iConnect 180745 Design Approach

It is common for copied applications/activities using the Copy with Changes option to have issues associated with them. Issues may include:

- Funds not existing on the product show up in the App/Sub and outbound data.
- Applications fail in production.
- Extra data interferes with rules, so rules are written to clear out data on copied applications.
- Fields are erroneously being set to disabled/read-only.
- Forms are removed while the activity is being edited; data from removed forms is transmitted on submit.

When Copy with Changes is initiated, the original activity's (e.g. application, illustration) unused data is being added to the new activity at the time it is created. This data is not relevant and should be removed.

Project Overview

The goal of this project is to remove unused data entered by the user in copied activities. This will occur when:

1. Activity is created – All unused data items will be removed when a copied activity is created. This applies to new cases that are created with "Copy As Is" and "Copy with Changes".
2. Activity is locked – To prevent removing data that is needed we will set a flag if any unused data exists when an activity is locked. Providers can then exclude data items as needed (one-time change to provider). This applies to all activities when locked.

Features/Requirements

The following will be added to clean up unused data in a copied activity.

1. Add a Source property to DataItem with data type:
   ```
   public enum DataItemSource
   {
   Unknown = 0
   User = 1
   Provider = 2
   Base = 3
   Rule = 4
   }
   ```
2. Modify DataItem.SetValueEx and Initialize method to accept an additional source parameter, defaulting to Unknown.
3. Modify calls that set base FLI_ data items in ApplicationMagager.CreateDefaultData to pass Base to SetValueEx source parameter.
4. Modify IntegrationProviderBase.UpdateDataItem and DTCCProviderBase.UpdateDataItem to pass Provider to SetValueEx source parameter.

5. Modify DataItem.Value property set to set Source property to User.
   As discussed, set DataItemSource to User on EditApplicationModel.UpdateField and EditWizardControllerBase.UpdateField instead of DataItem.Value.
6. Modify BaseDINode to pass Rule to SetValueEx source parameter.
7. Create a ApplicationManager.FindUnusedDataItems(Guid appId) method, which returns a list of any data items with Source == User that are not on any included forms or wizards.
   a. Also, update the method signature with two additional parameters, which are required to find unused data items.
      FindUnusedDataItems(Guid ApplicationId ,Dictionary<string, DataItem> FormData, IList<PackageItem> IncludedForms)
8. Modify copy with changes logic to remove any data items returned by FindUnusedDataItems.
   Note: This should be done "*before*" calling the provider copy and initiate data items.
9. Add a bool Unused property to DataItem class.
10. Modify application lock logic to set Unused = true for any data items returned by FindUnusedDataItems. These items *will not be removed*. Providers can then use the Source and Unused properties to remove data items as needed.

Use Cases / Workflow Changes

Admin Changes

No changes.

UI Mock Ups

N/A

How to Enable and Use This Feature

To test this item you will need to have an application or another transaction saved that has unused data items. When you copy it and submit it, you will need to check the outbound data to be sure the unused data items were removed.

Areas Impacted

System Area	Yes	Comment
Admin Tool		
- Form Library		
- Design Forms		
- Profile Administration		
- Reports		
- Deployment		
FireLight App		
- New Application		
- Edit Application		
- Signature Process		
- Review Queue		
- Manual Review		
- User Preferences		
- Inbound Integration		
- Outbound Integration		
- PDF Generation		
- Email System		
FireLight Console		
- Windows		
- iOS		
Other Systems		
- DTCC Integration		
- Commission Netting		
- Activity Reporting		