# Quality Assurance Checklist by Ticket types:

## Form Configuration

* Do all data entry fields have a name?
* Are all fields and buttons 508 compliant? This refers to the accessibility of technology to people with disabilities. Each data entry and button control should have text in the Accessibility configuration property.
  + The text should match the wording of the control’s label.
  + If the control is required, put the label, then type "field Required" after the label.
* Is Form ID formatted correctly?
  + Prefer Prefix in CAPS and indicative of the process. Leading zeros in sequence.
* Preview the form to make sure all labels are:
  + Rendered clearly and don’t overlap other labels.
  + Labels should not be truncated.
  + Check for spelling errors.
  + They should not wrap next to single line text boxes.
  + Fonts, heading colors and bolding are uniformly used throughout the form.
  + Layout is the same as other forms in the system.
* Preview to make sure all fields are:
  + At least 30px from the right-side margin of the form, to ensure that field validation error messages appear to right side of the field, not wrapped underneath.
* Tab Order: all controls should have a tab order of 0 to tab from left to right and top to bottom on the screen or the tab order should be explicitly set. Preview the form to ensure the tab flow makes sense.
* Containers: For any containers that are public facing or accessed by field staff, ensure that a non-default responsive flow option is selected on all containers that contain more than one control. 1 column or 2 column are the most often appropriate selections.
* Groups and Conditions:
  + Are there three standard groups for:
    - Admin: Security permissions for is visible to VaultAccess only. ONLY the admin override and admin save button container should be added to this group.
    - Hidden Fields: Is Visible conditions for Admin Override equal true. The standard hidden container should be in this group. Form field controls specific to the solution should also be hidden; it is acceptable if they are in their own group.
    - Read-Only Fields: Is ReadOnly conditions for Admin Override equal false. The standard read-only container should be in this group. Any other fields or containers that are read-only by default should also be added to this group.
  + Is each control or container in only one group?
  + Based on Field Validation Rules section of the design document, are fields visible and read-only under the correct circumstances?
    - Ensure that Admin Override allows admin users to access fields as needed.
* Upload Buttons:
  + Are all upload buttons configured for simple upload?
  + Are all index field mappings configured if required?
* Drop-Down and Query Configuration
  + Do all drop-downs contain the information required by the spec?
  + Are queries checked into GitHub?
  + Do all query names follow the documented naming convention?
  + Do all queries that rely on other form data consider Status fields, start and end date fields, and other relevant data points to ensure that only appropriate records are returned?
  + Are queries sorted alphabetically by display value unless specifically required to be sorted another way?
  + Are all queries used on forms created by the developer for this purpose? The automatically created VisualVault form query should not be used.
  + Do queries that drive drop-downs only return the fields that are needed for the form purpose? Should not see SELECT \* except where all fields truly need to be returned.
  + Do all queries return information? Where @Value is used, has data been staged to represent that value? Where a standard query is used, has testing data been staged? If no testing data has been staged, the developer has not tested appropriately.
  + Are cascading drop-downs or data lookup events configured?
    - Are data lookup events configured in the properties panel, not as event scripts? If configured as an event script, is the implementation reasonable? I.E. Only populating the value property of the drop-down into a text field, not making a server-side API call.
  + Are all drop-down fields an appropriate width, based on the possible values?
* Repeating Row Control Configuration
  + Is the repeating row control configured to the design specification?
  + Is each column an appropriate width?
* Preview the form to make sure the flow is intuitive for the process.

## Field Validation Rules

* Has the FormValidation template script been checked into GitHub?
* Validation scripts should list the fields in alphabetical order.
* Make sure blur events occur on each field.
  + Blur events should be single lines of codes or very simple.
  + If the logic is more than a handful of lines, was the logic checked into GitHub?
* Make sure required fields react based upon the Form Validation Rules section of the specifications document.
* Make sure each field accepts the correct range of information.
  + Phone number fields should only accept phone numbers, even when not required.
  + Zip code fields should only accept zip codes, even when not required.
  + Email addresses, web addresses, currency, FEIN, SSN; same as above.
* Make sure fields that accept information formatted in a certain way auto-format the information prior to field validation.
* Error messages should not appear on read-only fields that cannot be filled in directly by the user.
  + If the read-only field is populated by another field on the form that the user should input or correct, the error should be placed on that field.
* Make sure the error messages going to the user are spelled correctly and informative.
* Make sure error messages are being cleared out appropriately.
* Make sure business logic is not duplicated in multiple places.
* When retrieving information from drop-down lists, are the specialized functions to get text and values from the drop-down being used?

## All Code and Web Services

* All Code
  + Are any unnecessary code or variables present?
  + Appropriate comments in code and a comment at the top identifying where the code is used. A comment at the top should also identify the purpose of each parameter if parameters are passed in.
  + Make sure business logic is not duplicated unless there is a technical reason.
  + Are the names clear and informative of the purpose of the function? Does the name closely reflect the Business Rule section in the specification document?
  + In client-side code, when checking if a field is blank or something else, CentralValidation or another appropriate global script must be used. Do not use VV.Form.GetFieldValue('FieldName’) == ‘’ because text fields may also have a single space in them.
  + In client-side code, ensure that only appropriate actions are being taken. Any actions that need to remain secure should occur server-side.
    - For example, emails should not be sent directly from client-side code.
* All Web Service Scripts
  + Is the web service in GitHub with the same name as it appears in VV?
  + Are the libraries declared first?
  + Is the authentication function second with credentials?
  + Do the comments follow the pattern of Script Name, Customer, Purpose, Parameters (Names, types and purpose), Return Array, Pseudo Code (Required when reusable), Date of Dev, Last Rev Date, Revision Notes?
  + Are all areas of the comment header filled out accurately?
  + Are revision notes up to date?
  + Logger.info statement present to log the process started.
  + Configuration variables are at the top of the script. Examples if configurable variables:
    - Form Template names
    - Custom Query Names
    - Numbers or settings that may change in the future.
    - Variables that indicate whether the script is in a test state or production state.
  + Form Template Name is used instead of the GUID in the configuration section when identifying names of templates used. Name of variable should be descriptive of the form template. It should not be generic like FormTemplateID.
  + Are all variable names reflective of the information being stored?
  + Before performing a query or filter, are apostrophes appropriately escaped when data potentially can contain apostrophes?
  + When making API calls, does the code adequately handle positive and negative results?
  + Are there adequate comments throughout?
  + Are the error messages relevant to the context of the code?
  + Make sure the messages going to the user are not misspelled and are informative. User-facing messages should also be in a readable format.
    - For example, if a “not unique” error returns from a save, will the user know which fields need to be updated?
  + Are all communications returning to form events returning as an array with [0] being the status and [1] being a more detailed description?
  + When using async/await patterns, used a rudimentary for loop, for…in or for…of loop to process records.
  + Catch area, is there a logger.info to log the error?

## Form Web Service

* Ensure three elements are uploaded to GitHub with names that match what’s in VV:
  + Button click event
  + Template Script
  + Web Service Script
* Button Click
  + Make sure that business logic that is only specific to the button being selected is present.
  + Click events may have confirmations, call validation scripts and handle errors in the current context.
  + When retrieving information from drop-down lists, are the specialized functions to get text and values from the drop-down being used?
* Template Script
  + Messages going to the user should be spelled correctly, grammatically correct, and informative. User-facing messages should also be in a readable format.
  + If a save occurs, it should be the last thing and only on positive outcomes or when documented in the specifications document.
  + Make sure error handling has enough information to tell the user that there was an error.
  + API calls to run server-side logic should happen once under most circumstances during a single event.
* Web Service Script
  + Follow the standards above under All Code and Web Services.
  + Name of web service starts with the name or recognizable abbreviation of the form. Then it is followed with a name that reflects the business rules section so that it is recognizable and distinguished from other processes that may have similar names.
  + Make sure error handling has enough information to tell the user that there was an error.
  + Messages going to the user should be spelled correctly, grammatically correct, and informative.
    - For example, if a “not unique” error returns from a save, will the user know which fields need to be updated?
  + Thoroughly test the unique verification query.
* Test
  + Does the process address all task requirements documented in the Design Doc, any Sprint Spec Docs, and any other instructions provided to the developer?

## Scheduled Process Script

* Follow the standards above under All Code and Web Services.
* Name of the web services should start with the name or recognizable abbreviation of the form, followed by SCH, followed by a name that reflects the business rules section so that it is recognizable and distinguished from other process that may have similar names.
* Is there an API call at the end to communicate that the process has ended and a response.json at the beginning to tell VV that the process is running?
* Is the scheduled event configured to run at appropriate intervals?
* Is there a process for communicating with the appropriate users that the process has not been run or that errors occurred during the process?
* Does the process address all task requirements documented in the Design Doc, any Sprint Spec Docs, and any other instructions provided to the developer?

## Library script

* Follow the standards above under All Code and Web Services.
* Name of the web service should start with Lib. If it is a solution-specific library, then an abbreviation of the project name should follow Lib, followed by a name that reflects the purpose of the script and the business rules section of the design doc or specifications. If the library is not specific to this solution, then the naming convention should follow the pattern identified in Project Coding Standards.
* Are passed-in parameters validated prior to accessing the .value property?
* Are only necessary parameters required, while other parameters are optional?
* Does the process address all task requirements documented in the Design Doc, any Sprint Spec Docs, and any other instructions provided to the developer?

## Configure Form Tab and Context Menu Layout

* Make sure tabs reflect how the specification is configured.
* Make sure context menus reflect how the specification is configured.

## Configure security

* Make sure the security reflects the specifications document.

## Process Testing

* Create user accounts from the perspective of end users who will use the process.
* Make sure the flow is intuitive for the process.
* Were menu items and portal dashboards set up as part of the process?
* When finished, assign this ticket to a QA resource.