White Paper

Using Modals in a VisualVault Implementation

**Created by** Jason Hatch

**Date:** 2/24/2020

Table of Contents

[Table of Contents 1](#_Toc33440175)

[1 Introduction 2](#_Toc33440176)

[2 The Challenge 2](#_Toc33440177)

[3 The Solution 2](#_Toc33440178)

[3.1 Creating the Modal 3](#_Toc33440179)

[3.2 Loading the Modal with Dynamic Content 5](#_Toc33440180)

[3.3 Cleaning the Modal Content 6](#_Toc33440181)

[3.4 Canceling the Model Screen 6](#_Toc33440182)

[4 Summary 6](#_Toc33440183)

# Introduction

In the normal course of developing solutions using VisualVault, you may run into the circumstance where a message or confirmation dialog box are not enough to prompt users through their given process seamlessly. You may need their input to take the most intelligent, next step in the process. The purpose of this white paper is to document how you can implement a robust dialog box or modal that dynamically populates with information from the solution and provides mechanisms to interact with the dialog box.

# The Challenge

Specifically, what is the challenge we need to resolve. This example resolves a fundamental issue when users interact with solutions built in VisualVault. The fundamental issue is I need to find duplicate records within VisualVault, display the list of records to the user and allow the user to choose which of the duplicate records returned will be used to take an action related to the current record. The action may be to relate the current record to one of the duplicate records. It may be that I will redirect the user to the duplicate record instead of allowing them to create another record.

Another example is that the solution may have multiple actions that could occur. We want the user to select one button on the form, do something in the web service and provide several options for what they can do. Today we would need to build this into the form. This solution would allow us to dynamically show a dialog box with buttons prompting for one of the other actions.

The key is that we need to provide information from VisualVault to the user and invite them to take an action that they cannot take through actions they may take on a form.

# The Solution

The solution to this challenge is to show the user a modal dialog box that we can dynamically control the content of the box using jQuery. In order to implement this solution, you will need 4 parts to the solution. Those parts are as follows:

* Create a script that creates the modal. This can be created onload or when an event occurs by the user on the screen.
* Insert code that will use a response from the server to insert dynamic actions into the modal.
* Cancel code to clean up the modal and close it.
* Code to clean up the modal after it is displayed so it can be cleanly used in the future.

The following sections will document the code that will be used for this solution. Please refer to the implementation library for the actual examples.

## Creating the Modal

The first part of this process is to create the modal. You can do this when the page is loaded or when the user interacts with the form. We recommend that you load it closest to the time that you will need the dialog box. Maybe you have some code in the onload event that measures the status of the form so it is only loaded during certain states of the form record. The most important part is to make sure if you load and use the modal that you clean it up properly after an action is taken in case the modal will be used multiple times during a single session by the user.

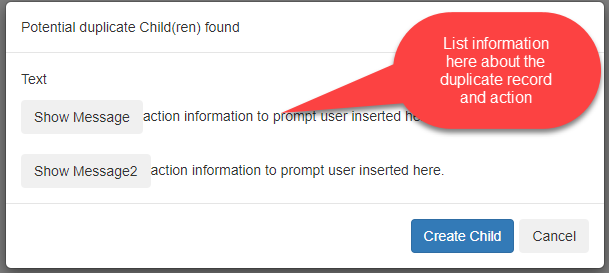
The image on the following page documents the different important sections of the Create Modal.



## Loading the Modal with Dynamic Content

The code that you will use in this section becomes what you need it to be to solicit a response or interaction from the user. You can display appropriate text messages or information from VisualVault to lead the user through the right response. This code is executed as part of the chain of code that executes to help get the right contextual information. What we mean by this is that the user selects a button, we validate data entry, call a web service that acquires information that the web service returns to the client. Then with that information, you load appropriate actions into the modal.

As an example, if the web service returns a list of duplicate records, I can add a button and then text about the duplicate record to help the user make a decision about what to do.



When inserting this information, you will insert using jQuery and HTML markup to help format the data in a way that is usable and readable by the user. The following is an example of code to load the message and buttons found above.

$('<p class="usermessage">Text</p>').appendTo('#messagearea');

$('<button type="button" class="btn btn-action" onclick="VV.Form.Template.RunConfirm(\'Yes\');">Show Message</button>action information to prompt user inserted here.<br><br>').appendTo('#buttonArea');

$('<button type="button" class="btn btn-action" onclick="VV.Form.Template.RunConfirm(\'Yes\');">Show Message2</button>action information to prompt user inserted here.<br>').appendTo('#buttonArea');

$('#childDuplicateModal').modal({ backdrop: 'static', keyboard: false });

Notice from the code above that we are assigning a class to the items added. This is used to control formatting as well as find the items to remove later. Also notice that the parameters passed into the RunConfirm form template function have the single quotes escaped. You will insert descriptive text after the button to communicate information to the user and invite them to take action. The text on the button can also be used to prompt for an action like “Redirect”, “Join”, “Create”, etc. Finally, notice the last line of code is showing the modal. The name used is the one that was used in the create logic to identify the modal.

## Cleaning the Modal Content

For every action that is taken in the scripts loaded into the modal, whether loaded as part of the process of creating the modal or actions that will be taken, you need to include code that will clean up the content that is dynamically inserted as part of the code example above. In order to clean the information inserted, use the following two lines of code.

//Clear out the message div.

$(".usermessage").remove();

//Clear out the div that contains action buttons.

$("#buttonArea").empty();

## Canceling the Model Screen

When canceling the modal screen, you need to take actions to also clean up the content and prepare the modal for the next time the user may take an action to show the modal. The code that you could use for this action are as follows:

//Call the function that contains the CleanModal code.

VV.Form.Template.CleanModal();

//Hide the modal that is displayed.

$('#childDuplicateModal').modal('hide');

# Summary

The above solution will allow the implementation team to dynamically build screens for users to interact with the solution. It will allow for flexible, create mechanisms to prompt users to take actions in the context of existing process, improving the user experience to the end user.