KIM Conversion: Person - KIM

The following is the conversion to migrate the Person table(s)/view(s) to KIM tables. This document does not address AuthZ or AuthN migration.

# Foreign Keys to Person

The foreign keys can remain unchanged since we will not be removing the Person table in this proposed solution. We may want to look at what references the Person table but does not have a foreign key. There are potential foreign keys that are “missing” in various tables. Not only would this help integrity-wise but it would also help our database schema make more sense at-a- glance without referring to O/R Mapping configuration.

# Views into Person

Rework/Delete Views {OSP$Person, OSP$User}

Reworking these views may be difficult due to the complex KIM structure. For example: Some entity data will need to be stored in a generic KIM table. To get a MOBILE\_PHONE\_NUMBER for entity ‘1’ (SELECT phone.PHONE\_NBR FROM ENTITY\_T entity, KRIM\_ENTITY\_PHONE\_T phone, KRIM \_PHONE\_TYP\_T type WHERE phone .ENTITY\_ID = entity.ENTITY\_ID AND phone. PHONE\_TYP \_CD = type.PHONE\_TYP AND entity.ENTITY\_ID = ‘1’ AND type.PHONE\_TYP = ‘MBL’)

Also, in places old KIM and new KIM store data in different formats. For example a compatible view would need to convert state code “MI” to the abbreviation “Michigan” or country code US to USA

# Standard KIM Entity Data

KIM supports a standard set of information about each entity. Each set of related information is normalized into a separate table(s). Many of these tables also reference type tables not specific to ENTITY. For example: phone information is located in table KRIM\_ENTITY\_PHONE\_T. This table refers to KRIM\_PHONE\_TYP\_T because there can be many types of phone numbers (work, home, etc.).

The “TYPE” tables are similar to KCs code tables where they come with a standard set of bootstrap data which is easily maintainable. For much of our data migration, we will have to decide whether we add a new “TYPE” or use a generic “Other” type which is usually available out-of-the-box.

# KIM Extended Entity Data

**At this time KIM does NOT support extended information about an entity. This is something that KC needs in order to convert to KIM.** This is also something KS needs as well. From initial discussions, this feature will look like a table containing a generic key-value pairs. No other information is known at this time.

# ENTITY PRIVACY IN KIM

For each Standard KIM Entity Data type, KIM allows data suppression via a suppress flag in KRIM\_ENTITY\_PRIV\_PREF\_T. External entity id’s in KRIM\_ENTITY\_EXT\_ID\_T can also be encrypted via an encrypt flag. Not sure if KC needs these features?

# AUDIT TRAILS

KIM will keep a record of all users in the KIM system so that users can never be completely deleted. This record resides in the KRIM\_ENTITY\_CACHE\_T but does not contain a complete set of entity information. As a result deleting a user will cause some information to be lost forever.

Change history to an entity is not captured. The ability to edit a user is not readily available therefore this is not supposed to be an issue. We still may want to explore the following scenario:

For example KIM entity id 1 in KIM might refer to Chris Denne, later refer to Terry Durkin, and then finally be deleted from the system. Any of the documents referring to KIM entity id 1 would never know these changes occurred.

# GENERAL OPEN POINTS/NOTES/STRANGENESS

* Currently a person can exist in the Person table w/o existing in the KIM\_PERSON table. This is similar to an ENTITY existing in KIM without a PRINCIPAL.
* Why does KIM have an ENTITY\_TYP\_CD field in many of the KRIM\_ENTITY\_XXX\_T tables? There is already a Foreign Key to the Entity table which also has an ENTITY\_TYP\_CD. This seems redundant/error prone unless there is some other meaning… If this is redundant data then all new entries in these tables should match the KRIM\_ENTITY\_T table’s ENTITY\_TYP\_CD
* KRIM\_ENTITY\_XXX\_T tables have active & default indicators. We should probably have all entries in these tables be active upon data migration. As for default, most entries should be default unless multiple entries exist which then we will have to choose.
* KIM has a parallel table structure. For Example: KRIM\_ENTITY\_ADDR\_T and KRIM\_PND\_ADDR\_T share the same structure. We should not have to worry about the “PND” tables. They are there for KIM administration.

## Field Conversion

|  |  |  |
| --- | --- | --- |
| PERSON | KIM | Notes |
| PERSON\_ID (VARCHAR(10)) | KRIM\_ENTITY\_T.ENTITY\_ID (VARCHAR(40)) |  |
| VER\_NBR | KRIM\_ENTITY\_T.VER\_NBR |  |
| OBJ\_ID | KRIM\_ENTITY\_T.OBJ\_ID |  |
| ACTIVE\_FLG (CHAR(1)) | KRIM\_ENTITY\_T.ACTV\_IND (VARCHAR(1)) |  |
| USER\_NAME | KRIM\_PRNCPL\_T.PRNCPL\_ID | Seems related to KRIM\_PRNCPL\_T.PRNCPL\_NM |
| UPDATE\_TIMESTAMP | N/A |  |
| UPDATE\_USER | KRIM\_ENTITY\_T.LAST\_UPDT\_DT |  |
| ERA\_COMMONS\_USER\_NAME | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| SSN | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| AGE | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) | deprecate in favor of KRIM\_ENTITY\_BIO\_T.BIRTH\_DT |
| AGE\_BY\_FISCAL\_YEAR | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) | deprecate in favor of KRIM\_ENTITY\_BIO\_T.BIRTH\_DT |
| EDUCATIONAL\_LEVEL | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| DEGREE | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| MAJOR | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| IS\_HANDICAPPED | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| HANDICAP\_TYPE | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| IS\_VETERAN | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| VETERAN\_TYPE | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| VISA\_CODE | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| VISA\_TYPE | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| VISA\_RENEWAL\_DATE | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| HAS\_VISA | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| SCHOOL | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| YEAR\_GRADUATED | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| DIRECTORY\_DEPARTMENT | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| PRIMARY\_TITLE | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| DICTORY\_TITLE | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| VACATION\_ACCURAL | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| IS\_ON\_SABBATICAL | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| ID\_PROVIDED | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| ID\_VERIFIED | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) |  |
| OFFICE\_LOCATION | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) | Assuming this is a name of a location. Do we need this? If so maybe KIM should add a location label field to KRIM\_ADDR\_T |
| SECONDARY\_OFFICE\_LOCATION | EXTENDED\_ENTITY\_DATA\_T (UNKNOWN) | Assuming this is a name of a location. Do we need this? If so maybe KIM should add a location label field to KRIM\_ADDR\_T |
| FAX\_NUMBER | KRIM\_ENTITY\_PHONE\_T. PHONE\_NBR | Must create a new KRIM\_PHONE\_TYPE\_T column for new phone type or use KRIM\_PHONE\_TYPE\_T . PHONE\_TYPE\_CD ‘OTH’ |
| PAGER\_NUMBER | KRIM\_ENTITY\_PHONE\_T. PHONE\_NBR | Must create a new KRIM\_PHONE\_TYPE\_T column for new phone type or use KRIM\_PHONE\_TYPE\_T . PHONE\_TYPE\_CD ‘OTH’ |
| MOBILE\_PHONE\_NUMBER | KRIM\_ENTITY\_PHONE\_T. PHONE\_NBR | KRIM\_PHONE\_TYPE\_T . PHONE\_TYPE\_CD ‘MBL’ |
| OFFICE\_PHONE | KRIM\_ENTITY\_PHONE\_T. PHONE\_NBR | KRIM\_PHONE\_TYPE\_T . PHONE\_TYPE\_CD ‘WRK’ |
| SECONDARY\_OFFICE\_PHONE | KRIM\_ENTITY\_PHONE\_T. PHONE\_NBR | Must create a new KRIM\_PHONE\_TYPE\_T column for new phone type or use KRIM\_PHONE\_TYPE\_T . PHONE\_TYPE\_CD ‘WRK’ |
| LAST\_NAME | KRIM\_ENTITY\_NM\_T.LAST\_NM | NM\_TYP\_CD = ‘PFRD’ |
| FIRST\_NAME | KRIM\_ENTITY\_NM\_T.FIRST\_NM | NM\_TYP\_CD = ‘PFRD’ |
| MIDDLE\_NAME | KRIM\_ENTITY\_NM\_T.MIDDLE\_NM | NM\_TYP\_CD = ‘PFRD’ |
| SALUTATION | KRIM\_ENTITY\_NM\_T.TITLE\_NM | NM\_TYP\_CD = ‘PFRD’ |
| FULL\_NAME | N/A | Full name is calculated in KIM from the first, middle, last name (not prefix, suffix, etc.) – we could use an ext field or use KIM auto calculation which could be inconsistent with Person data |
| PRIOR\_NAME | KRIM\_ENTITY\_NM\_T.LAST\_NM | Must create a new KRIM\_NM\_TYPE\_T column for new name type or use NM\_TYP\_CD = ‘OTH’. Also must fill in entries for first middle etc. since prior name will be a new row in KRIM\_ENTITY\_NM\_T |
| EMAIL\_ADDRESS | KRIM\_ENTITY\_EMAIL\_T.EMAIL\_ADDR | EMAIL\_TYP\_CD = ‘WRK’ |
| DATE\_OF\_BIRTH | KRIM\_ENTITY\_BIO\_T.BIRTH\_DT |  |
| GENDER (VARCHAR (30)) | KRIM\_ENTITY\_BIO\_T.GNDR\_CD (VARCHAR(1)) | Must convert “Male” & “Female” to “M” & “F” |
| RACE | KRIM\_ENTITY\_BIO\_T.ETHNCTY\_CD | Should convert fine but not sure if the same classifications will be used in new KIM. No code table with various Ethnic groups to confirm. |
| IS\_FACULTY | KRIM\_ENTITY\_AFLTN\_T | AFLTN\_TYP\_CD = “FCLTY” |
| IS\_GRADUATE\_STUDENT\_STAFF | KRIM\_ENTITY\_AFLTN\_T | Must create a new KRIM \_AFLTN\_TYP\_T column for new type or use AFLTN\_TYP\_CD = “STDNT” |
| IS\_RESEARCH\_STAFF | KRIM\_ENTITY\_AFLTN\_T | Must create a new KRIM \_AFLTN\_TYP\_T column for new type |
| IS\_SERVICE\_STAFF | KRIM\_ENTITY\_AFLTN\_T | Must create a new KRIM \_AFLTN\_TYP\_T column for new type |
| IS\_SUPPORT\_STAFF | KRIM\_ENTITY\_AFLTN\_T | AFLTN\_TYP\_CD = “FCLTY” |
| IS\_OTHER\_ACADEMIC\_GROUP | KRIM\_ENTITY\_AFLTN\_T | Must create a new KRIM \_AFLTN\_TYP\_T column for new type |
| IS\_MEDICAL\_STAFF | KRIM\_ENTITY\_AFLTN\_T | Must create a new KRIM \_AFLTN\_TYP\_T column for new type |
| COUNTRY\_OF\_CITIZENSHIP (VARCHAR(30)) | KRIM\_ENTITY\_CTZNSHP\_T .POSTAL\_COUNTRY\_CODE (VARCHAR(2)) | Must convert from description to 2 digit code |
| HOME\_UNIT | N/A | Seems like this field is located in a different set of KIM tables not related to Entity |
| ADDRESS\_LINE\_1 (VARCHAR(80)) | KRIM\_ENTITY\_ADDRESS\_T.ADDR\_LINE\_1 (VARCHAR(50)) | Not long enough to store old KC data. Should KIM make columns longer? |
| ADDRESS\_LINE\_2 (VARCHAR(80)) | KRIM\_ENTITY\_ADDRESS\_T.ADDR\_LINE\_2 (VARCHAR(50)) | Should KIM make columns longer? |
| ADDRESS\_LINE\_3 (VARCHAR(80)) | KRIM\_ENTITY\_ADDRESS\_T.ADDR\_LINE\_3 (VARCHAR(50)) | Should KIM make columns longer? |
| CITY | KRIM\_ENTITY\_ADDRESS\_T.CITY |  |
| COUNTY | N/A | Should KIM add this? |
| STATE (VARCHAR(30)) | KRIM\_ENTITY\_ADDRESS\_T.POSTAL\_STATE\_CD (VARCHAR(2)) | Must convert from description to postal code |
| POSTAL\_CODE | KRIM\_ENTITY\_ADDRESS\_T.POSTAL\_CD |  |
| COUNTRY\_CODE (CHAR(3)) | KRIM\_ENTITY\_ADDRESS\_T.POSTAL\_CNTRY\_CD (VARCHAR(2)) | Must convert from 3 digit code to 2 digit code |
| KIM\_PERSON\_ID | N/A | Entities will referenced to principals in another manner |

# Migration Plan

Now that we have an approach for mapping the old KIM data to new KIM we need to figure out how to use the new KIM in KC. The goal of this plan is to minimize the impact on KC and well as insulate KC from future changes to KIM.

# KIM Person & PersonService

The KIM Person Object is a little different than many BOs in rice. It is Externalizable not persistable. Since, KC’s Person object does persist it is not a simple matter of just implementing the Person interface with our existing KC Person object (because it does persist). Instead what we can do the following

1. Extend the KIM Person interface adding new methods to handle to access all the data required by KC. Then we will need to provide an implementation of that interface.
2. Provide a new implementation of the KIM PersonService to act on the enhanced Person. Internally we may want to wrap or extend of KIM PersonServiceImpl in order reuse some of the logic.
3. Override the PersonService in the Spring file.
4. Write tests for the new Service Implementation ☺

# Person table, BO, repository.xml…

Currently, Person contains a denormalized bunch of data related to Persons. We can keep the Person table but alter it to remove all non-infrastructure related columns. Then add a column to reference the KIM Entity table. This column will not have a foreign key constraint at the database level since KIM can be installed on a separate database than KC. The referential integrity will have to be taken care of at the application level. Below is an example of the new Person table in its entirety:

**CREATE** **TABLE** PERSON ("PERSON\_ID" VARCHAR2(10) **NOT** **NULL** ENABLE,

"ENTITY\_ID" VARCHAR2(40) **NOT** **NULL** ENABLE,

"VER\_NBR" NUMBER(8,0) **DEFAULT** 1 **NOT** **NULL** ENABLE,

"OBJ\_ID" VARCHAR2(36) **DEFAULT** SYS\_GUID() **NOT** **NULL** ENABLE,

**CONSTRAINT** "PK\_PERSON\_KRA" **PRIMARY** **KEY** ("PERSON\_ID") ENABLE);

Then we will keep the Person BO largely unchanged (API wise). We will do this by removing most of the private fields and using delegation. For example:

**public** **class** Person **extends** KraPersistableBOBase **implements** Contactable {

**private** String personId;

**private** String kcKimPersonId;

//this is the new extended Person interface mentioned above

**private** org.kuali.kra.kim.KCKimPerson kcKimPerson;

**public** String getLastName() {

**return** **this**.kcKimPerson.getDefaultName().getLastName();

}

}

Don’t forget to cleanup the **toStringMapper()**, **equals(Object o)**, and **hashCode()** methods (if present).

The field mapping is not a perfect match where some data is in differing formats as shown in the chart above. We have a couple options to correct this situation. We can:

1. Modify the field returned from the entity before returning from the Person. This may be a little more error prone but will limit the impact across the application.
2. We could change client code to deal with different formatted data.

The Person.xml file needs to be updated to add a relationship section. For example:

\*KFS uses these relationships a lot. See KFS for a working example.

<property name=*"relationships"* >

<list>

<dd:relationship objectAttribute=*"kcKimPerson"*

*targetClass=”* org.kuali.kra.kim.KCPerson*”*>

<dd:primitiveAttribute source=*"kcKimPersonId"* target=*"entityId"* />

</dd:relationship>

</property>

The Person.xml file may need to be updated to reflect any formatting changes in the data returned from the Person BO. For example are we now returning state codes rather than descriptions (ex: MI versus Michigan). Should we use a lookup for those codes?

Finally, the repository.xml, files will need to get updated to model the new Person BO and Person table.

Important: Since there will not be a foreign key to the KIM entity table, we have to handle the situation where the internal entity object in the Person BO is null (not in the database).

This structure has an added benefit that we will NOT have to worry about changed primary keys when doing a data migration. For example: ProposalPerson has a reference to a Person.personId. The ProposalPerson will NOT need to be updated because the Person.personId is not changing. Internally, the Person will be pointing to an entity but that is not important to ProposalPerson.

# Alternative Changes

Keeping the existing Person table, BO, etc was done for a few reasons. 1) To insulate KC from KIM changes, 2) to make it easier to move to the new KIM w/o requiring mass changes to KC, and 3) to allow for easier data migration (we don’t have to update foreign keys). It is possible to reference the KIMPerson directly. That would involve updating O/R Mappings, DD files, and source for any BO that references the existing KC Person object and update foreign keys.

# Entity Maintenance

With the migration, KC will no longer need to handle entity maintenance (for the most part).

**\*Enter reference to loading up maintenance screens\***

With, the proposed design, KC will need a way to “add” a KIM entity as a KC Person. This will be done through a simple maintenance screen which will allow a user to create a Person and associate that Person with an entity. As discussed previously a Person is really just a KC specific wrapper around a KIM Person. We will most likely have to build a custom maintenance screen because the Person BO in KC has-a KimPerson which is not a persisted BO.

# Testing, Potential Problems, etc.

Technically, the Person BO still adheres with the JavaBean standard and assuming we don’t change method signatures we should be free to remove actual private data and use delegation. In practice, some frameworks and libraries (did someone say rice) depend on private fields being present. Here is a list of some things we should remember to test.

* lookups (ex: ProposalDevelopmentKeyPersonnel.jsp, user.tag)
* lookups - multi-value (ex: budgetPersonnel.jsp)
* direct inquiry (ex: )
* dwr (ex: unitHierarchy,jsp)
* Person maintenance
* Does the system behave when a user no longer exists in KIM but is referenced in KC’s Person table?
* KC’s PersonService Implementation
* I’m sure there more…