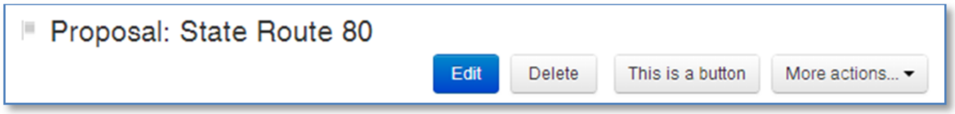
Adding Custom Buttons

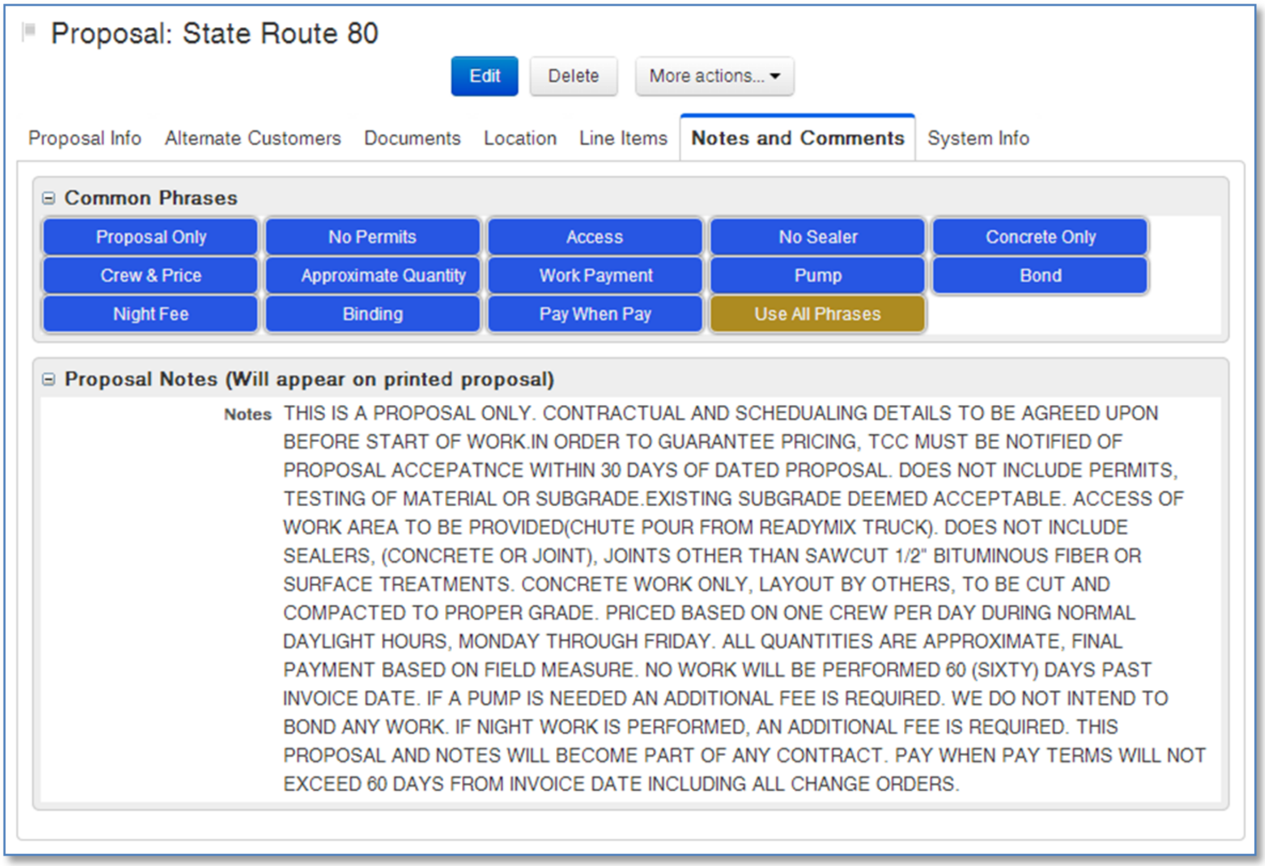
Using script components and CSS to call triggers and enhance the UI

One of the design requirements of this project was to support phrase snippets from the legacy software. As a project manager prepares the proposal, it needs to have one or many standard phrases entered into the proposal notes fields, depending on the project requirements.

Rollbase already has a mechanism in place to create buttons, but it has a few limitations. First, you don't have control over the look of the button. Additionally, you can't make the button tab specific.

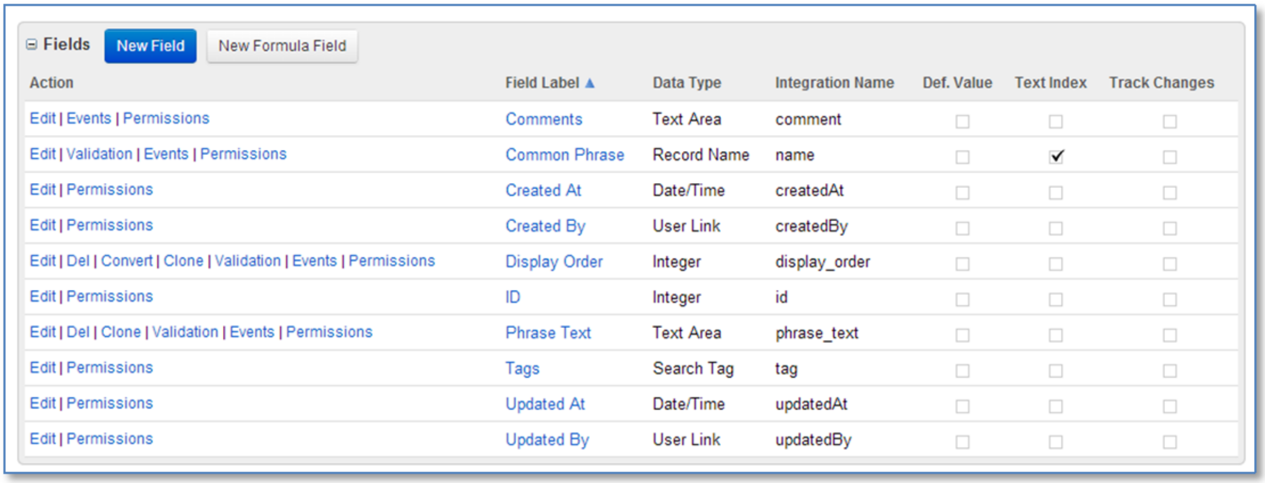


As seen above, I can add a button like the sample above. Unfortunately, for my proposal screen, I have numerous tabs, but I only want the buttons available on the Notes and Comments tab. Below is a preview of the finished solution. The user is presented with a section containing buttons representing the common phrases. The buttons are much more styled and enhance the user experience.



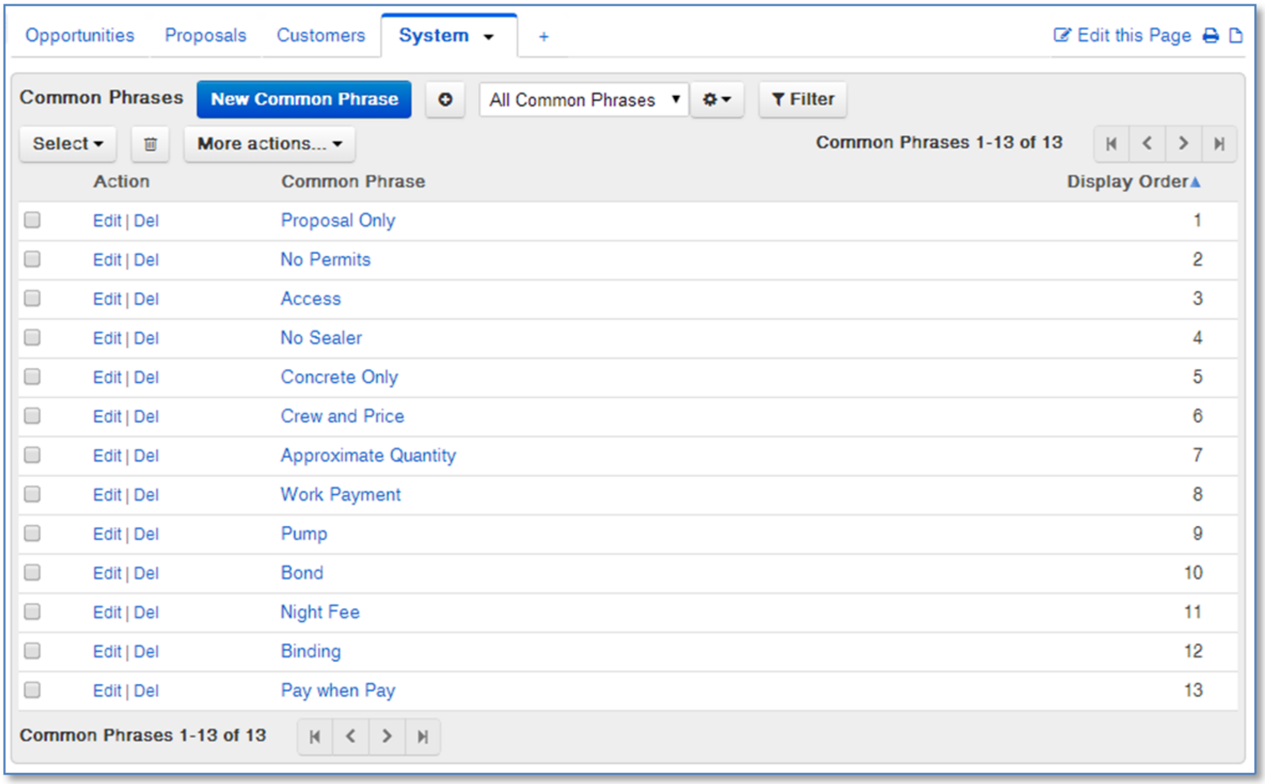
Notes: THIS IS A PROPOSAL ONLY. CONTRACTUAL AND SCHEDUALING DETAILS TO BE AGREED UPON BEFORE START OF WORK.IN ORDER TO GUARANTEE PRICING, TCC MUST BE NOTIFIED OF PROPOSAL ACCEPATNCE WITHIN 30 DAYS OF DATED PROPOSAL. DOES NOT INCLUDE PERMITS, TESTING OF MATERIAL OR SUBGRADE.EXISTING SUBGRADE DEEMED ACCEPTABLE. ACCESS OF WORK AREA TO BE PROVIDED(CHUTE POUR FROM READYMIX TRUCK). DOES NOT INCLUDE SEALERS, (CONCRETE OR JOINT), JOINTS OTHER THAN SAWCUT 1/2" BITUMINOUS FIBER OR SURFACE TREATMENTS. CONCRETE WORK ONLY, LAYOUT BY OTHERS, TO BE CUT AND COMPACTED TO PROPER GRADE. PRICED BASED ON ONE CREW PER DAY DURING NORMAL DAYLIGHT HOURS, MONDAY THROUGH FRIDAY. ALL QUANTITIES ARE APPROXIMATE, FINAL PAYMENT BASED ON FIELD MEASURE. NO WORK WILL BE PERFORMED 60 (SIXTY) DAYS PAST INVOICE DATE. IF A PUMP IS NEEDED AN ADDITIONAL FEE IS REQUIRED. WE DO NOT INTEND TO BOND ANY WORK. IF NIGHT WORK IS PERFORMED, AN ADDITIONAL FEE IS REQUIRED. THIS PROPOSAL AND NOTES WILL BECOME PART OF ANY CONTRACT. PAY WHEN PAY TERMS WILL NOT EXCEED 60 DAYS FROM INVOICE DATE INCLUDING ALL CHANGE ORDERS.

There are a couple of prerequisites for this challenge. First off the users need to be able to expand the list of phrases if desired so this will require its own object. We will create an object with the integration name of ccCommon_phrases and hang it off of the system tab like some of the other lesser used objects. It will have the following structure:



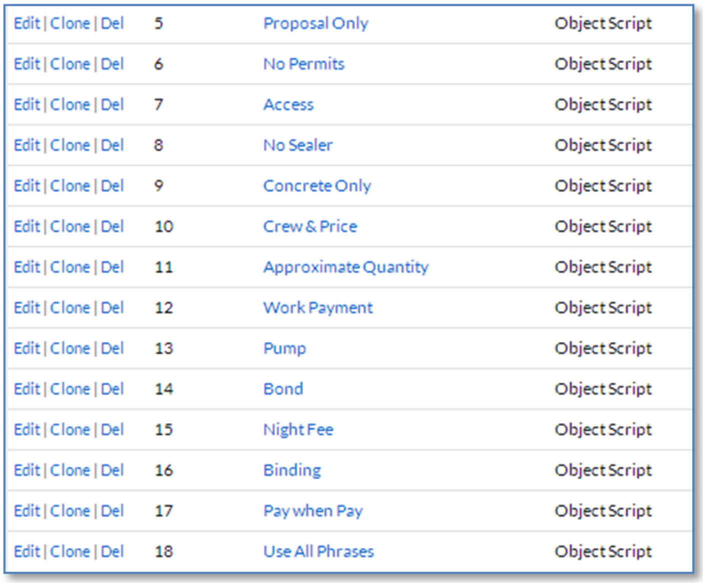
Action	Field Label ▲	Data Type	Integration Name	Def. Value	Text Index	Track Changes
Edit | Events | Permissions	Comments	Text Area	comment	☐	☐	☐
Edit | Validation | Events | Permissions	Common Phrase	Record Name	name	☐	☑	☐
Edit | Permissions	Created At	Date/Time	createdAt	☐	☐	☐
Edit | Permissions	Created By	User Link	createdBy	☐	☐	☐
Edit | Del | Convert | Clone | Validation | Events | Permissions	Display Order	Integer	display_order	☐	☐	☐
Edit | Permissions	ID	Integer	id	☐	☐	☐
Edit | Del | Clone | Validation | Events | Permissions	Phrase Text	Text Area	phrase_text	☐	☐	☐
Edit | Permissions	Tags	Search Tag	tag	☐	☐	☐
Edit | Permissions	Updated At	Date/Time	updatedAt	☐	☐	☐
Edit | Permissions	Updated By	User Link	updatedBy	☐	☐	☐

Once created, we can populate the object with the desired phrases.



	Action	Common Phrase	Display Order ▲
☐	Edit | Del	Proposal Only	1
☐	Edit | Del	No Permits	2
☐	Edit | Del	Access	3
☐	Edit | Del	No Sealer	4
☐	Edit | Del	Concrete Only	5
☐	Edit | Del	Crew and Price	6
☐	Edit | Del	Approximate Quantity	7
☐	Edit | Del	Work Payment	8
☐	Edit | Del	Pump	9
☐	Edit | Del	Bond	10
☐	Edit | Del	Night Fee	11
☐	Edit | Del	Binding	12
☐	Edit | Del	Pay when Pay	13

The next thing we’re going to need are some object script triggers that are capable of looking up the phrase snippet based on the phrase name needed. We will create a trigger to represent each code snippet, as well as one to apply all snippets back to back.



Edit | Clone | Del	5	Proposal Only	Object Script
Edit | Clone | Del	6	No Permits	Object Script
Edit | Clone | Del	7	Access	Object Script
Edit | Clone | Del	8	No Sealer	Object Script
Edit | Clone | Del	9	Concrete Only	Object Script
Edit | Clone | Del	10	Crew & Price	Object Script
Edit | Clone | Del	11	Approximate Quantity	Object Script
Edit | Clone | Del	12	Work Payment	Object Script
Edit | Clone | Del	13	Pump	Object Script
Edit | Clone | Del	14	Bond	Object Script
Edit | Clone | Del	15	Night Fee	Object Script
Edit | Clone | Del	16	Binding	Object Script
Edit | Clone | Del	17	Pay when Pay	Object Script
Edit | Clone | Del	18	Use All Phrases	Object Script

The actual code is pretty simple, and is the same for each one, with the exception of the key value sent to the query.

```
//First line queries the phrases where the record name = “Proposal Only”
var phraseText = rbv_api.selectValue("SELECT phrase_text FROM ccCommon_phrase where name =
?","Proposal Only");
//Second line queries the current record to get any existing contents from the notes field
var originalNotes = rbv_api.selectValue("SELECT proposal_notes FROM ccProposal where id =
{!id}");
//Third line appends the phrase text to any existing notes.
originalNotes = originalNotes + " " + phraseText;
//Had the existing notes been empty, this line clears the returned null. It’s a creative way
//to minimize code required to concatenate the strings.
originalNotes = originalNotes.replace("null","");
//Last line updates the record with the new notes value.
rbv_api.setFieldValue("ccProposal", {!id}, "proposal_notes", originalNotes);
```

An observation about the query to insert all phrases; when this choice is selected, it makes an assumption that any existing notes will be removed. Additionally, I attempted to simply call the other triggers, but ended up with timing issues, so the result was to simply concatenate all strings within the one single trigger. Solution works well.

Building the View Screen

From the Notes and Comments tab on the Proposal View Screen, we will “Edit This Page” as we always do. Drag a new section onto the screen, above the notes section, and call it “Common Phrases” with the typical Title & Rounded Border style. With that in place, drag and drop a New Script Component onto the next section. Not only does this section support Javascript, but it supports full HTML, so we’ll take advantage of HTML, Javascript and CSS.

```
<html>
<head>
<style>
/***FIRST STYLE THE BUTTON***/
input#gobutton{
width: 13em;
cursor:pointer; /*forces the cursor to change to a hand when the button is hovered*/
padding:5px 25px; /*add some padding to the inside of the button*/
background:#2756E3; /*the color of the button*/
border:1px solid #E8E8E8; /*required or the default border for the browser will appear*/
/*give the button curved corners, alter the size as required*/
-moz-border-radius: 5px;
-webkit-border-radius: 5px;
border-radius: 5px;
/*give the button a drop shadow*/
-webkit-box-shadow: 0 0 4px rgba(0,0,0, .75);
-moz-box-shadow: 0 0 4px rgba(0,0,0, .75);
box-shadow: 0 0 4px rgba(0,0,0, .75);
/*style the text*/
color:#f3f3f3;
font-size:0.9em;
}
/***NOW STYLE THE BUTTON'S HOVER AND FOCUS STATES***/
input#gobutton:hover, input#gobutton:focus{
background-color :#1A3CA3; /*make the background a little darker*/
/*reduce the drop shadow size to give a pushed button effect*/
-webkit-box-shadow: 0 0 1px rgba(0,0,0, .75);
-moz-box-shadow: 0 0 1px rgba(0,0,0, .75);
box-shadow: 0 0 1px rgba(0,0,0, .75);
}
/***We want the ALL BUTTON to have a different look. Just copy/paste and change color***/

input#allbutton{
width: 13em;
cursor:pointer; /*forces the cursor to change to a hand when the button is hovered*/
padding:5px 25px; /*add some padding to the inside of the button*/
background:#AD8B21; /*the colour of the button*/
border:1px solid #E8E8E8; /*required or the default border for the browser will appear*/
/*give the button curved corners, alter the size as required*/
-moz-border-radius: 5px;
-webkit-border-radius: 5px;
border-radius: 5px;
/*give the button a drop shadow*/
-webkit-box-shadow: 0 0 4px rgba(0,0,0, .75);
-moz-box-shadow: 0 0 4px rgba(0,0,0, .75);
box-shadow: 0 0 4px rgba(0,0,0, .75);
/*style the text*/
color:#f3f3f3;
font-size:0.9em;
}
/***NOW STYLE THE BUTTON'S HOVER AND FOCUS STATES***/
input#allbutton:hover, input#gobutton:focus{
background-color :#856A1B; /*make the background a little darker*/
/*reduce the drop shadow size to give a pushed button effect*/
-webkit-box-shadow: 0 0 1px rgba(0,0,0, .75);
-moz-box-shadow: 0 0 1px rgba(0,0,0, .75);
box-shadow: 0 0 1px rgba(0,0,0, .75);
}
</style>
```

```
/***Empty Callback function. We don’t use, but it needs to be stubbed out.***/

<script>
function myCallback(){
}
/***The following are all of the scripts to call the individual triggers by integration name.***/
//each one reloads the page to refresh it and selects the current tab element. This technique was
//used to ensure proper firing of each button in succession.
function proposalOnly(){
rbf_runTrigger("ccProposal", 96387626, "btnProposalOnly", false, myCallback);
location.reload();
document.getElementById("rbe_viewTabSpan6").focus();
}
function noPermits(){
rbf_runTrigger("ccProposal", 96387626, "btnNoPermits", false, myCallback);
location.reload();
document.getElementById("rbe_viewTabSpan6").focus();
}
function access(){
rbf_runTrigger("ccProposal", 96387626, "btnAccess", false, myCallback);
location.reload();
document.getElementById("rbe_viewTabSpan6").focus();
}
function noSealer(){
rbf_runTrigger("ccProposal", 96387626, "btnNoSealer", false, myCallback);
location.reload();
document.getElementById("rbe_viewTabSpan6").focus();
}
function concreteOnly(){
rbf_runTrigger("ccProposal", 96387626, "btnConcreteOnly", false, myCallback);
location.reload();
}
function crewAndPrice(){
rbf_runTrigger("ccProposal", 96387626, "btnCrewAndPrice", false, myCallback);
location.reload();
document.getElementById("rbe_viewTabSpan6").focus();
}
function approxQuant(){
rbf_runTrigger("ccProposal", 96387626, "btnApproxQuant", false, myCallback);
location.reload();
}
function workPayment(){
rbf_runTrigger("ccProposal", 96387626, "btnWorkPayment", false, myCallback);
location.reload();
document.getElementById("rbe_viewTabSpan6").focus();
}
function pump(){
rbf_runTrigger("ccProposal", 96387626, "btnPump", false, myCallback);
location.reload();
document.getElementById("rbe_viewTabSpan6").focus();
}
function bond(){
rbf_runTrigger("ccProposal", 96387626, "btnBond", false, myCallback);
location.reload();
document.getElementById("rbe_viewTabSpan6").focus();
}
function nightFee(){
rbf_runTrigger("ccProposal", 96387626, "btnNightFee", false, myCallback);
location.reload();
document.getElementById("rbe_viewTabSpan6").focus();
}
function binding(){
```

```
rbf_runTrigger("ccProposal", 96387626, "btnBinding", false, myCallback);
location.reload();
document.getElementById("rbe_viewTabSpan6").focus();
}
function payWhenPay(){
rbf_runTrigger("ccProposal", 96387626, "btnPayWhenPay", false, myCallback);
location.reload();
document.getElementById("rbe_viewTabSpan6").focus();
}

function useAllPhrases(){

rbf_runTrigger("ccProposal", 96387626, "btnUseAllPhrases", false, myCallback);
location.reload();
document.getElementById("rbe_viewTabSpan6").focus();
}
</script>
</head>
/***LASTLY, PUT THE BUTTONS ON THE SCREEN AND BIND THEM TO THE PROPER FUNCTION***/
<body>
<input id="gobutton" type="submit" onclick="proposalOnly()" value="Proposal Only" />
<input id="gobutton" type="submit" onclick="noPermits()" value="No Permits" />
<input id="gobutton" type="submit" onclick="access()" value="Access" />
<input id="gobutton" type="submit" onclick="noSealer()" value="No Sealer" />
<input id="gobutton" type="submit" onclick="concreteOnly()" value="Concrete Only" />
<input id="gobutton" type="submit" onclick="crewAndPrice()" value="Crew & Price" />
<input id="gobutton" type="submit" onclick="approxQuant()" value="Approximate Quantity" />
<input id="gobutton" type="submit" onclick="workPayment()" value="Work Payment" />
<input id="gobutton" type="submit" onclick="pump()" value="Pump" />
<input id="gobutton" type="submit" onclick="bond()" value="Bond" />
<input id="gobutton" type="submit" onclick="nightFee()" value="Night Fee" />
<input id="gobutton" type="submit" onclick="binding()" value="Binding" />
<input id="gobutton" type="submit" onclick="payWhenPay()" value="Pay When Pay" />
<input id="allbutton" type="submit" onclick="useAllPhrases()" value="Use All Phrases" />
</body>
</html>
```

That’s it. You now have fully styled buttons with hover capability which call the appropriate trigger. You have an awful lot of control within this element to format your screen. We will be exploring this feature in great depth later in this project.