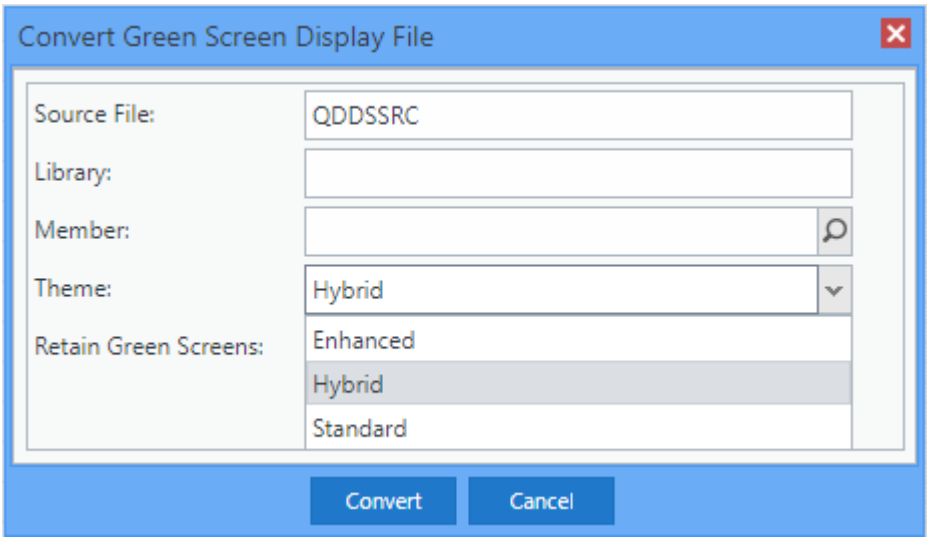
Themes

A theme is a collection of DDS conversion rules. The theme is selected by its name from a dropdown list on the DDS Conversion Dialog.



While Profound UI is shipped with several prebuilt themes, it is often necessary to create a custom theme in order to modify the rules used by the conversion process.

Theme definitions are stored in the IFS directory /www/profoundui/htdocs/profoundui/userdata/custom/themes. Each theme is defined in its own .js file containing a list of properties in JSON (JavaScript Object Notation) format. To create a custom theme, copy an existing theme file to a new file name and then modify the JSON properties inside the file. Make sure to modify the "name" property to reflect the new theme name.

The following properties are available:

- **name** - the name of the theme; should generally be the same as the file name in which the theme is defined
- **input css class** - the class applied to input elements (DDS field types "B" and "I") created by the DDS conversion
- **alt input css class** - optional css class width for 132 wide screens
- **input css class 2** - the second class applied to input elements (DDS field types "B" and "I") created by the DDS conversion
- **constant css class** - the class applied to elements created from DDS constants
- **altconstantcss class** - optional css class width for 132 wide screens
- **output field css class** - the class applied to output
  fields (DDS field type "O") and also to fields with DATE, TIME, USER, and SYSTEM keywords
- **alt output field css class** - optional css class width for 132 wide screens
- **always set width** - by default, output fields are not given a width so that the browser will automatically resize them to the size of the content within them. If you set this option to true, a fixed width will be set on output fields during DDS conversion
- **css class prefix** - the DDS conversion automatically maps DDS display attributes and colors to CSS classes with equivalent names, such as "HI" for High Intensity or "BLU" for Blue; this property allows you to prepend a prefix to each of the generated class names
- **top line color** - optional property that specifies a color to assign to all header elements, or elements that appear on row 1 of a screen
- **top line css class** - optional property that specifies a CSS class to assign to all header elements, or elements that appear on row 1 of a screen
- **default font family** - optional font family to use on elements during the conversion; if not specified, the font family is picked up from the CSS
- **default font size** - optional font size to use on elements during the conversion; if not specified, the font size is picked up from the CSS
- **default locale** - sets the locale for date, time, or timestamp fields. Also used in Edit Code processing to assign appropriate formatting on field binding
- **left offset** - offset from the left side of the screen in pixels for all converted elements; this allows you to create an area for a left sidebar with function key buttons
- **alt left offset** - optional left offset width for 132 wide screens
- **top offset** - offset from the top side of the screen in pixels for all converted elements
- **column width** - determines how wide the space equivalent to a green-screen column should be in pixels
- **alt column width** - optional column width for 132 wide screens.
- **row height** - determines how tall the space equivalent to a green-screen row should be in pixels
- **alt row height** - optional row height for 132 wide screens
- **extra textbox width** - the width of textboxes generated by the conversion process is determined by multiplying the "column width" property by the character width of the green-screen field; this property allows you to add a few extra pixels to the textbox width
- **auto arrange** - specified the "auto arrange" property of a function key button, the default is true
- **button type** - the type of button to use for function key buttons and for GUI DDS elements, such as those created from the PSHBTNFLD keyword; valid values are "button", "graphic button", "styled button", "css button", or "hyperlink"
- **button style** - if a styled button is used for the "button type" property, this specifies the button style to use
- **button theme** - if a css button is used, this specifies the button theme to use
- **button straight edge** - if a css button is used, this specifies a value for the "straight edge" property
- **button width** - the standard width of a function key button
- **button height** - the standard height of a function key button
- **button css class** - optional CSS class of a function key button
- **button tab index** - optional tab index of a function key button
- **button left offset** - optional left offset in pixels for function key buttons
- **alt button left offset** - optional left offset in pixels for function key buttons, when the screen is a 132x27 screen

- **button top** - optional top position in pixels of the first generated function key button
- **horizontal button spacing** - the horizontal space between each button when buttons are arranged horizontally
- **vertical button spacing** - the vertical space between each button when buttons are arranged vertically or when buttons wrap from one row to the next
- **buttons per row** - number of buttons to arrange horizontally before a new row of buttons is created
- **add submit button** - true or false value indicating whether a Submit button should be added to the screen; clicking the Submit button is similar to hitting the Enter key
- **help button** - true or false value indicating whether the a Help button should be added to the screen and associated with the DDS HELP keywords; the default is true
- **submit button text** - optional value to override the text that will appear on the Submit button; if not specified the word "Submit" will be used
- **buttons start at the top** - determines if buttons are placed at the top of a screen going down or if they begin at the bottom of a screen
- **show fkey name** - determines if the name of the function key (i.e. F1, F2, F3, etc.) shows on the button or hyperlink
- **show fkey name as tool tip** - determines if the name of the function key (i.e. F1, F2, F3, etc.) should appear as a tool tip when the user hovers over the button or hyperlink
- **show fkey text** - determines if the function key text shows on the button; the text is retrieved from the keyword that defined the function key or a constant function key label that appears in the converted Display File
- **page down text** - default text for the Page Down button
- **page up text** - default text for the Page Up button
- **remove option labels** - true or false value indicating whether to automatically remove option labels, such as 2=Change or 5=Display
- **remove fkey labels** - true or false value indicating whether to automatically remove function key labels, such as F3=Exit or F5=Refresh
- **use aliases** - true or false. If true: If an ALIAS keyword is found, LONGNAMEALIASES will be enabled and the parameter of the alias keyword will be used as the field name. Defaults to true.
- **use fkey labels** - true or false value indicating whether constant function key labels, such as F3=Exit or F5=Refresh, are used to determine the text to display on the buttons or hyperlinks created for function keys; defaults to true
- **use fkey label indicators** - true or false value indicating whether conditioning indicators on labels, such as F3=Exit or F5=Refresh, should be applied to the buttons or hyperlinks created for function keys; defaults to false
- **fkey suffix char** - optional character to consider when looking for function key labels; for example if "x" is specified, F3x=Exit is a valid function key label
- **allow field exit** - determines if the field exit key should be allowed on input elements generated by the conversion; this can later be changed by modifying the "allow field exit" property on the widget
- **onload** - optional onload event, specified as a string, that will be applied to every converted screen
- **process field** - optional JavaScript function that allows you to add custom processing to the fields in the conversion; the function will receive 4 parameters:
    - **field** - field object with the following properties
        - **name** - name of the field
        - **formatName** - the name of the DDS record format to which the field belongs
        - **memberName** - the name of the DDS source member being converted
        - **const** - constant text if the field is a constant
        - **field type** - DDS field type, such as "B" for both, "I" of input, "H" for hidden, or "O" for output
        - **data type** - DDS data type
        - **length** - field length
        - **decimal** - decimal position if the field is numeric
        - **row** - original DDS row position of the field
        - **col** - original DDS column position of the field
        - **keywords** - array of DDS keywords, each array entry having the following properties:
            - **name** - keyword name
            - **parms** - array of keyword parameters
            - **line** - original DDS line number
    - **item** - item object being created by the conversion; the properties of the item are identical to those found in the Visual Designer; your JavaScript code can set the properties on the item as desired
    - **subfile flag** - true or false value indicating whether the field belongs to a subfile
    - **window flag** - true or false value indicating whether the field belongs to a window
- **process fkey** - optional JavaScript function that allows you to add custom processing for function key buttons or links created by the conversion; the function will receive 3 parameters:
    - **keyword** - object representing the DDS keyword for the function; the object will contain the following properties:
        - **name** - keyword name
        - **parms** - array of keyword parameters
    - **item** - item object for the link or button being created by the conversion; the properties of the item are identical to those found in the Visual Designer; your JavaScript code can set the properties on the item as desired
    - **format** - object representing the rich display record format being created; it will contain the following properties:
        - **screen** - screen-level properties, such as "record format name" assigned by the conversion; the screen-level properties are identical to those found in the Visual Designer
        - **items** - array of all item objects created by the conversion; the properties of the items are identical to those found in the Visual Designer
    - **member** - the DDS source member name that is being converted
- **fkey sort** - optional custom JavaScript function allowing you to specify the sort order of function keys; if not specified, the sort order is by key name (i.e. F1, then F2, then F3, etc.); the custom function will be called repeatedly and will receive 2 fkey keyword parameters to compare; it must return -1 if the first keyword should come before the second keyword, or 1 if the first keyword should come after the second keyword; each keyword will have the following properties:
    - **name** - keyword name
    - **parms** - array of keyword parameters
    - **line** - original DDS line number
- **add enhancements** - optional JavaScript function that allows you to add custom enhancements to the screen after each format is converted; the function receives the following parameters:
    - **format** - representation of the new rich display record format; contains the following properties:
        - **screen** - screen level name/value pairs for record format properties; the properties are identical to those found in the Visual Designer; your JavaScript code can modify or set new properties as desired

- **items** - array of items created by the conversion; each item contains a list of properties that are identical to those found in the Visual Designer; your JavaScript code can set the properties on the items as desired or add new items to the array
- **subfile flag** - true or false value indicating whether the record format is a subfile
- **window flag** - true or false value indicating whether the record format is a window
- **member** - the DDS source member name that is being converted
- **formatDDS** - a representation of the DDS keywords and fields in the format; contains the following properties:
  - **keywords** - an array of format level keywords; each keywords will contain the following properties:
    - **name** - keyword name
    - **parms** - array of keyword parameters
    - **line** - original DDS line number
  - **fields** - an array of fields in the format; each field will contain the following properties:
    - **name** - name of the field
    - **formatName** - the name of the DDS record format to which the field belongs
    - **memberName** - the name of the DDS source member being converted
    - **const** - constant text if the field is a constant
    - **field type** - DDS field type, such as "B" for both, "I" of input, "H" for hidden, or "O" for output
    - **data type** - DDS data type
    - **length** - field length
    - **decimal** - decimal position if the field is numeric
    - **row** - original DDS row position of the field
    - **col** - original DDS column position of the field
    - **keywords** - array of DDS keywords, each array entry having the following properties:
      - **name** - keyword name
      - **parms** - array of keyword parameters
      - **line** - original DDS line number

- **grid enhancements** - optional JavaScript function that allows you to perform custom subfile enhancements for each converted subfile; the function receives the following parameters:
  - **control record** - representation of the new rich display control record format; contains the following properties:
    - **screen** - screen level name/value pairs for record format properties; the properties are identical to those found in the Visual Designer; your JavaScript code can modify or set new properties as desired
    - **items** - array of items created by the conversion; each item contains a list of properties that are identical to those found in the Visual Designer; your JavaScript code can set the properties on the items as desired or add new items to the array
  - **grid item** - name/value pairs for the grid item; the properties are identical to those found in the Visual Designer; your JavaScript code can modify or set new properties as desired
  - **options** - array of subfile options, such as 2=Change or 5=Display, that have been found by the conversion process on the screen; each option is an object consisting of the following properties:
    - **option** - represents the option number, such as 2 or 5
    - **text** - represents the option text, such as "Change" or "Display"
  - **member** - the DDS source member name that is being converted
- **grid** - object that determines how subfiles and grids are handled during the DDS conversion; the following options are available:
  - **properties** - list of miscellaneous properties that should be assigned to the Grid widget; the properties are identical to those found in the Visual Designer
  - **top offset** - offset specified in pixels to add to the top position of each element within the grid
  - **left offset** - offset specified in pixels to add to the left position of each element within the grid
  - **extra width** - extra width specified in pixels to add to the grid width that is automatically calculated based on subfile columns
  - **header rows** - this value determines the height of of the grid header in rows; the value can be specified as the number of rows,such as 0, 1, or 2; the value can also be set to "auto" to allow the DDS conversion to automatically determine the number of header rows based on constant fields present in the control record format or in the format returned by the function specified in the getHeaderFormat property; the value can also be specified as a custom function that receives control record and the subfile record as two separate parameters and returns the number of header rows
  - **merge headings** - this true or false value determines if heading fields are merged into the grid component itself using the grid's "column headings" property; merging the headings may provide a nicer look and may be easier to maintain; however, the positioning of the headings may not be accurate at first
  - **set heading font** - when set to true, the font properties or the "css class" property on heading output fields is set to match the standard grid headings; this is only done on heading output fields that were not merged or could not be merged into the grid heading
  - **header field class** - the css class to use on heading fields
  - **message subfile class** - the css class assigned to the grid when the grid represents a message subfile
  - **show options as dropdown** - when set to true, the DDS conversion will look for option labels, such as 2=Change or 5=Display, and create a dropdown out of these options in the first column of the grid instead of using the standard textbox option field
  - **show options as combo** - when set to true, the DDS conversion will look for option labels, such as 2=Change or 5=Display, and create a combo box out of these options in the first column of the grid instead of using the standard textbox option field
  - **show options as context menu** - when set to true, the DDS conversion will look for option labels, such as 2=Change or 5=Display and create a context menu. You cannot have context menus and dropdowns or combo boxes, these options are mutually exclusive.
    - **context menu css class** - This option assigns a CSS class to the created menu widget, you will need to set up a CSS class to handle it. This option is required.
    - **context menu hover image** - Optional property used to populate the “option hover image” property of the menus.
    - **context menu option indent** - Optional property used to populate the “menu option indent” property of the menus.
  - **single row zoom** - if set to true, the DDS conversion will enable the single row zoom feature on folding subfiles; single row zoom allows users to expand collapsed rows one at a time using a zoom icon
  - **sortable** - optional true or false property; enables the "sortable columns" feature on load-all subfiles; if omitted, true is assumed
  - **find** - optional true or false property; enables the "find option" feature on load-all subfiles; if omitted, true is assumed
  - **filter** - optional true or false property; enable the "filter option" feature on load-all subfiles; if omitted, true is assumed
  - **expander icon** - optional true or false value that determines if an expander icon should be added for grids with fold/unfold capabilities; if omitted, true is assumed
  - **expander z index** - expander icon z index value that is created for folding subfiles
  - **fold button** - optional true or false value that determines whether a standard function key button or hyperlink should be added for folding and unfolding grid data; if omitted, false is assumed

    - **csv export** - optional true or false property; enables the "csv export" feature on load-all subfiles; if omitted, false is assumed
    - **xlsx export** - optional true or false property; enables the "xlsx export" feature on subfiles; if omitted, false is assumed
    - **export with headings** - optional true or false property used together with "csv export" or "xlsx export"; sets the "export with headings" property on load-all subfiles; if omitted, false is assumed
    - **getHeaderFormat** - optional function that receives a subfile control record format name and returns the name of another record format that contains the headings for the subfile; intended for DDS code that stores subfile headings outside of the control record; ideally, there would be a predictable naming convention for such record formats
- **subfile option pattern** - use a regular expression to match subfile options instead of the default mode, which looks for patterns like "someNum=some text". Example: /([A-Z0-9]{1,2}) ?(=) ?(\S+ *)/.
- **subfile option pattern greedy** - set to false to prevent "subfile option pattern" from capturing text between matches. By default, the "subfile option pattern" algorithm looks for text between each match on a field and up to the end of a field; any found are added to the option text. For example, a field including the text, "9=Work with Print Status", would only get the text "Work" if "subfile option pattern greedy" is false, and the option text would get the entire "Work with Print Status" if this option is omitted or not false.
- **items** - array of widgets, such as background panels, to add to each screen; each widget is an object of Visual Designer properties; you should at least specify the "id", "field type", "left", "top", "height", and "width" property; special properties "alt_width" and "alt_height" can specify the height and width for elements to be created on a screen that was originally a 132x27 character screen
- **windows** - object that determines how windows are handled during the DDS conversion; the following options are available:
    - **css class** - sets the "css class" property on the generated panel widget
    - **panel type** - green-screen windows are converted to formats that utilize a panel or a dialog; this option specifies the "field type" of the panel widget; at this time, only "css panel" and "panel" are valid values for this option; if the option is omitted, the panel type defaults to "panel". **Note**: If a panel type is required that is not supported here, the type can be set to "panel", then in the add enhancements function the panel will appear in the format.items array and the properties can be changed as required.
    - **panel style** - specifies the panel style for the dialog or panel when the panel type is "panel"
    - **header theme** - optional header theme of a "css panel" dialog; if not specified, the default theme of "B - Blue" is used
    - **body theme** - optional body theme of a "css panel" dialog; if not specified, the default theme of "C - Gray" is used
    - **has header** - optional "has header" property of a "css panel"; valid values are "true" or "false"; if omitted, the css panel will by default have a header
    - **header height** - optional value to specify the height of a "css panel" header; if omitted, a default header height is used
    - **straight edge** - optionally specifies which part of the"css panel" will have a straight edge instead of rounded corners; valid values are "all", "left", "right", "top", or "bottom"
    - **title text align** - the default text alignment for the dialog title
    - **top offset** - offset from the top side of the screen in pixels for all converted elements within the window
    - **bottom offset** - additional height in pixels to add to the created window dialog
    - **left offset** - offset from the left side of the screen in pixels for all converted elements within the window
    - **right offset** - additional width in pixels to add to the created window dialog
    - **add submit button** - optional true or false value indicating whether a Submit button should be added to the screen; clicking the Submit button is similar to hitting the Enter key; if omitted, the value is inherited from the top level "add submit button" property
    - **button type** - optional; the type of button to use for function key buttons and for GUI DDS elements, such as those created from the PSHBTNFLD keyword; valid values are "button", "graphic button", "styled button", or "hyperlink"; if omitted, the value is inherited from the top level "button type" property
    - **button css class** - optional CSS class of a function key button within the window; if omitted, the value is inherited from the top level "button css class" property
    - **button tab index** - optional tab index of a function key button within the window; if omitted, the value is inherited from the top level "button tab index" property
    - **button left offset** - optional left offset in pixels for function key buttons within the window
    - **alt button left offset** - optional left offset in pixels for function key buttons within the window, when the screen is a 132x27 screen
    - **button offset** - top position offset from the bottom of the window in pixels for function key buttons created within the window
    - **button width** - the standard width of a function key button in the window
    - **horizontal button spacing** - the horizontal space between each button within the window
    - **vertical button spacing** - the vertical space between buttons when buttons have to wrap to the next row within the window
    - **buttons per row** - number of buttons to arrange horizontally before a new row of buttons is created within the window; can be specified as "auto", in which case the value is automatically determined based on the width of the window
    - **items** - array of widgets to add to each window; each widget is an object of Visual Designer properties; you should at least specify the "id", "field type", "left", "top", "height", and "width" property

    The order in which the optional JavaScript functions are called, is as follows:
    1. For each record format, **process field** is called for each field, followed by **add enhancements**
    2. **process fkey**
    3. **grid enhancements**