Generic Workflow Engine Project

## Overview

This document covers the design of the Generic Workflow Engine (GWE). The standard tools for building workflows in SharePoint include the SharePoint designer workflow features or the Windows Workflow engine with visual studio. The GWE project is geared to serve as a simple solution for a large volume of common infopath forms/workflows with the following features:

* Work with any infopath form/form library
* Can be maintained by administrators without the need for any coding
* Improved portability/reusability across multiple workflows and form libraries
* A stateless workflow engine which is easier to maintain/administer than the out-of-box workflow engine
* Capability to add plugins for performing custom actions above and beyond the capabilities of the core GWE

## Formal GWE Definition

An infopath workflow definition can be represented by a simple state machine diagram and described as a series of states and transitions. A state represents the status of any form within the library at any static point in time. A transition represents the mechanical procedures that occur after the form is submitted by a user and before it reaches the next static state. The following assumptions are made:

* Forms can have unlimited number of states and a single initial state
* Between any given states A and B inside a workflow definition a maximum of TWO transitions should exist (a->b, b->a)
* The submitted form supplies the information on which transition to execute
* All forms within a given form library follow the same workflow, therefore a form library can only be associated with a single workflow definition.



Transitions perform the following work:

* + Send template emails
  + Log workflow activity
  + Manipulate specific form fields
  + Adjust permissions of the form
  + Expose transition event hooks to external custom observer classes (allowing for extensibility)

## GWE Mechanics

**Installation/Deployment**

1. Users design form template using InfoPath form designer.
2. Administrator creates form library and deploys form template.
3. Administrator links GWE to the form library as an event receiver.
4. Administrator opens GWE configuration and defines all of the transitions within the workflow.

**Standard usage**

1. User submits the form (create/update)
   1. *Note: User has the ability to save the form without actually triggering any workflow transition.*
   2. The form contains a field dictating which transition to perform
2. Form is added/updated within the library
3. GWE workflow engine is asynchronously triggered after the addition/update is completed by SharePoint
   1. GWE observes the form to find out which transition should be performed
   2. GWE reads the workflow configuration associated with the library and gathers what to do for the given transition
   3. GWE performs the work
      1. Form fields are updated
      2. Activity logged
      3. Emails sent
      4. Permissions updated
      5. Execute any custom hooks
4. Transition complete, form resides in static state until next submission (return to step 1)



## Workflow configuration template syntax

Quite often there are fields inside of the form that need to appear in the content of emails, logs and permission sets. We use a simple templating syntax to provide this ability throughout the workflow. Each transition configuration contains text with variables that can be replaced with form fields during execution. For example, we could send an email containing the following message:

*Hello* ***${[my:fields/my:approverName]}*** *would you please approve this form?*

In this case, if the form contained an “approverName” field matching the xpath above, the form’s value would be filled in to the email before sending. This templating syntax is used throughout the entire workflow configuration.

## Workflow Configuration Data Structure

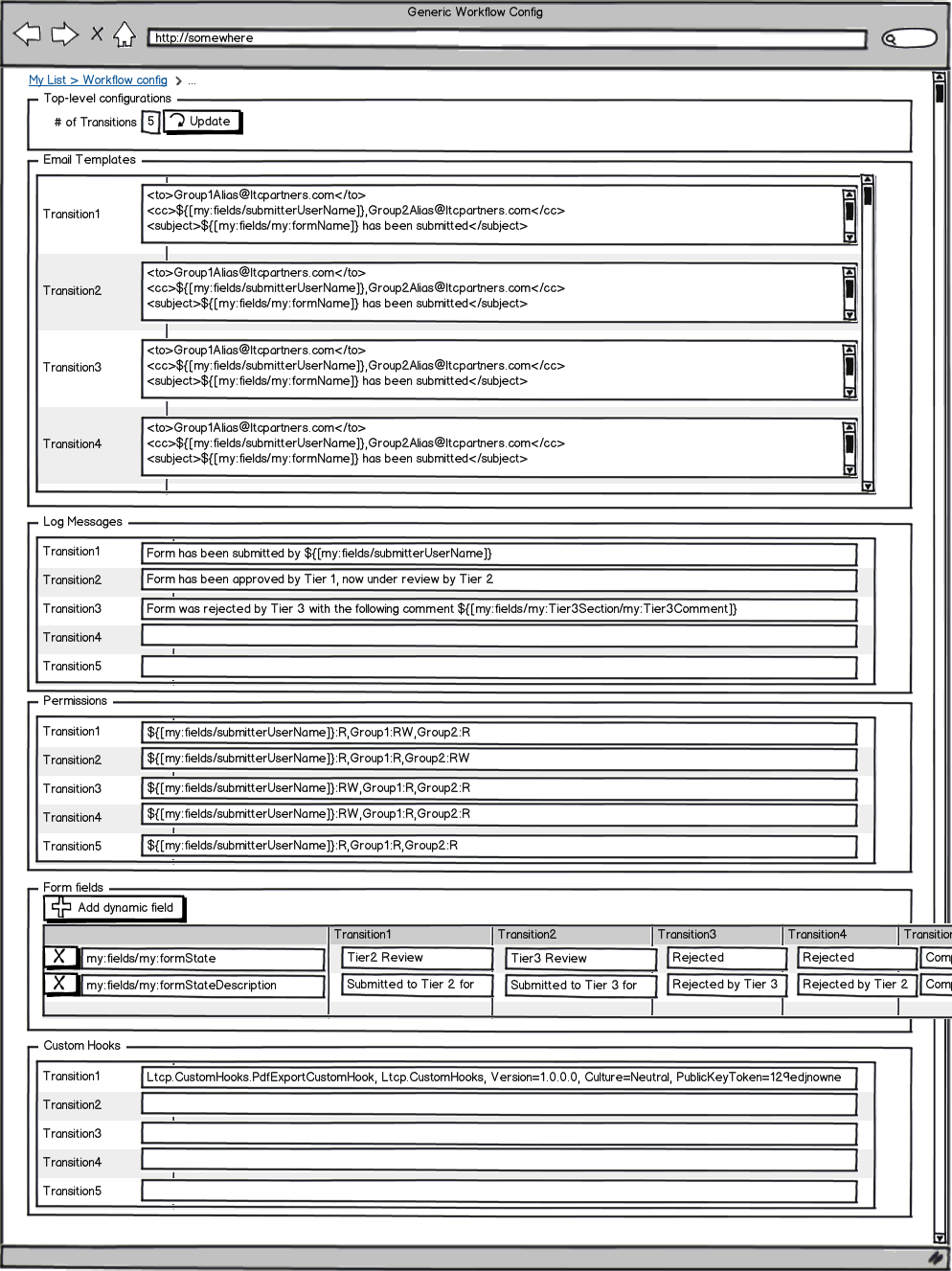
* **From Email address (string)** – A valid email address to use when distributing emails from the workflow. If this is blank, the default SharePoint SMTP server configuration from address will be used. If it is invalid, an error will be logged and the workflow will not run.
* **Transitions** (collection of transitions… below is the definition of a transition)
  + **Transition**
    - **TransitionEmailTemplate** (string)– For the given transition, an email template dictates the email to send, including the subject and recipients. An email template contains mandatory <To></To><Cc></Cc> and <Subject></Subject><Body></Body> sections which allow for HTML formatted emails with variable replacement (see templating syntax section). Email addresses are expected in a comma-delimited format.
    - **LogMessage (string)** – For the given transition, a record can be logged to an activity history list.
    - **PermissionSet (string)** – For the given transition, custom list-item level permissions can be provisioned to the next state of the form. The permission set is defined in comma-delimited key/value pairs of the following format: **<Principal>:<Permission>**
      * **Principal –** The name of a user in the domain\username format OR a SharePoint group.
      * **Permission** – SharePoint has a fixed set of permission settings for any object. The following codes apply:
        + R - Read Only
        + RW - Contribute
        + A – Full control
      * Form variable substitution is allowed, but it is important the form fields accurately represent a SharePoint group or a domain user. You can also use variable substitution to represent multiple permission sets.
      * It is assumed anyone not stated within the permission set will have NO access
    - **FormFields** (collection of FormFields… below is the definition of a FormField)
      * **FormField**
        + **XPath** – The Xml path to the form field that needs to be set
        + **Value** – The value to change the field into at the given transition

Example: Transition 1: XPath=my:Fields/my:approverName, Value=”Dan” results in the “approverName” field set to “Dan” when the form is submitted and Transition1 is executed.

* + - **CustomHook** (string) –For the given transition, a fully-qualified assembly and class name that is accessible within the GAC. This class implements a simple interface that accepts a list item and performs any custom code beyond the capabilities of the GWE. Custom hooks can be thought of as plugins to the GWE. Custom hooks are called before the GWE executes workflow actions.

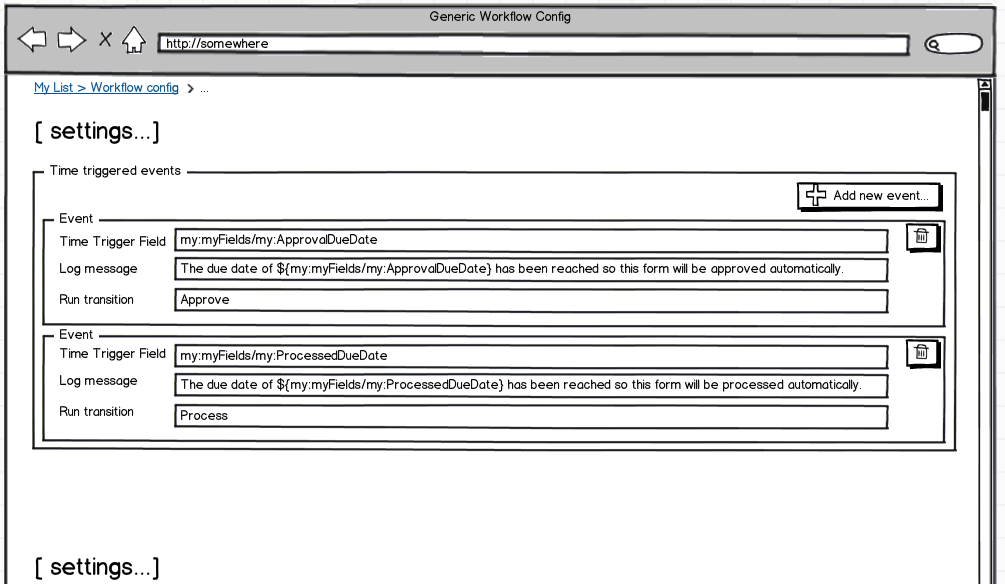
## Workflow Configuration UI

As stated earlier, the GWE comes in the form of an event receiver that must be attached to a form library. The GWE settings reside within the properties of the library. By default, a GWE has zero transitions and does nothing until configured. The GWE configuration is performed on a custom ASPX page that can only be reached by administrators under the form settings context menu.



## Time-triggered events

Requirements:  
• A GWE workflow can have multiple time-triggered events  
• For each event, the user defines a date field hidden inside the form that tells GWE when to execute the transition  
o If the date field is empty or in the future, nothing will be executed  
• For each time-triggered event, the following must be configured inside the GWE config page:  
o A valid XPATH that pointing to the hidden date field  
o The name of a transition to execute (must match one of the existing transitions already defined)  
  
From a mechanics point of view:  
• Make the GWE a farm-level feature.  
o At the time the GWE feature is activated create a timer job and install it to the farm.  
o When the GWE feature is deactivated delete the timer job.  
o Timer frequency can be adjusted in the central admin but by default we can let it run every 5 minutes.  
• The timer job scans the farm for web applications containing GWE workflows  
• For each web application, search for site collections containing GWE workflows  
• For each site collection, search for sites containing GWE workflows  
• For each site containing GWE workflows, find each GWE workflow  
• For each GWE workflow, find all the time-triggered events that should happen  
• For each time-triggered event, find the field containing the date AND the transition to execute. Then read each form  
• For each form, read the specified date field. If the date value is in the past:  
o Erase the date field (so the trigger doesn’t keep firing every 5 minutes)  
o Set the “next action” field to the specified “next action” from GWE configuration  
o Save the form (thereby triggering the GWE to perform the transition as currently designed)



## **GWE Dashboard Web Part**

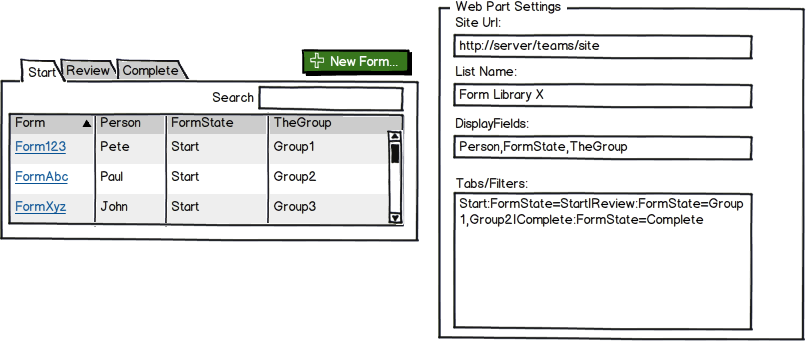
The out-of-the-box sharepoint list views are simple to use but aren’t very good at partitioning forms out into a natural format for a workflow dashboard. The GWE dashboard web part is designed so the users can separate out forms by common attributes into a tabbed interface. Each tab contains a grid view of forms where the fields displayed and their order of display is configurable. All columns can be sorted in an ascending/descending fashion in real time. Each grid has a search box at the heading that provides the ability to search any field within the grid in real time. Paging and page size capabilities are optional. See [<http://www.reallifedata.com/KeyboardGrid/demo.html>] for an example of the grid component. Selected tabs, sort orders and filters are remembered for each user such that re-visiting the page at a later point will maintain the state of their filters and selected tabs. Users can create a new form by clicking on the “new form” button which will bring up an empty form in the initial state either inside infopath client or browser form.

The web part is designed to be linked to a specific form library. Using the web part settings pane, the administrator configures the following items:

* The location of the form library containing the forms – The Site Url and the list name settings specify the location of the form library. Note that the web part can exist in a different location than the dashboard web part as long as they reside within the same farm.
* The display columns and the order – This setting is a comma-delimited collection of field names found within the form library. The order in which the fields displayed dictate the display order. The first column is always the name of the form with a link to the form view in the browser or the client.
* The tabs, their labels, filters, and order of display – This setting uses a pipe (‘|’) delimited syntax of the following tab definition elements: <TabName>:<Form Field>=<Value1>,<Value2>…<ValueN> where
  + TabName – the label of the tab
  + Form Field- the name of a field from the library to filter
  + Value1,2,3… -A comma delimited list of values that serve as the filter

In the example below, we have configured the web part to display forms from the library named ‘Form Library X’ inside the SharePoint site ‘http://server/teams/site’. The grid displays the form fields ‘Person’, ‘FormState’, and ‘TheGroup’ as columns. There are three tabs:

* A tab labeled ‘Start’ which contains all forms where the field named ‘FormState’ is equal to “Start”
* A tab labeled ‘Review’ which contains all forms where the field named ‘FormState’ is equal to “Group1” or “Group2”
* A tab labeled ‘Complete’ which contains all forms where the field named ‘FormState’is equal to “Complete”



GWE Deliverables

* All source code
* WSP containing GWE assemblies and custom UI
* Documentation