Profound Logic Product Updates

April 2019

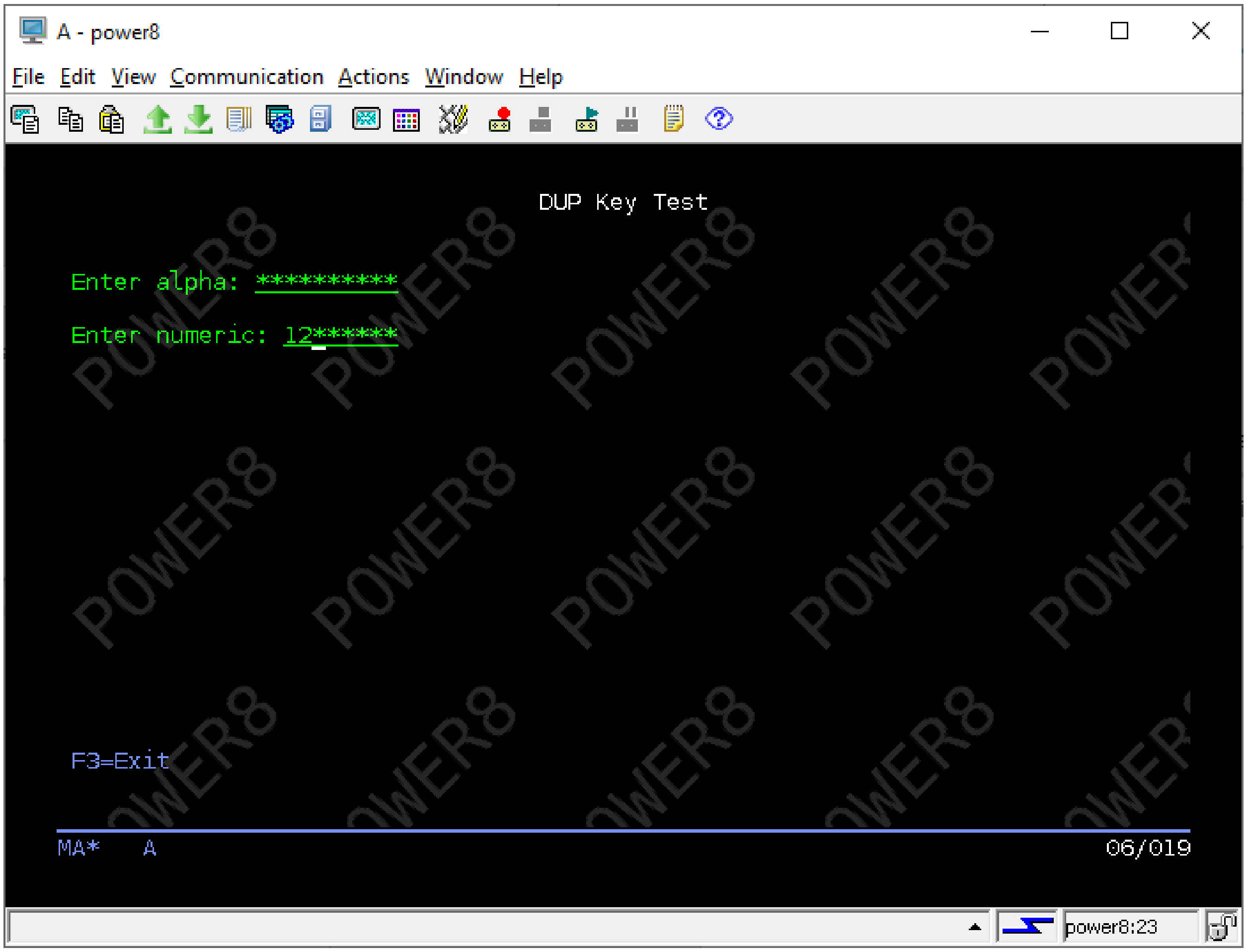
Download the latest versions at: www.profoundlogic.com/download

Profound UI 6 Fix Pack 3.0/3.1 Enhancements:

Genie Now Supports the 5250 DUP Keyword

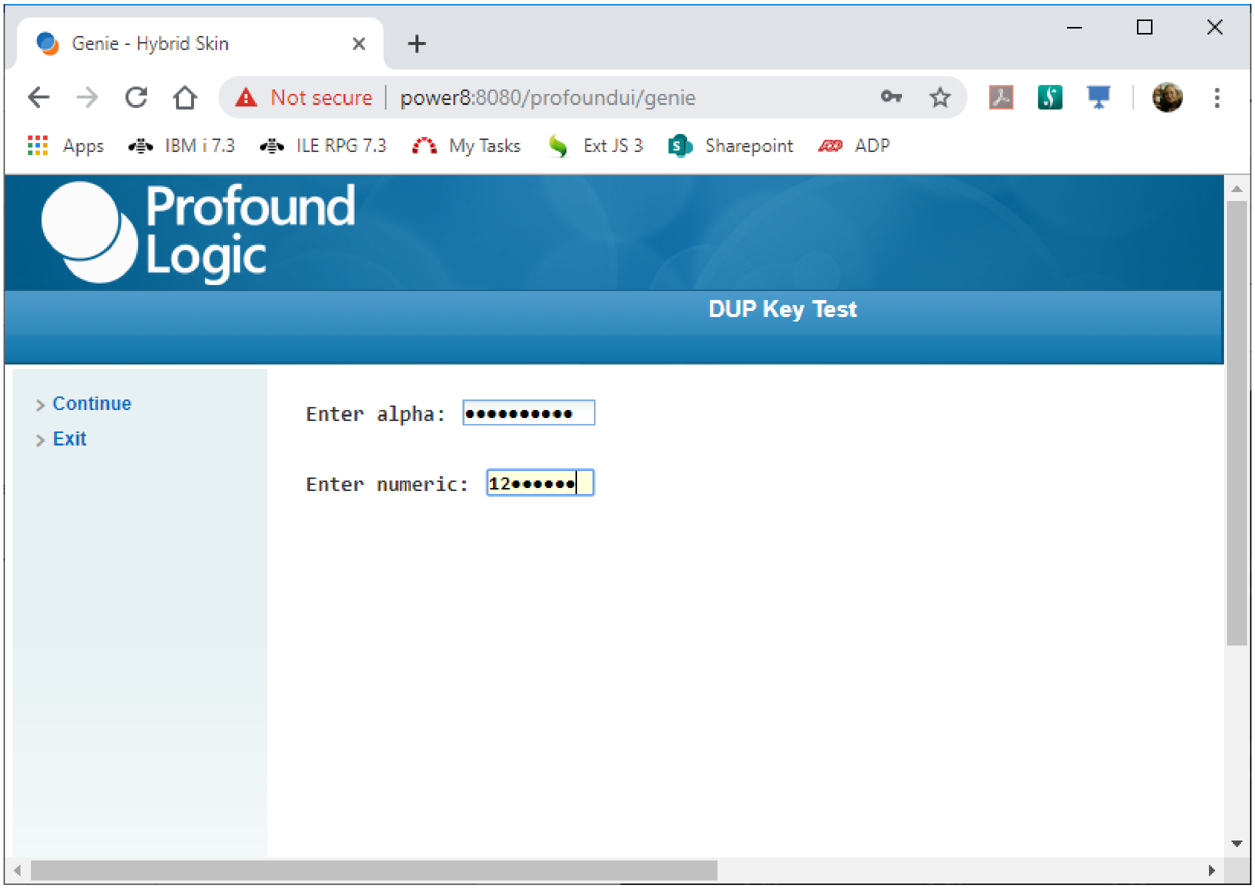
Some green screen applications use a field-level keyword called DUP to enable quick duplication of entry fields. When the user presses the DUP key on a DUP-enabled field, all positions in the field after where the cursor is positioned are filled with asterisks.

It looks like this:



```
                         DUP Key Test

  Enter alpha: **********

  Enter numeric: 12******




  F3=Exit

MA*    A                                              06/019
```

power8:23



DUP Key Test

Enter alpha: ●●●●●●●●●●

Enter numeric: 12●●●●●●

When the screen is submitted to the program for processing, the program can check if fields have been DUP-ed by looking at a response indicator (for any type of field), or by checking the field data for hex code 1C in place of the asterisks (for character fields only).

What happens next is entirely up to the programming logic. Usually a program will process this by storing non-DUP user input in hidden fields as the user is entering records. Then when the user DUPs a field, the program automatically populates the field with the last non-DUP value that the user keyed.

The default key mapping for DUP is Shift+Insert, but can be customized using these configuration options:

http://www.profoundlogic.com/docs/display/PUI/Dup+Key

Genie: Browser Handles Home Key

In fix pack 2.0, the home key was updated to work the same way in Genie as it would in other 5250 emulators. However, some customers preferred our old method of letting the browser handle the home key, in a more web-like fashion. In fix pack 3.0, we added a configuration option that you can use to restore Genie's original behavior.

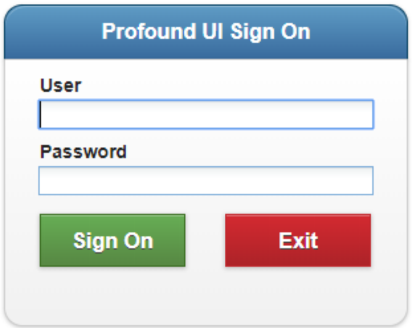
http://www.profoundlogic.com/docs/display/PUI/Genie+Browser+Handles+Home+Key

More Modern Styling in PUISCREENS

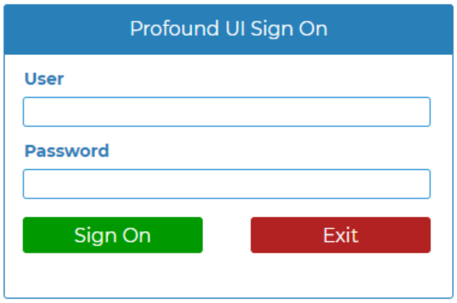
We have recently updated all the record formats within the PUISCREENS to use the same flat look theme that we've been shipping in the Blueprint widget set.

Here are a couple screen shots so that you can see the differences.

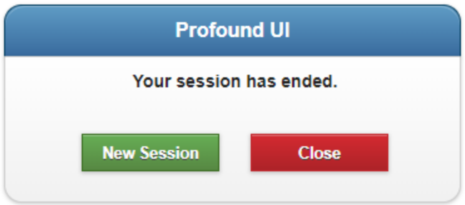
Before



After

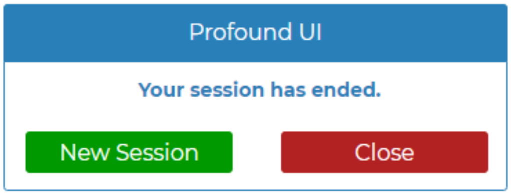


Before



Your session has ended.

After



Your session has ended.

Profound UI does not automatically replace your PUISCREENS when upgrading an existing installation. Existing customers, if you want to take advantage of the new look, you can recompile the PUISCREENS display file from the Profound UI 6 fix pack 3 source code by running the CRTDSPF command.

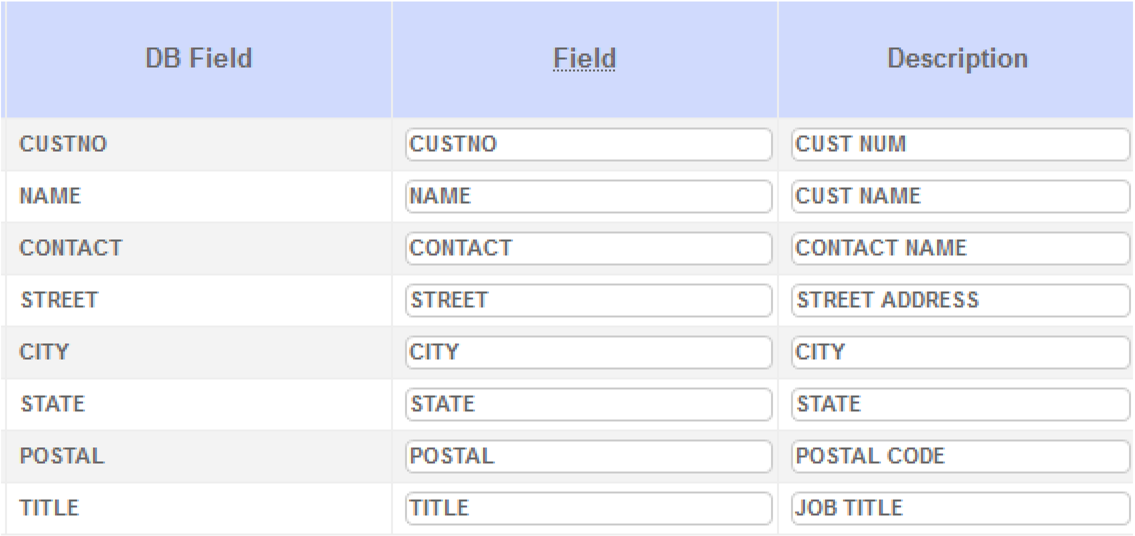
JumpStart Field Description Transform Function

We added a hook within the JumpStart generation process that gives you the ability to override the generated table descriptions.

Some reasons you would want to use this hook:

1- Change description to lower case
2- Fix Spelling
3- Add/Remove/Fix abbreviations

For example, it's not unusual in IBM i shops to have database tables with field descriptions that are abbreviated or all uppercase like this one:



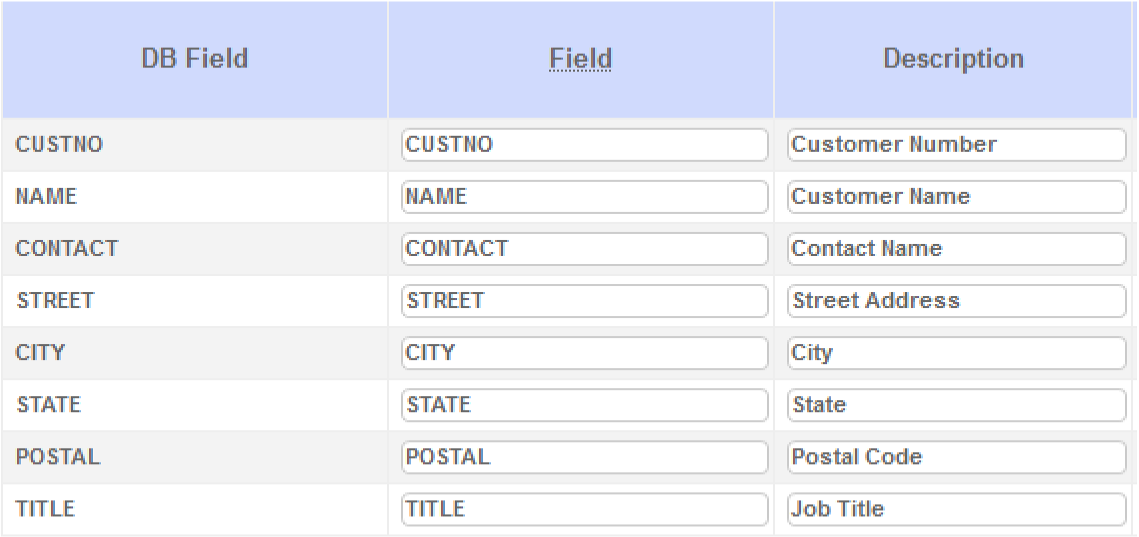
DB Field	Field	Description
CUSTNO	CUSTNO	CUST NUM
NAME	NAME	CUST NAME
CONTACT	CONTACT	CONTACT NAME
STREET	STREET	STREET ADDRESS
CITY	CITY	CITY
STATE	STATE	STATE
POSTAL	POSTAL	POSTAL CODE
TITLE	TITLE	JOB TITLE

With this new feature, you can write JavaScript code that transforms these descriptions. In this case, make the description lowercase, but uppercase the first letter of each word. Also, convert abbreviations like ‘num’ and ‘cust’ to full words like ‘number’ and ‘customer’.

```
pui.DbDescriptionTransform = function(fieldDesc) {
  var lowerDesc = fieldDesc.toLowerCase();
  lowerDesc = lowerDesc.replace(/cust\b/, "customer");
  lowerDesc = lowerDesc.replace(/num\b/, "number");
  lowerDesc = lowerDesc.replace(/\b./g, function(a) {
                return a.toUpperCase();
              });
  return lowerDesc;

}
```

The \b code in the regular expressions means ‘word boundary’.

The result looks like this:



DB Field	Field	Description
CUSTNO	CUSTNO	Customer Number
NAME	NAME	Customer Name
CONTACT	CONTACT	Contact Name
STREET	STREET	Street Address
CITY	CITY	City
STATE	STATE	State
POSTAL	POSTAL	Postal Code
TITLE	TITLE	Job Title

To use this hook: http://www.profoundlogic.com/docs/display/PUI/Custom+JavaScript+Files

Convert Grid Row Numbers to RRNs and Vice-Versa

We added some new Grid methods that allow you to retrieve the Row Number for a given RRN and retrieve the RRN for a given Row Number. With these methods, you could mimic an Excel *like* experience (such as using the Arrow keys up/down to move to the cells directly above or below).

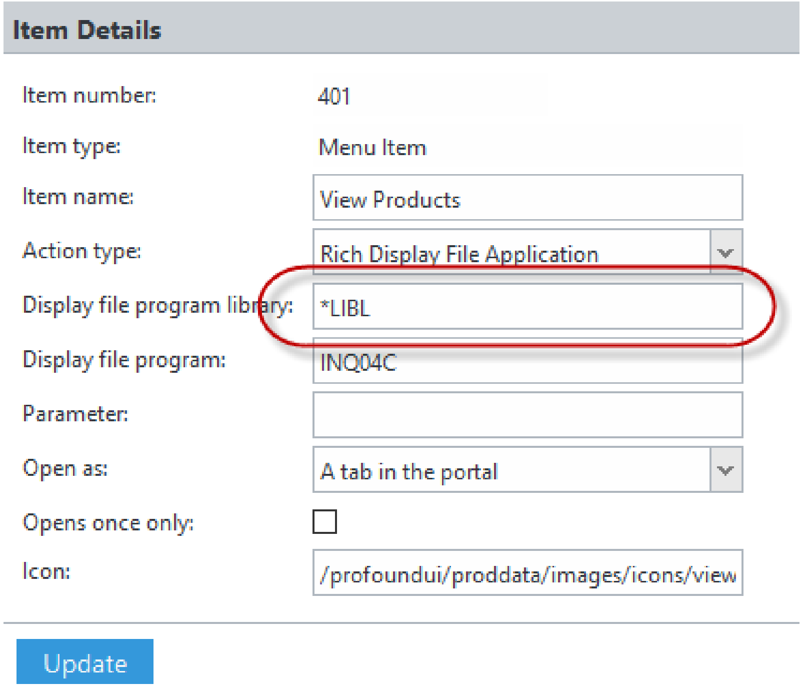
1) Get the RRN based on the Row -- http://www.profoundlogic.com/docs/pages/viewpage.action?pageId=44794197
2) Get the Row based on RRN -- http://www.profoundlogic.com/docs/pages/viewpage.action?pageId=44794198

Along with that, we enhanced the scrollToRow API so that it too will return the RRN of the scroll to row.

http://www.profoundlogic.com/docs/pages/viewpage.action?pageId=15728644

Atrium: Allow Library List When Calling Rich Displays

Prior to this release, when you define an Atrium menu item that runs a Rich Display application, the “Display file program library” had to be a library name. With this release, you can use the special value of *LIBL and Atrium will call the first program found in the library list as defined in the *Library List for Rich Display File Applications* setting for the user or group. With this enhancement, customers can move the program and Rich Display objects through the usual development lifecycle (e.g. development library, test library, and production library) without having to change the Atrium configuration for each menu item at each stage.



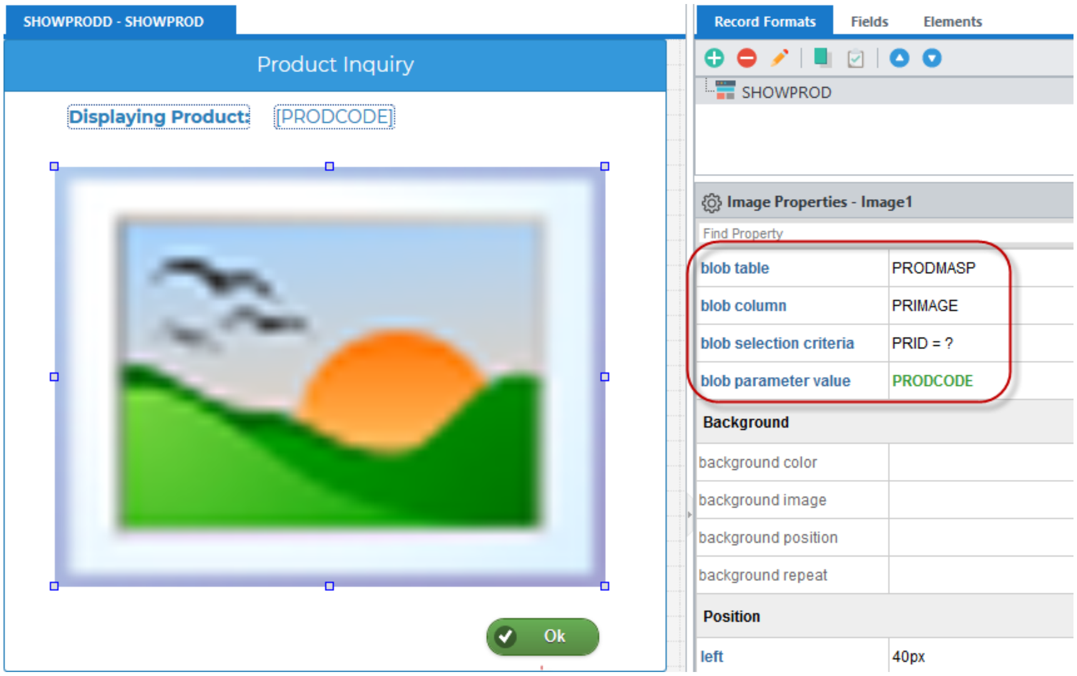
Item Details

Item number:	401
Item type:	Menu Item
Item name:	View Products
Action type:	Rich Display File Application
Display file program library:	*LIBL
Display file program:	INQ04C
Parameter:	
Open as:	A tab in the portal
Opens once only:	□
Icon:	/profoundui/proddata/images/icons/view

Image Widget Can Load from a BLOB

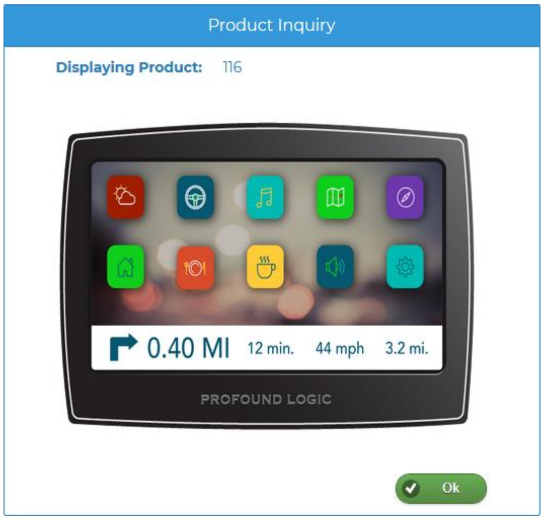
Modern databases provide the ability to store binary data in columns called Binary Large Objects (BLOB for short). Customers often use a BLOB to store images that they'd like to view in the browser, but in older releases of Profound UI you had to manually write code to retrieve the BLOB data and store it in the IFS before it could be displayed. With Profound UI 6 fix pack 3, the image widget can open and view a BLOB directly.

In this example, the image of a product is stored in a BLOB column named PRIMAGE in the PRODMASP table. I can configure the image widget properties as shown to display it:



blob table	PRODMASP
blob column	PRIMAGE
blob selection criteria	PRID = ?
blob parameter value	PRODCODE
Background	
background color	
background image	
background position	
background repeat	
Position	
left	40px

When run, the image contents are loaded from the BLOB and displayed in the image widget:

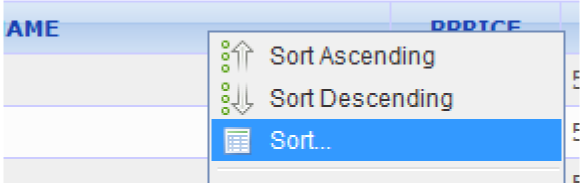

Product Inquiry

Displaying Product: 116

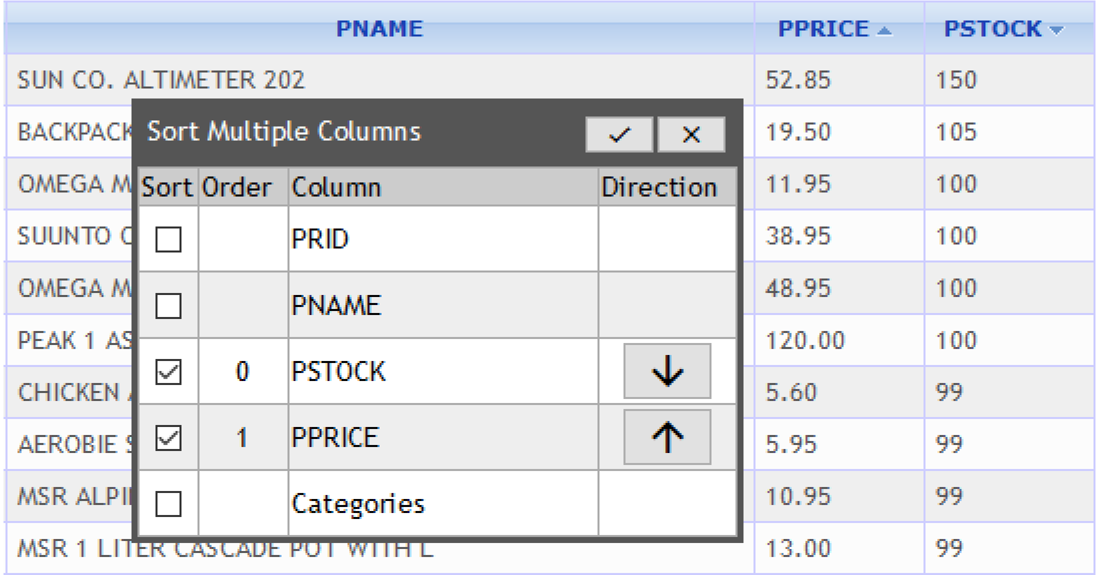
Added Grid Ability to Sort Multiple Columns

We've added a new capability to sort multiple columns simultaneously. This useful feature gives users more control over how data is presented on screen.

Users can set sort directions and set the primary sort column, secondary sort column, etc. Multi-sorting works for all grids, and many require no changes to take advantage of this feature. If a grid already has "Sort" enabled, then simply right-clicking a column heading and picking the new "Sort…" option displays a dialog for setting multi-sorting.



Once columns are chosen, clicking the checkmark sorts the data.



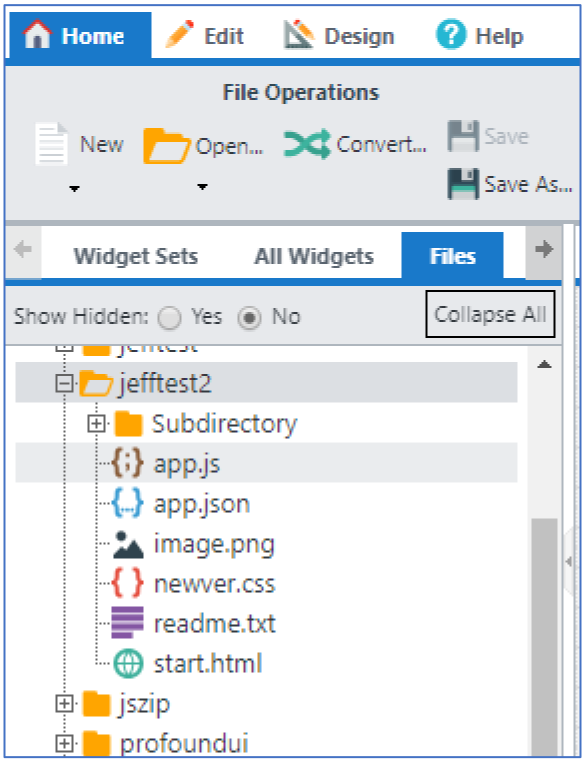
PNAME	PPRICE ▲	PSTOCK ▼
SUN CO. ALTIMETER 202	52.85	150
BACKPACK	19.50	105
OMEGA M	11.95	100
SUUNTO C	38.95	100
OMEGA M	48.95	100
PEAK 1 AS	120.00	100
CHICKEN	5.60	99
AEROBIE S	5.95	99
MSR ALPI	10.95	99
MSR 1 LITER CASCADE POT WITH L	13.00	99

Sort Multiple Columns ✓ ✕

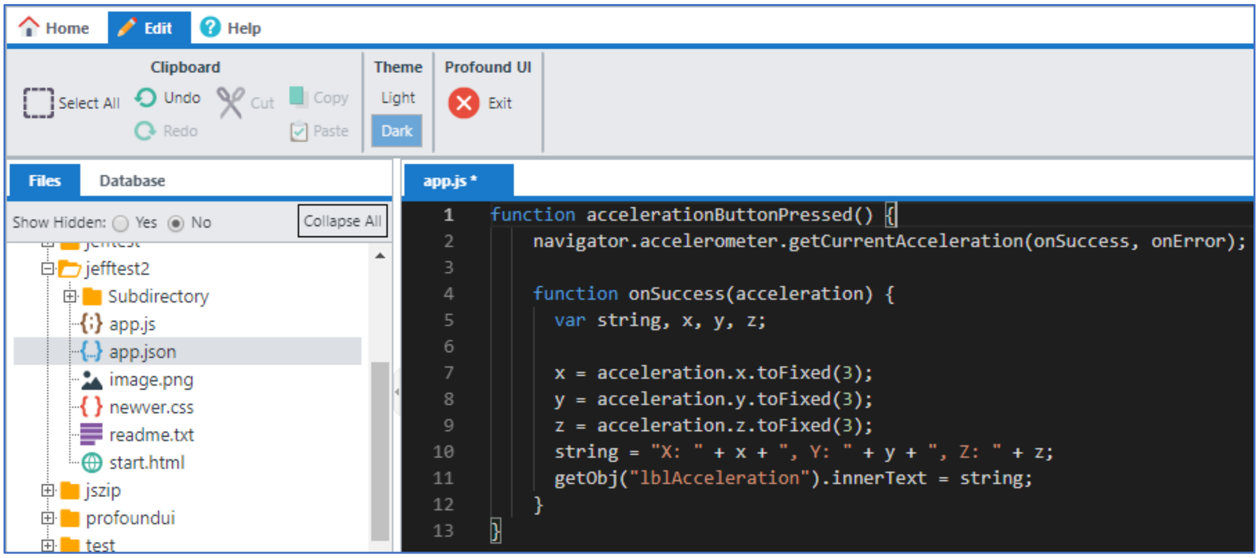
Sort	Order	Column	Direction
☐		PRID	
☐		PNAME	
☑	0	PSTOCK	↓
☑	1	PPRICE	↑
☐		Categories	

That's it! Each sorted column indicates that it is part of a sort via a triangle icon.

Miscellaneous Visual Designer Enhancements

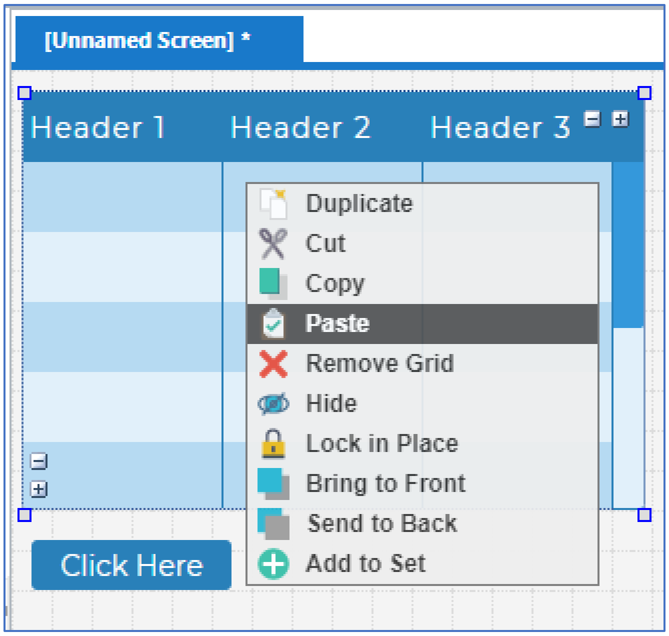


- The Files tab now shows icons for common file types (like JavaScript, JSON, CSS, etc.).
- The Save button is now only enabled when something in the editor has actually changed.



```
function accelerationButtonPressed() {
  navigator.accelerometer.getCurrentAcceleration(onSuccess, onError);

  function onSuccess(acceleration) {
    var string, x, y, z;

    x = acceleration.x.toFixed(3);
    y = acceleration.y.toFixed(3);
    z = acceleration.z.toFixed(3);
    string = "X: " + x + ", Y: " + y + ", Z: " + z;
    getObj("lblAcceleration").innerText = string;
  }
}
```

The code editor now provides dark mode, along with the previous light mode. The Edit tab now contains buttons for toggling between the light and dark themes.



Pasting items in the Visual Designer now behaves more intuitively than before. When right clicking and choosing Paste, items are now positioned at the origin of the right click. When pasting into grids or layouts, the pasted item is now placed *inside* the grid/layout, as opposed to on top of it, if the grid or layout is selected when the paste occurs.

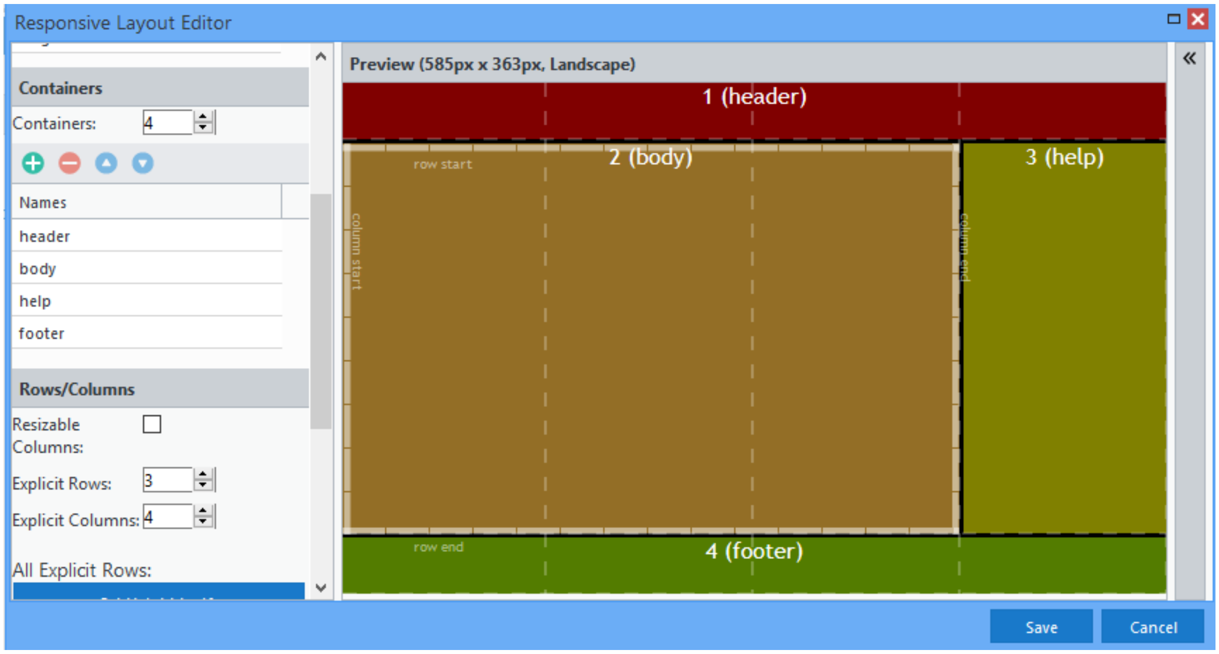
Mobile Photo Capture Enhancements

For mobile applications, the pui.capturePhoto API has been expanded to accept any configuration option that the underlying camera plugin allows. For example, the forward-facing camera can now be used, whereas before this change the options were more restricted.

More details can be found here: www.profoundlogic.com/docs/pages/viewpage.action?pageId=8519752

Responsive Layout Editor Improvements

The responsive layout editor can now add descriptive names to containers to assist in design.



Containers: 4

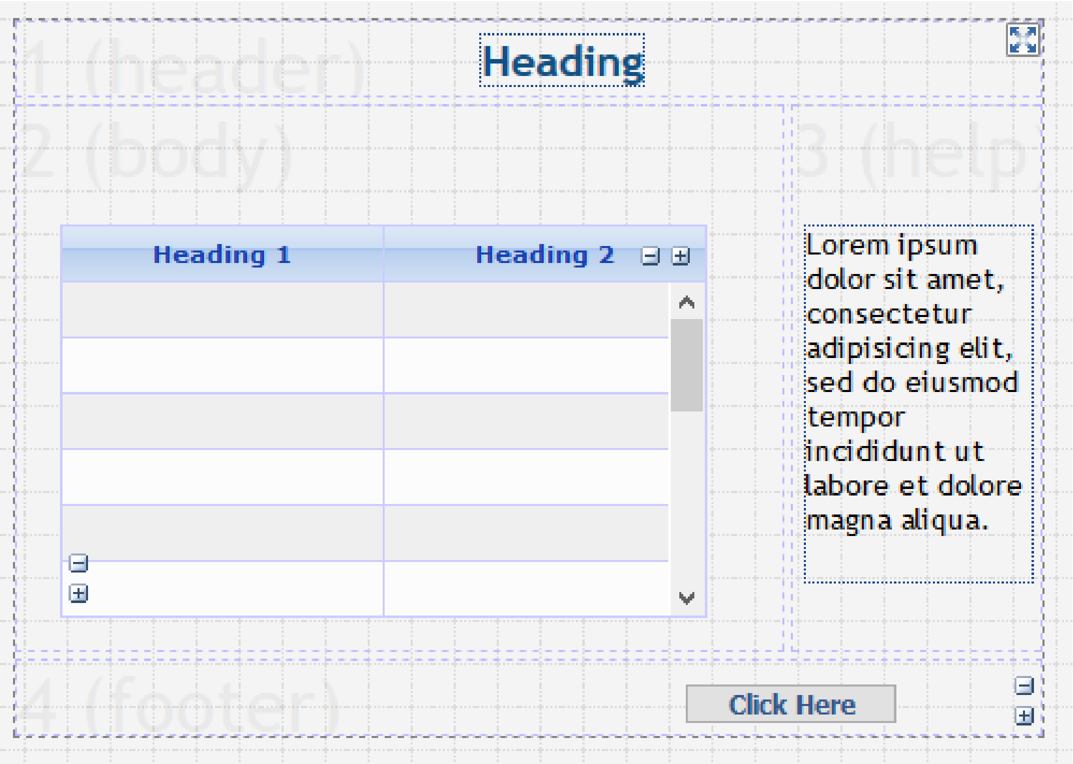
Names
header
body
help
footer

Resizable Columns: □

Explicit Rows: 3

Explicit Columns: 4

These descriptors help users keep track of the different layout containers and their expected purpose. Names can be set within the Responsive Layout Editor for ease, and the names appear on the design canvas when set:

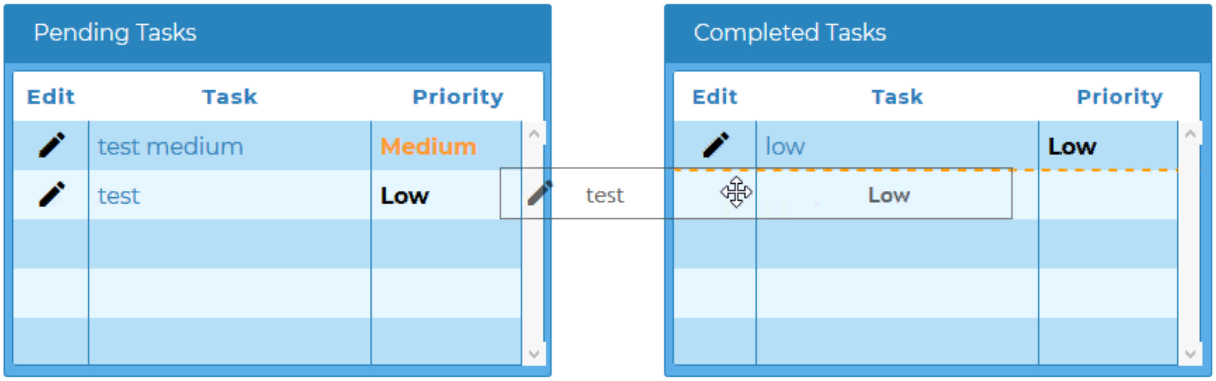


Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

(Note: The container names will not appear in the running program.)

Easier Drag and Drop for Side-by-Side Grids

We’ve made it easier to drag items from one grid into another nearby grid. Before, records could not be dropped until the dragged proxy image was dragged entirely outside of the original grid. That made it difficult to drag with wide grids, which had wide proxy images. Now, wherever the mouse is, that dragged record can be dropped, even when the proxy image is over both grids.



Pending Tasks

Edit	Task	Priority
✎	test medium	Medium
✎	test	Low

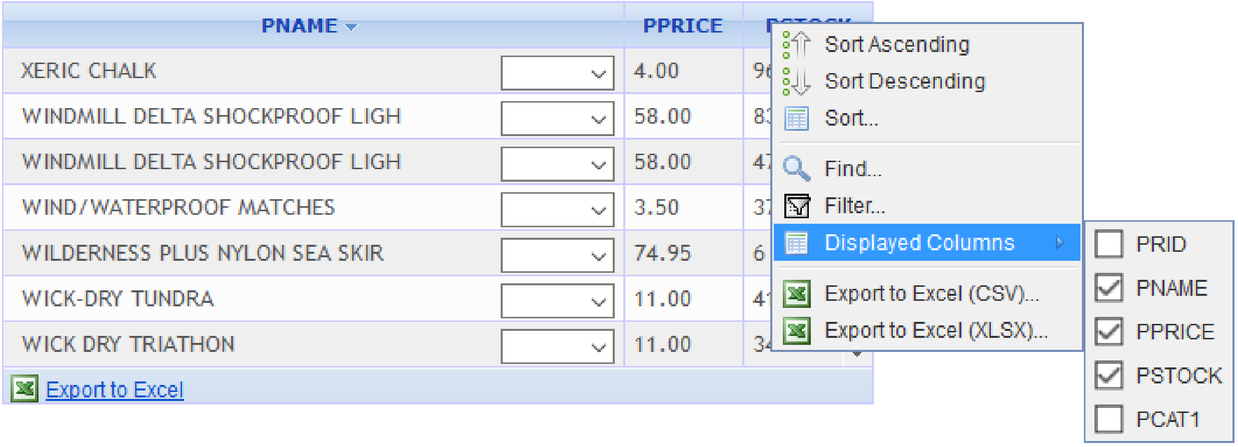
Completed Tasks

Edit	Task	Priority
✎	low	Low

New Grid API, isColumnHidden

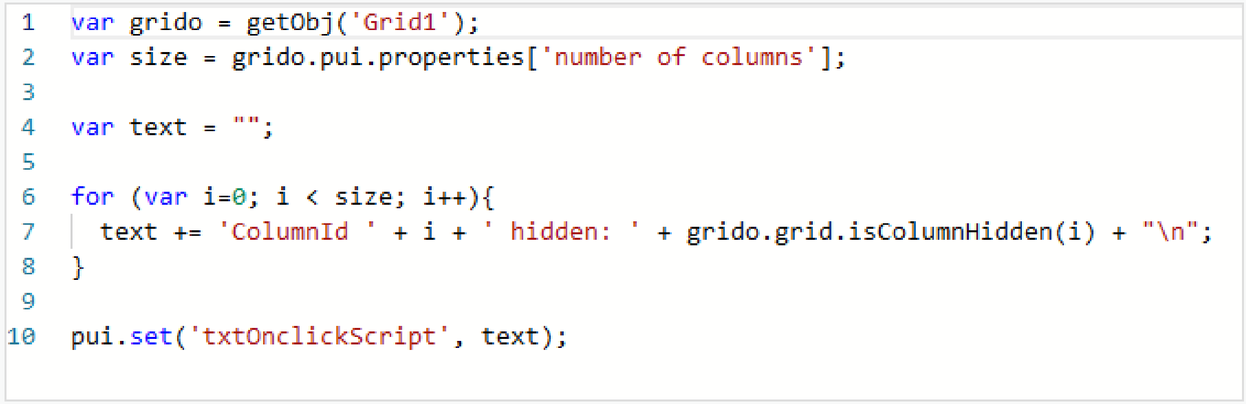
With a recent improvement, users can hide columns in grids. Programmers may want certain actions to work only on visible columns, and the isColumnHidden API allows distinguishing between visible and hidden columns.

The context menu showing hidden/shown columns is shown below.

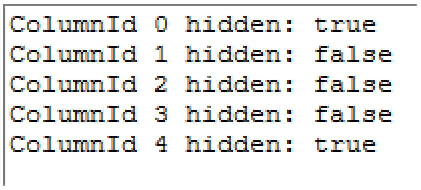


PNAME		PPRICE	PSTOCK
XERIC CHALK		4.00	96
WINDMILL DELTA SHOCKPROOF LIGH		58.00	83
WINDMILL DELTA SHOCKPROOF LIGH		58.00	47
WIND/WATERPROOF MATCHES		3.50	37
WILDERNESS PLUS NYLON SEA SKIR		74.95	6
WICK-DRY TUNDRA		11.00	41
WICK DRY TRIATHON		11.00	34

An example code segment below demonstrates displaying which columns are hidden:



```
var grido = getObj('Grid1');
var size = grido.pui.properties['number of columns'];

var text = "";

for (var i=0; i < size; i++){
  text += 'ColumnId ' + i + ' hidden: ' + grido.grid.isColumnHidden(i) + "\n";
}

pui.set('txtOnclickScript', text);
```

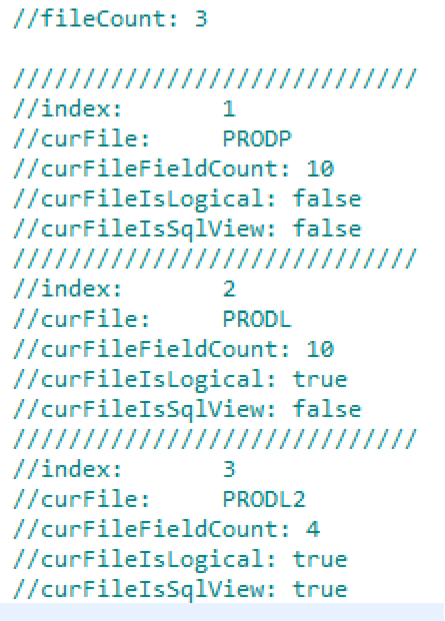


```
ColumnId 0 hidden: true
ColumnId 1 hidden: false
ColumnId 2 hidden: false
ColumnId 3 hidden: false
ColumnId 4 hidden: true
```

New JumpStart Template Variables

New template variables have been added to JumpStart, giving programmers more flexibility when generating templates. With the new file-type variables, the template can know the type of file selected in JumpStart: whether it is a logical file, an SQL view, or neither. Depending on the file type, templates can be set to generate application code one way for one file type and another way for a different file type.

Below is sample output from a template using the new variables:



```
//fileCount: 3

//////////////////////////////
//index:       1
//curFile:     PRODP
//curFileFieldCount: 10
//curFileIsLogical: false
//curFileIsSqlView: false
//////////////////////////////
//index:       2
//curFile:     PRODL
//curFileFieldCount: 10
//curFileIsLogical: true
//curFileIsSqlView: false
//////////////////////////////
//index:       3
//curFile:     PRODL2
//curFileFieldCount: 4
//curFileIsLogical: true
//curFileIsSqlView: true
```

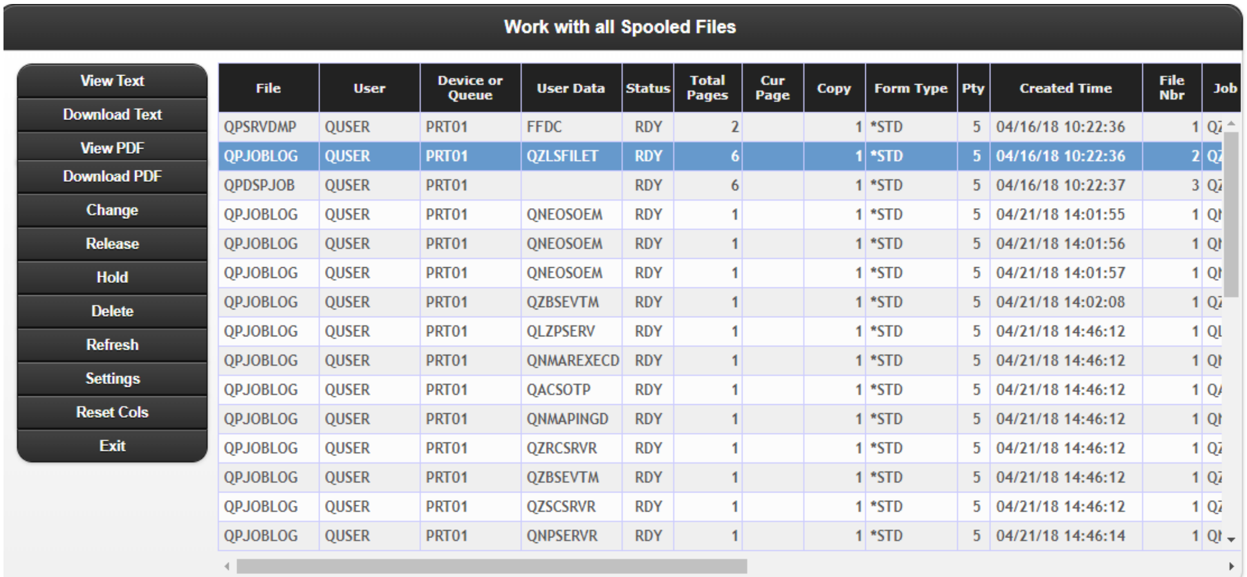
Programmers are also given the number of fields in each file, the description of the file object, and the long SQL name for a file if it has one:

```
//////////////////////////////
//index:       1
//curFile:     LONGT00001
//curFullFile: LONGT00001
//curLongFile: LONGTABLENAME
//curFileFieldCount: 3
//curFileIsLogical: false
//curFileIsSqlView: false
//curFileDescription: This is a table with a long name
```

Updates to Profound UI Work with Spooled Files

There were some small changes made to the Profound UI Work with Spooled Files (PUIWRKSPLF) utility. These are:

- Added separate buttons for downloading vs, viewing PDF and text spooled files
- The download buttons replace the old “View Inline” setting, which has been removed.
- The “Create Date” and “Create Time” columns have been combined into a single column to improve sorting
- The generated file name used when downloading a file has more information in it.

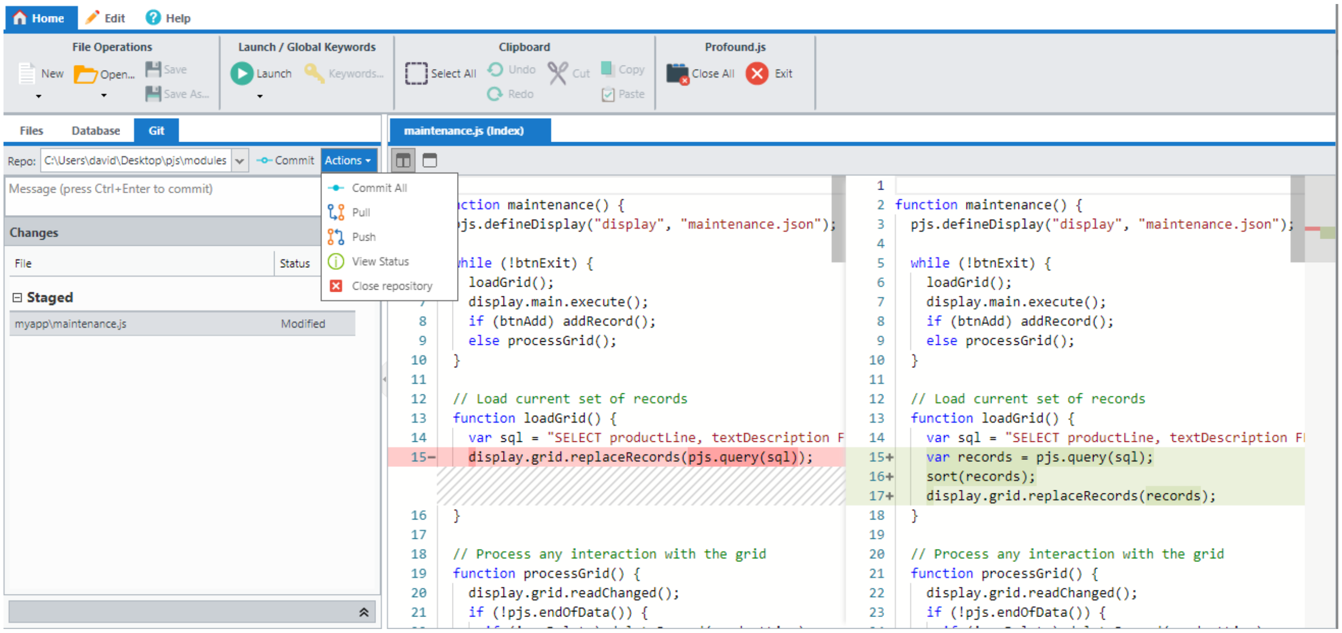


File	User	Device or Queue	User Data	Status	Total Pages	Cur Page	Copy	Form Type	Pty	Created Time	File Nbr	Job
QPSRVDMP	QUSER	PRT01	FFDC	RDY	2		1	*STD	5	04/16/18 10:22:36	1	QZ
QPJOBLOG	QUSER	PRT01	QZLSFILET	RDY	6		1	*STD	5	04/16/18 10:22:36	2	QZ
QPDSPJOB	QUSER	PRT01		RDY	6		1	*STD	5	04/16/18 10:22:37	3	QZ
QPJOBLOG	QUSER	PRT01	QNEOSOEM	RDY	1		1	*STD	5	04/21/18 14:01:55	1	QN
QPJOBLOG	QUSER	PRT01	QNEOSOEM	RDY	1		1	*STD	5	04/21/18 14:01:56	1	QN
QPJOBLOG	QUSER	PRT01	QNEOSOEM	RDY	1		1	*STD	5	04/21/18 14:01:57	1	QN
QPJOBLOG	QUSER	PRT01	QZBSEVTM	RDY	1		1	*STD	5	04/21/18 14:02:08	1	QZ
QPJOBLOG	QUSER	PRT01	QLZPSERV	RDY	1		1	*STD	5	04/21/18 14:46:12	1	QL
QPJOBLOG	QUSER	PRT01	QNMAREXECD	RDY	1		1	*STD	5	04/21/18 14:46:12	1	QN
QPJOBLOG	QUSER	PRT01	QACSOTP	RDY	1		1	*STD	5	04/21/18 14:46:12	1	QA
QPJOBLOG	QUSER	PRT01	QNMAPINGD	RDY	1		1	*STD	5	04/21/18 14:46:12	1	QN
QPJOBLOG	QUSER	PRT01	QZRCSRVR	RDY	1		1	*STD	5	04/21/18 14:46:12	1	QZ
QPJOBLOG	QUSER	PRT01	QZBSEVTM	RDY	1		1	*STD	5	04/21/18 14:46:12	1	QZ
QPJOBLOG	QUSER	PRT01	QZSCSRVR	RDY	1		1	*STD	5	04/21/18 14:46:12	1	QZ
QPJOBLOG	QUSER	PRT01	QNPSERVR	RDY	1		1	*STD	5	04/21/18 14:46:14	1	QN

Profound.js 4.10.0 Enhancements

Git Integration for Profound.js IDE

The Profound.js IDE now features seamless integration with Git, the popular free/open source version control system. Users can now create Git repositories in the workspace, manage changes, make commits, integrate with GitHub, and more -- all without leaving the IDE.



Repo: C:\Users\david\Desktop\pjs\modules

Message (press Ctrl+Enter to commit)

File	Status
myapp\maintenance.js	Modified

```
ction maintenance() {
js.defineDisplay("display", "maintenance.json");

hile (!btnExit) {
  loadGrid();
  display.main.execute();
  if (btnAdd) addRecord();
  else processGrid();
}

// Load current set of records
function loadGrid() {
  var sql = "SELECT productLine, textDescription F
- display.grid.replaceRecords(pjs.query(sql));
}

// Process any interaction with the grid
function processGrid() {
  display.grid.readChanged();
  if (!pjs.endOfData()) {
```

```

function maintenance() {
  pjs.defineDisplay("display", "maintenance.json");

  while (!btnExit) {
    loadGrid();
    display.main.execute();
    if (btnAdd) addRecord();
    else processGrid();
  }

  // Load current set of records
  function loadGrid() {
    var sql = "SELECT productLine, textDescription F
+   var records = pjs.query(sql);
+   sort(records);
+   display.grid.replaceRecords(records);
  }

  // Process any interaction with the grid
  function processGrid() {
    display.grid.readChanged();
    if (!pjs.endOfData()) {
```

Git integration is available in the Profound.js IDE for Windows, Mac, and Linux. Support is planned for IBM i in the future.

For more information, see the documentation here:

http://www.profoundlogic.com/docs/display/PUI/Git+Integration

Profound.js Docker Image

Profound.js is now available as a Docker image via Docker Hub. *Docker* is a tool designed for creating, deploying, and running applications by using OS-level virtualization technology called containers. Docker Hub is a cloud-based repository in which Docker users and partners create, test, store and distribute container images.

Launching a Profound.js instance in a container is now as simple as running one Docker command. For example, to start a container running the latest version of Profound.js:

```
docker run -d --name profoundjs -p 8081:8081 profoundlogic/profoundjs
```

The Docker Hub repository will be updated continually as new Profound.js releases become available. For more information, visit the Profound.js repository page on Docker Hub here:

https://hub.docker.com/r/profoundlogic/profoundjs