Masking 2

Masking will customize the display of a field, when the full value cannot be shown to the user of a lookup, inquiry, or document. When masking occurs on a document, the document authorizer is consulted so both the user and the state of the document may be taken into account.

When masking a value in a lookup, inquiry or *maintenance* document via the data dictionary, there is a tag to specify the mask with three value options:

1. Literal String.
   E.g., 'Not Displayed', or '**********', or "...

```
<displayMask><maskLiteral literal="**********"/></displayMask>
```

2. Preset tag for masking up to a certain number of characters.

```
<displayMask>
   <maskTo maskCharacter="*" maskLength="6"/>
 </displayMask>
```

   This would mask the first 6 characters of the string, with the actual characters of the remaining string displayed as is.
3. Specify maskFormatter class.

```
<displayMask>

<maskFormatter>org.kuali.module.financial.mask.MaskSSNFormmatter</maskFormatter>
 </displayMask>
```

   This class must implement the MaskFormatter interface. The framework will send in the actual value, and the implementation can implement any masking needed.

Lookups & Inquiries

The lookup and inquiry frameworks check the authorization tag of a field before it is displayed in a result table or inquiry page. If the user is not in the required workgroup, the mask string will be displayed. In addition, no sub-inquiry links will be rendered on the field (if applicable).

To mask a value in lookups and inquiries, simply specify a workgroup and mask in the attribute definition in the data dictionary. For example...

org.kuali.module.financial.datadictionary.Payee.xml

```
<attributeAuthorization>
    <displayWorkgroup>KUALI_DV_TAXGRP</displayWorkgroup>
    <displayMask><maskLiteral literal="**********"/></displayMask>
</attributeAuthorization>
```

What if a masked fields is part of the key for another inquiry? Because the URL for a given inquiry page is generated by KFS, KFS uses the encrypted value in the URL itself, therefore protecting that data from being viewed.

A secure field is displayed without restrictions if specified as a lookup search field in the data dictionary xml file. If the need occurs to not display the field later on, we will add an attribute to the lookupField tag to specify the authorization. The need for a different workgroup on the search field could also be needed. If the user does not have permissions on the field, it will still show masked in the result set.

⚠ The masking functionality is a very powerful and useful feature. However, it is not all-encompassing. Here are a few things to keep in mind regarding the masking service:

- The masking framework does *NOT* encrypt the fields in the database.
- Masking only applies to fields that are being passed around by forms or within GET urls.

Maintenance Documents

Maintenance documents use the maintenance document data dictionary definition to specify that fields on a document should be masked. When a `<maintainableField>` is defined for a sensitive field, it is associated with an edit mode. Note: multiple sensitive fields may be associated with the same edit mode, but each sensitive field must be associated with exactly one edit mode.

For example, the following code defines the taxIdNumber field of the Payee BO as a secure field.

```
<maintainableField name="taxIdNumber">
    <fieldAuthorization>
        <displayEditMode>taxEntry</displayEditMode>
        <displayMask><maskLiteral literal="*********"/></displayMask>
    </fieldAuthorization>
</maintainableField>
```

The above XML snippet may be interpreted as follows: There is a field named taxIdNumber on the document. This field is only viewable (and editable, if appropriate) if the "taxEntry" edit mode is in the map returned by the document authorizer's getEditMode(). If the edit mode is not in the map, then the "*********" masking string will be used to hide the sensitive value.

The Disbursement Voucher Payee maintenance document uses org.kuali.module.financial.document.authorization.PayeeDocumentAuthorizer as its implementation of DocumentAuthorizer; it's override of getEditMode() shows how the "taxEntry" edit mode is exported:

org.kuali.module.financial.document.authorization.PayeeDocumentAuthorizer

```
1.  public Map getEditMode(Document document, UniversalUser user) {
2.    Map editMode = super.getEditMode(document, user);

3.    if (user.isMember(SpringContext.getBean(ParameterService.class)
          .getParameterValue(DisbursementVoucherDocument.class,
KFSConstants.FinancialApcParms.DV_TAX_WORKGROUP))) {
4.      editMode.put(AuthorizationConstants.DisbursementVoucherEditMode.TAX_ENTRY,
"TRUE");
5.    }

6.    return editMode;
7.  }
```

In line 3, the method checks if the current user is a member of whichever workgroup is specified by KFSConstants.FinancialApcParms.DV_TAX_WORKGROUP for the DisbursementVoucherDocument class; if the current user is a member of that workgroup, then on line 4, the "taxEntry" edit mode is added to acceptable edit modes, and therefore the field will not be masked/encrypted. Users not within the specified DV_TAX_WORKGROUP will see the max and have an encrypted variable.

By default, the framework does not mask fields when the following three conditions are true:

1. the maintenance document is for creating a new record or copying an existing record (note: the masked field value is not copied over from the old record when using copy to create a new record)
2. the maintenance document is in the "initiated" or "saved" state
3. the user viewing the document is the initiator of the document

The rationale for this is to allow people to edit all fields when creating/copying a record, and to route it for approval. However, during the approval process, the approver may decide to enter another value for the sensitive field. This mechanism allows the approver's change to not be visible to the user, unless the document authorizer is rewritten to allow it.

Transactional Documents

For documents other than maintenance, the edit mode in which to display and the mask should be given as attributes to the htmlControlAttribute.

work/web-root/WEB-INF/tags/fin/procurementCardAccountingLines.tag

```
      <kul:htmlControlAttribute
attributeEntry="${cardAttributes.transactionCreditCardNumber}"
        property="document.procurementCardHolder.transactionCreditCardNumber"
readOnly="${fullEditMode}"
        displayEditMode="${cardDisplayMode}" displayMask="*********"/>
```

displayEditMode and displayMask will only be examined if the field is read only.

To mask fields other than literals (as dd tags), jstl expressions can be used for the displayMask value.

The Document Authorizer would then have to return a correct edit mode for the field to be unmasked.

Unable to render {include} The included page could not be found.