**Form.io Tips & Tricks**

**Important Note: NUcore is currently referencing an older form.io Library Version. Here’s the description from Table XI:**

Form.IO's documentation examples for embedding a form (which is what we do in NUcore) point to the Form.IO libraries' latest versions. As a result of this, any time Form.IO published a new version of that library, we were using it (because the URL we were linking to included latest). When they published the 4.x series, it had a bunch of backwards-incompatible changes, and this is what broke the form display. I have fixed Form.IO to version 3.28.0 for now, because the latest version seems to have some hiccups even in Form.IO's own interface (if you use next.form.io instead of portal.form.io you'll notice this), and I have reported it to Form.IO.

Form Setup & Integration

1. When setting up a form for the first time, the “Anonymous” user must be provided with access rights for NUcore integration to function properly
   1. Create the new form
   2. Click on the “Access” tab
   3. Reload the browser page to view the “Submission Data Permissions” section
   4. In “Submission Data Permissions”, add the “Anonymous” role to:
      1. Create Own Submissions
      2. Create All Submissions
      3. Read Own Submissions
      4. Read All Submissions
      5. Update Own Submissions
      6. Update All Submissions
   5. Leave the “Form Definition Access” section alone
      1. “Read Form Definition” will show “Administrator”, “Authenticated”, and “Anonymous”
2. The link to be entered and activated as an “Order Form” in NUcore is located in the “Embed” tab; use the link listed just below the “Form Embedded” text

NUcore API Links

1. NUcore can port the following information to a form instance:
   1. User Name
   2. User Email
   3. User ID
   4. Owner Name
   5. Owner Email
   6. Owner ID
   7. Payment Source
   8. Department Name
   9. Order Date
   10. NUcore Order Number
2. To accomplish this, the key located in the “API” tab for each respective form component (listed parenthetically) must be keyed as follows:
   1. User Name – orderedForName (Basic Components: Text Field)
   2. User Email – orderedForEmail (Basic Components: Text Field)
   3. User ID – orderedForUsernname (Basic Components: Text Field)
   4. Owner Name – accountOwnerName (Basic Components: Text Field)
   5. Owner Email – accountOwnerEmail (Basic Components: Text Field)
   6. Owner ID – accountOwnerUsername (Basic Components: Text Field)
   7. Payment Source – paymentSourceAccountNumber (Basic Components: Text Field)
   8. Department Name – departmentName (Basic Components: Text Field)
   9. Order Date – orderedAtDate (Advanced: Date/Time)
   10. NUcore Order Number – nucoreOrderNumber (Basic Components: Text Field)

Advanced Topics

1. File Upload
   1. When testing, forms setup in the form.io Staging environment must be pointed at the NUcore Staging Server; likewise, forms setup in the form.io Live environment must be pointed at the NUcore Production Server (different directories are setup in S3 for the different environments)
   2. When setting up the “File Component”, select S3 as the “Storage” option
   3. Set a reasonable “File Maximum Size” value (e.g. 10MB)
   4. Selecting the “Multiple Values” box enables users to upload multiple files to the form at once
2. Actions tab – setup multiple “Actions” upon form submission
   1. Possible to generate emails upon submission by adding an “Email Action”
      1. Can use form data in the Subject Line, Email Destination, and other fields by including the element with this syntax: {{ data.form\_component\_api }}
   2. Possible to port data from one form to a separate “Resource” form
      1. Setup a separate “Resource” form in the “Resources” left-hand tab
      2. Create destination fields for the ported data
         1. Component in the destination should match the components in the source form
         2. If porting data from an “Edit/Data Grid”, the API keys of parent and child components must match in both the source and the destination
      3. Once the destination “Resource” is setup, a new “Action” must be setup in the source form, this “Action” must be the first/top “Action” in the source form for data to remain accessible upon form retrieval in NUcore
      4. In the existing Save Submission Action, change Save Submission from “This Form” to the new receiver form that you have setup
      5. Map the “Resource Fields” as appropriate and be sure to “Save Action”
      6. Create a second Save Submission Action, keeping the defaults for this second “Action” (Save Submission to “This Form”) and be sure to “Save Action”
3. Calculated Value Syntax
   1. Possible to populate a tertiary component with the output from combining other component values
   2. For the calculated component, in the “Data” tab, under the “Calculated Value” section, enter custom code to accomplish this
      1. Example syntax to add two cells: value = (data.component\_1\_api\_key || 0) + (data.component\_2\_api\_key || 0);
      2. This syntax manages cases of “Component 1” or “Component 2” = 0
4. Advanced Conditional Syntax
   1. Possible to use the “Advanced” section in the “Conditional” tab to show a component conditionally based on Boolean logic (javascript syntax)
   2. To show a component based on either selection of a Radio Box, use the following Advanced Conditional syntax:
      1. show = data.radio\_box\_api == 'radio\_box\_value\_1' || data.radio\_box\_api == ‘radio\_box\_value\_2’;
      2. This syntax will work for multiple component types
   3. To show a component based on either selection of a Select Box, use the following Advanced Conditional syntax:
      1. show = data.select\_box\_api.select\_box\_value\_1 || data.select\_box\_api.select\_box\_value\_2;
      2. This syntax is unique to basing a conditional on a Select Box component
5. Form View Pro – Provide Core Managers Access to Forms without Access to form.io Project Management
   1. In the given form.io Stage Environment, go to the “User” Resource
   2. Click on the “Use” tab
   3. Enter the desired Email and Password for the core manager and then “Submit” to add this as a “log in” data point
   4. Decide which form will be accessed by a given core manager
      1. If the form is a separate “Resource” populated by data ported over from a primary source form, then the customer-submitted data can be preserved independent of any edits or deletions to the new form
      2. In “Submission Data Permissions”, add the “Authenticated” role to:
         1. Create Own Submissions
         2. Create All Submissions
         3. Read Own Submissions
         4. Read All Submissions
         5. Update Own Submissions
         6. Update All Submissions
   5. Click the “Try Our Next Portal” button (upper-right corner)
   6. Click the left-hand “Forms” tab
   7. Click on the form whose access has been modified
   8. Click on the “Launch” tab
   9. Copy the URL available on this tab
   10. The core manager can log in using this URL to access this form only, they can review all submissions, edit submission, delete submissions; they will not have access to the primary form.io Stages
6. PDF Generation Workflow
   1. For all PDF-Generating Forms, the correct “PDF Token” is required; find it as follows:
      1. Log in to form.io
      2. Select “Project”
      3. Go to the appropriate “Stage”
      4. Go to “Settings” (left-hand navigation panel)
      5. Go to “PDF Management” (new left-hand navigation panel)
      6. Go to the “Enterprise” tab (top navigation selector)
      7. See the “PDF Token” field
   2. For all related Forms, “Access” must be set to “Anonymous” as noted in “Form Setup and Integration” section 1(d)
   3. Generate PDF of Current Form View
      1. Replaces the “Submit” button with a button that submits the form and generates a PDF download of the current form view (including the data)
      2. In the Submit “Button Component” make these changes
         1. Change the “Label” to indicate the button’s dual functions
         2. Change the “Action” to “Custom”
         3. Add “Button Custom Logic” from the text file “Form.io Examples” under the section “Generate PDF of Current Form View”
            1. Must set “var fileToken” with the appropriate PDF Token
            2. Must set “var currentForm” with the “embed” link of the current form
            3. The “var pdfFileName” sets the default file name for the downloaded PDF (can be altered if desired)

Currently set to concatenate fields from our API ported fields (User ID and NUcore Order ID)

* + - * 1. Update the three “instance.label” values, if desired

Default is “Generating Preview”, “Downloading PDF”, and “Saving”

* 1. Port Form Data to a Custom PDF
     1. Allows a user to download a custom PDF with data from a “Parent Form”
     2. Create a new, separate “PDF Form” (set the “Access” permissions)
        1. Upload a blank PDF to be used as the template for the PDF
        2. Add “Text Fields” to the “PDF Form” as needed, positioning appropriately
           1. The API value of each “Text Field” will need to match the API value of the data that will be ported from the “Parent Form” (Parent Form is the form where the PDF will be nested)
           2. It is also possible to use “Text Area” components, but note that if a “Text Area” is used, it must be a “Text Area” in both the “PDF Form” and in the component in the “Parent Form” mapping to the “Text Area” in the “PDF Form”
     3. In the “Parent Form” where the PDF will be generated, setup as follows:
        1. Add a “Button” Component to download the PDF
           1. Change the “Label” to indicate the Download function
           2. Change the “Action” to “Custom”
           3. Add “Button Custom Logic” from the text file “Form.io Examples” under the section “Generate PDF of Current Form View”

Must set “var fileToken” with the PDF Token

Must set “var pdf” with the “embed” link of the “Child” PDF Form (not the embed link of the “Parent Form”)

The “var pdfFileName” sets the default file name for the downloaded PDF (can be altered if desired)

Currently set to concatenate fields from our API ported fields (User ID and NUcore Order ID)

Update the “instance.label” value, if desired

Default is “Print Tag”