as either “List” or “MultiList”. An attribute of type “MultiList” may be assigned a set of one or more values. As new values are assigned, they are added to the set of values already assigned. An attribute of type “List” will accept only a single value. When a new value is assigned, the previous value (if present) is replaced.

This decision is largely predetermined by the business logic behind the attribute and the context in which it is used.

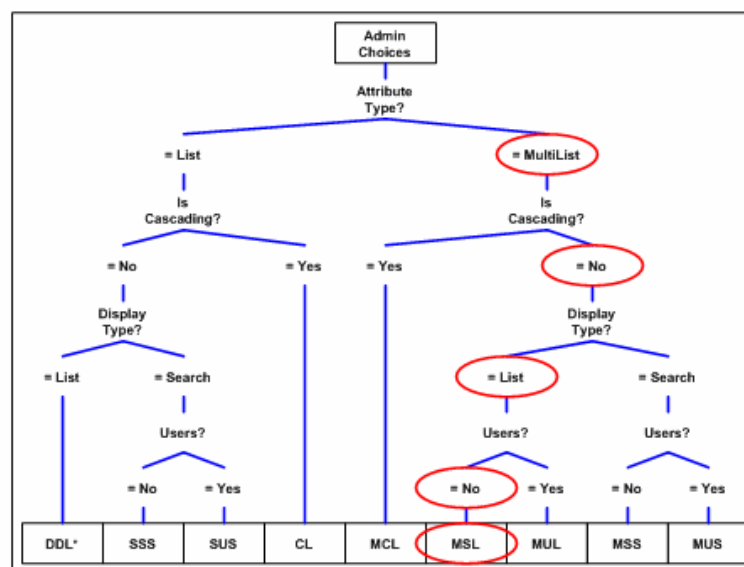
The second decision based on the structure of the data. Is the data flat (such as a customer list) and best presented as a table or is it hierarchically structured data set (such as countries which can be divided up into states or provinces which, in turn, contain cities) that is best presented as a cascade or tree. This decision is made when a new list is created by setting the “Is Cascading?” flag to “Yes” or “No”.

All Agile business objects are, by definition, associated with predefined, non-cascading lists. Administratively defined lists may or may not be cascading. This decision is driven by the nature of the data.

If the list is not cascading, the third decision is based on the amount of data in the list. If the list is relatively short (500 values or less) the administrator may set its display type as “List”. If a single value can be selected, then the browser default Drop-Down List Control will be used. If multiple selections are allowed, a custom pill based control will be used so that the user may take the best advantage of the Agile UI’s Type-In/Auto-Complete and Filter features.

If the list may contain more than 500 values, then its display type must be set to “Search” which will guarantee that all of the values will be accessible. This decision is (more than the others) up to the administrator’s discretion. Where possible, “List” is preferred over “Search” although either will work for short lists.

These decisions (leading to a Multi-Select List) are summarized below:



* DDL = Browser standard Drop-Down List (not one of the 8 custom Agile Pill Based Controls).

Except where noted in this specification, the Multi-Select List complies with the Pill Based Control and Palette UI Specifications.

The building blocks (from which the eight custom Agile Pill Based Controls are constructed) are primarily documented in these specifications. The features applicable to the Multi-Select List are indicated below.

From the Pill Based Control UI Specification	From the Palette UI Specification
<input checked="" type="checkbox"/> Target Element	<input checked="" type="checkbox"/> Controls
<input checked="" type="checkbox"/> Has Initial Focus	<input checked="" type="checkbox"/> Filter
<input checked="" type="checkbox"/> Type-In Area	<input type="checkbox"/> Search
<input checked="" type="checkbox"/> Auto-Complete	<input type="checkbox"/> User/User Group Filter
<input checked="" type="checkbox"/> Immediate Suggestions	<input type="checkbox"/> User/User Group Search
<input type="checkbox"/> Delayed Suggestions Delay:	<input checked="" type="checkbox"/> Data Area
<input checked="" type="checkbox"/> Pill Area	<input checked="" type="checkbox"/> Table Presentation
<input type="checkbox"/> Single Pill	<input type="checkbox"/> Cascade (Tree) Presentation
<input checked="" type="checkbox"/> List of Pills	<input checked="" type="checkbox"/> Selection
<input type="checkbox"/> Height = 1 Row	<input type="checkbox"/> Single Select
<input checked="" type="checkbox"/> Height = 3.5 Rows	<input checked="" type="checkbox"/> Multiple Select
<input checked="" type="checkbox"/> Launch Button	<input checked="" type="checkbox"/> Possible Palette Messaging
<input type="checkbox"/> Has Initial Focus	<input checked="" type="checkbox"/> No Data Available
	<input type="checkbox"/> Pre-Search Prompt
	<input type="checkbox"/> No Records Found
	<input type="checkbox"/> Too Many Records Found
	<input checked="" type="checkbox"/> Insufficient User Privilege
<input checked="" type="checkbox"/> Automatically convert to a Multi-Select Search (MUS) control if the number of entries in the List exceeds 500	

For a high level understanding of the Multi-Select List, this specification will suffice. For a complete and detailed understanding, the above referenced specifications are required reading.

Multi-Select Lists will be used to manage attributes that may be assigned one or more objects (except Users and Special User Lists) or non-object values from a short list (≤ 500 values).

Examples include:

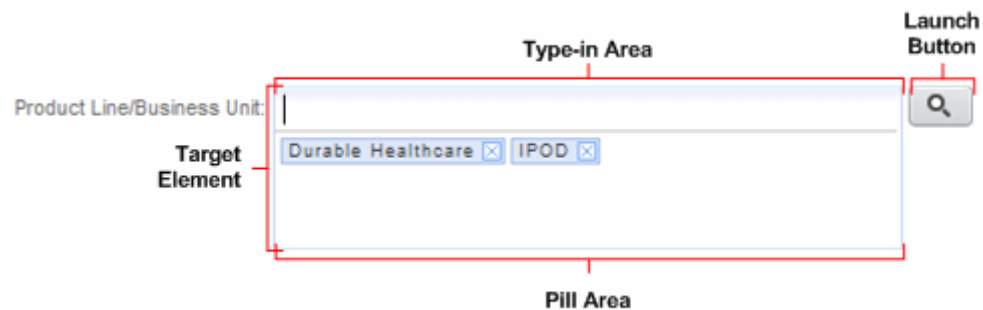
- Customers (object)
- Programs (object)
- Product Lines (non-object)
- Test Failure Modes (non-object)

Form Element

Behavior (Summary)

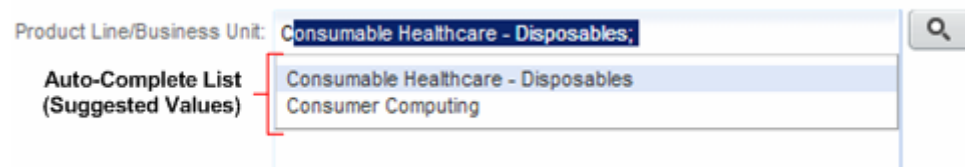
When the user opens a form for editing that contains a Multi-Select List, up to 500 values will be pre-loaded into the client machine. Thus, most of the processing described in the following sections occurs on the client.

The user sets the focus on a Multi-Select List by tabbing to it or clicking on it, or on the label to its immediate left.



The blue focus ring indicates that the Target Element has the focus.

When the user starts typing in the Type-In Area, the Auto-Complete List will appear and the MSL will "type-ahead" in the Type-In Area to match the currently highlighted suggestion. Please note that the user's typed characters are not case sensitive. Thus, typing "c" yields:



The Auto-Complete List may partially (as in this case) or fully cover the Pill Area. It will first appear at full length (up to a maximum of 20 suggestions) and become shorter as additional characters are typed and non-matching suggestions are eliminated.

The user may take advantage of this list to help with data entry by using the up and down arrow keys to move the highlight and right arrow to accept the highlighted selection.

Multiple values may be entered in a semi-colon separated list. In this example, the user has typed a right-arrow key to accept the top suggestion and has started to type an entry that is not defined in the database.



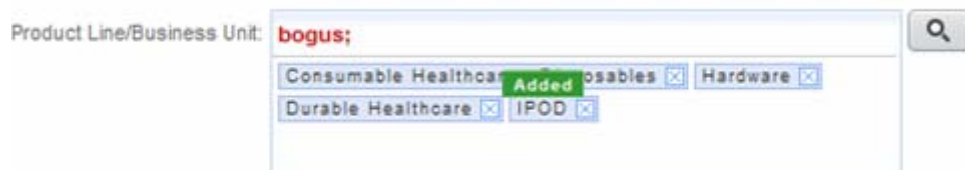
When the last entry is invalid (e.g. "bogus" above), the entire Type-In Area will be displayed the Agile **Bold Error** Font to indicate the failure to find any matching values.

The Auto-Complete feature only applies to the last value in the list.

As long as the last value is valid (i.e. matching suggestions exist) the contents of the Type-In Area will be displayed in the Agile **Normal** Font even if it contains erroneous content as shown below.



When the user triggers the validation process, valid values are converted to pill format and added to the list of pills in the Pill Area. A green "Added" indicator will flash briefly over the Target Element to confirm this process. Invalid data will be left in the Type-In Area in the Agile **Bold Error** Font.



Note that the new values are added to the front of the list in the Pill Area. If a value is already in the list, it is not moved to the front. The list will be sorted when the form is saved.

The details of the above process are fully documented in the [Pill Based Controls UI Specification](#).

Please see the [Fonts](#) UI Specification for the description of the **Normal** and **Bold Error** Font.

Type-ahead and auto-complete provide an efficient mechanism that allows the power-user to enter a great deal of data quickly without removing hands from the keyboard. This mechanism starts to break down when the following conditions occur:

- The user is not familiar with the leading characters in the name of the data elements of interest.
- The leading characters in these names are not sufficiently unique requiring more characters to be typed correctly before the Auto-Complete List starts to narrow.
- The list is long, thus providing more possible matches and delaying the narrowing of the list.

When these conditions occur, the user can fall back on the more powerful features of the Filter Palette as outlined in the following section.

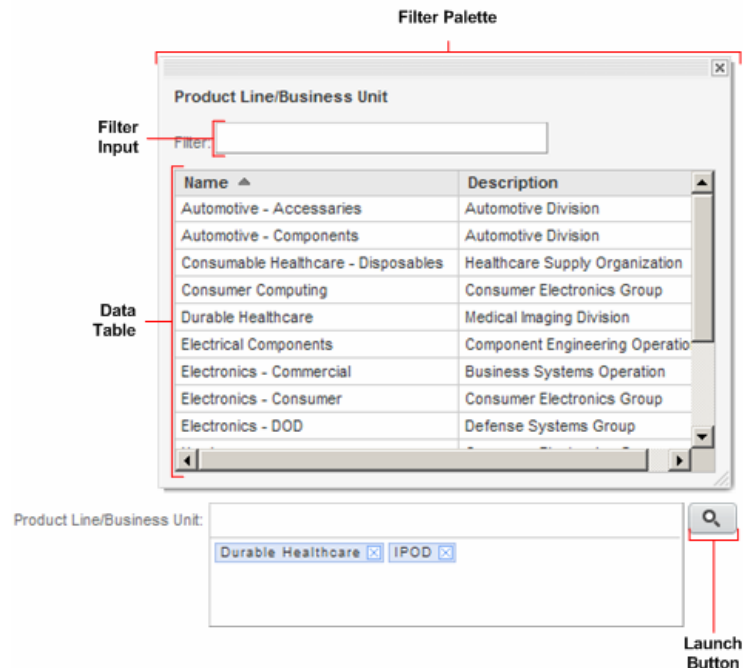
Layout, Appearance and Styling

The Layout, Appearance and Styling of the MSL Form Element comply with the Forms UI Specification.

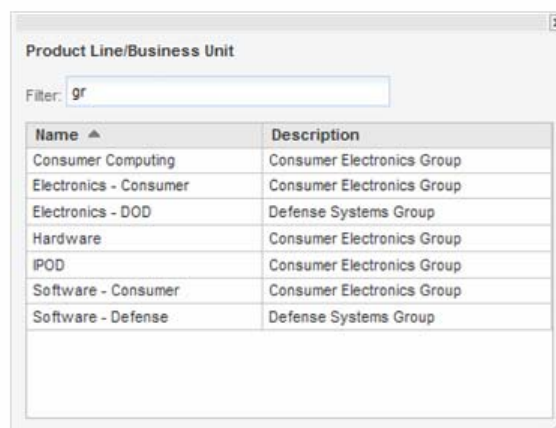
Filter Palette

Behavior (Summary)

The user may press the Launch Button to bring up a Filter Palette. The Filter Palette uses the same list of up to 500 preloaded list elements.

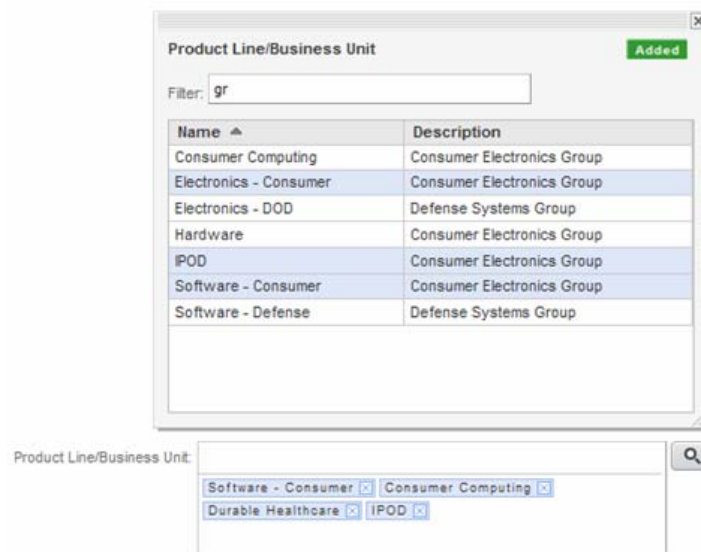


Typing into the Filter Input will reduce the list of possible values by applying a more powerful “contains” matching algorithm to all of the columns in the Data Table rather than the faster “starts with” algorithm that Auto-Complete applies the Name field only. In this case, the user remembers that the product lines of interest are managed by a group (as opposed to an organization or division). Typing “gr” into the Filter Input reduces the list and isolates the desired elements.



The user may also sort the data in ascending or descending order by any column (not shown).

The user may then select one or more rows and send these rows back to the Target Element for validation and addition to the list of pills in the Pill Area. An “ADDED” indicator will flash to confirm this process – either in the Target Element (after drag-and-drop) or in the upper right-hand corner of the palette (after double-click or Enter key).



The details of the above behaviors are fully documented in the [Palette](#) UI Specification.

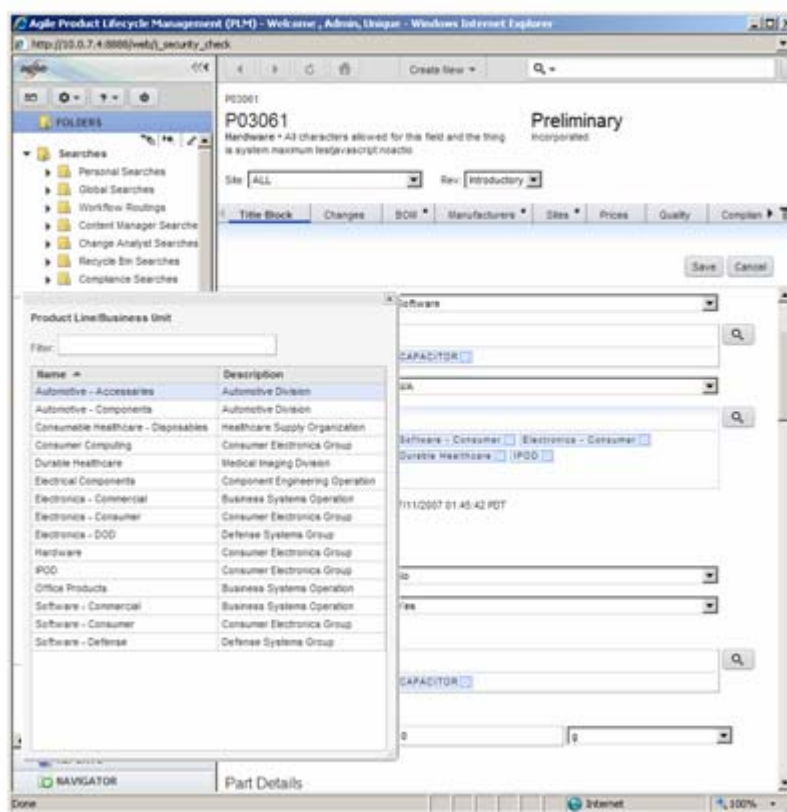
Layout, Appearance and Styling

The Layout, Appearance and Styling are described in the [Palette](#) UI Specification.

Palette Persistence

Behavior (Summary)

Once opened, the Palette may be resized and moved to a more convenient location on the screen or even to a different display if the user dual displays.



The user may then continue to edit the form.

If the palette has been moved or resized within a page edit session, it will always reopen at the location and size specified by the user, thus the user will always know where to find it. If the user saves or cancels this edit and edits a new page, the palette will be launch in its default location relative to the Target Element.

When the user sets the focus on another Pill Based List Control, the contents of the open palette will change to serve the new Target Element as shown below.

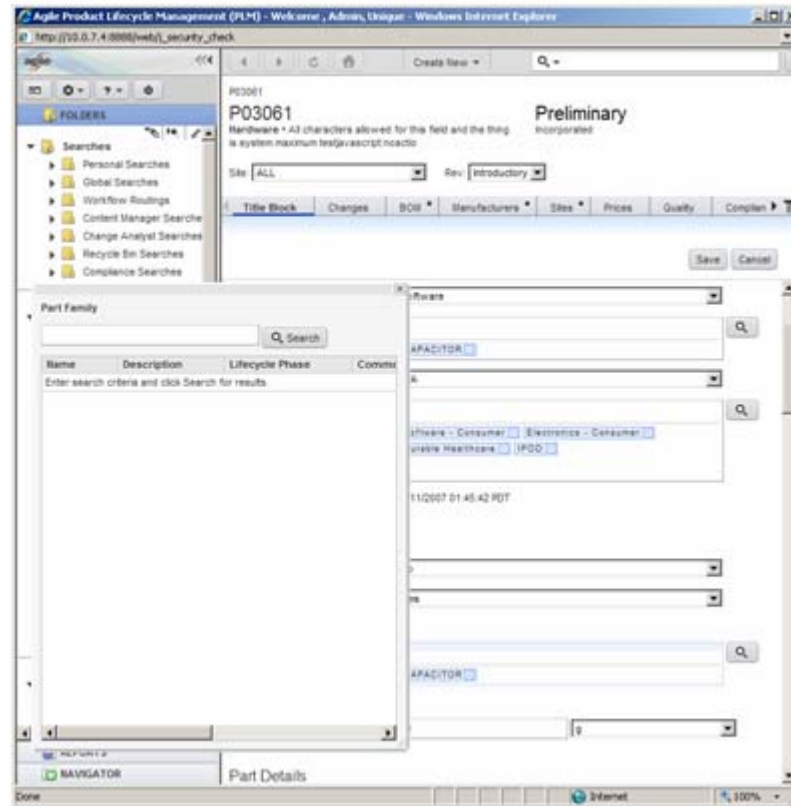


Table Cell

The Multi-Select List as applied to table cells is similar to the MSL as applied to form elements.

- While not in edit mode, list elements are displayed in semi-colon separated lists; objects are display as links; and non-objects are displayed as text.
- While in edit mode, an edit frame surrounds the table cell and contains a Type-In Area, a Pill Area and a Palette Launch Button. Values assigned to the cell are displayed as Pills in the Pill Area.
- Type-In and Auto-Complete work the same way, as does the Filter Palette.

There are some differences as well.

- While all of the elements in a form are in edit mode while the form is in edit mode, each cell in a table is opened separately.
- The Palette will serve all of the cells in a column. Thus, the user may work from cell to cell up or down a column while the contents of the palette remain the same. It is only when the user edits a cell in a different column that the palette will be updated to serve that column.

There will, no doubt, be more to say after the table comes on-line. For now, moving on.

Issues

- 1) Cannot be finalized until the implementation of MSL in tables is understood and operational to a level that can provide screenshots.

Change History

November 21, 2007	Vern McGeorge	Complete. Revised to incorporate decision to convert if more than 500 values into At a Glance diagram.
November 6, 2007	Vern McGeorge	All sprint 2.6 comments except Kanda's in.
October 8, 2007	Vern McGeorge	Published for final review.
October 4, 2007	Vern McGeorge	Finalized style for At a Glance diagram.
September 27, 2007	Vern McGeorge	First draft of new, specific MSL spec.
June 12, 2007	Vern McGeorge	Existing MSL spec generalized into new Pill Based Controls Specification