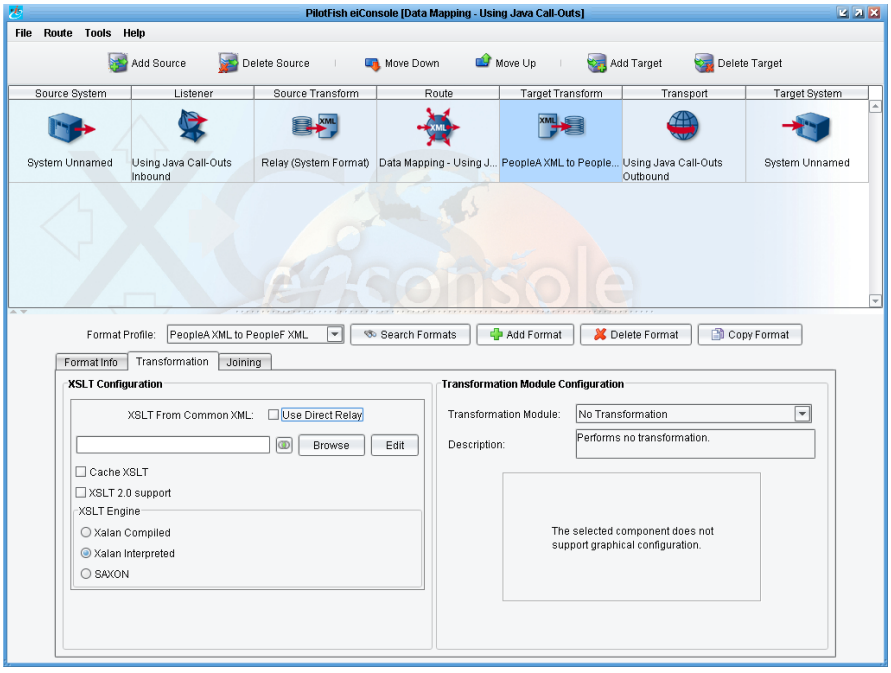
Advanced Mapping – Using Java Call-Outs

Overview

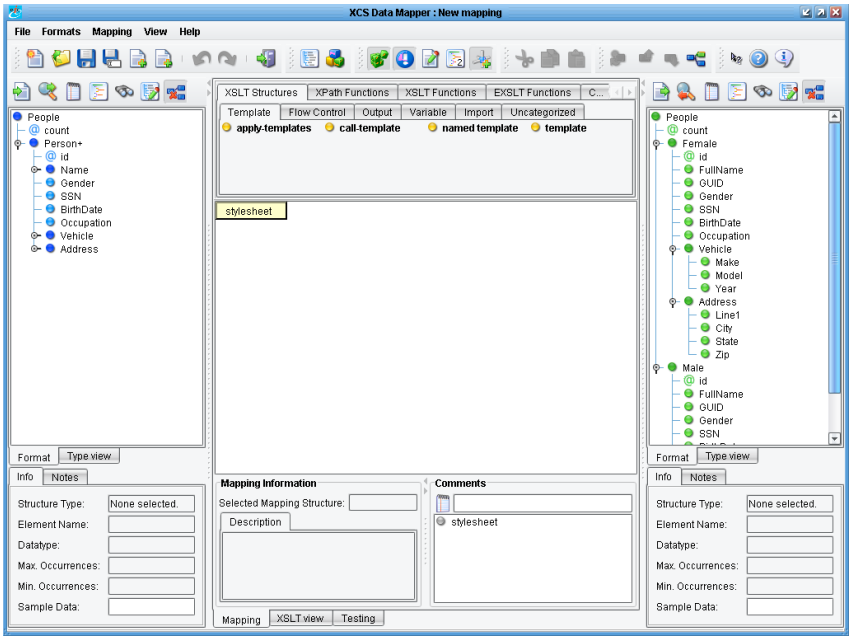
This tutorial covers the use of Java Call-Outs from within the Data Mapper to augment transformations with additional, programmatic functionality. This tutorial expands on concepts covered in "Data Mapping – Using Tabular Mappings," so users are expected to be familiar with that content.

Steps

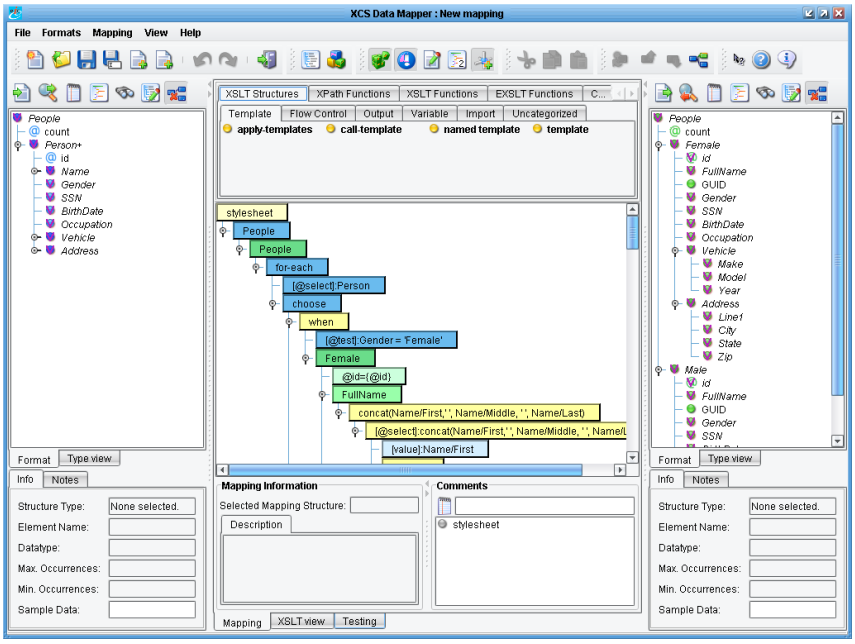
Start by creating and configuring a Route similar to the prior tutorial's, and add a new Format named "PeopleA to PeopleF" on the Target Transform:



Click “Edit” to create a new mapping. Load “PeopleA.xml” as the Source format and “PeopleF.xml” as the Target format:



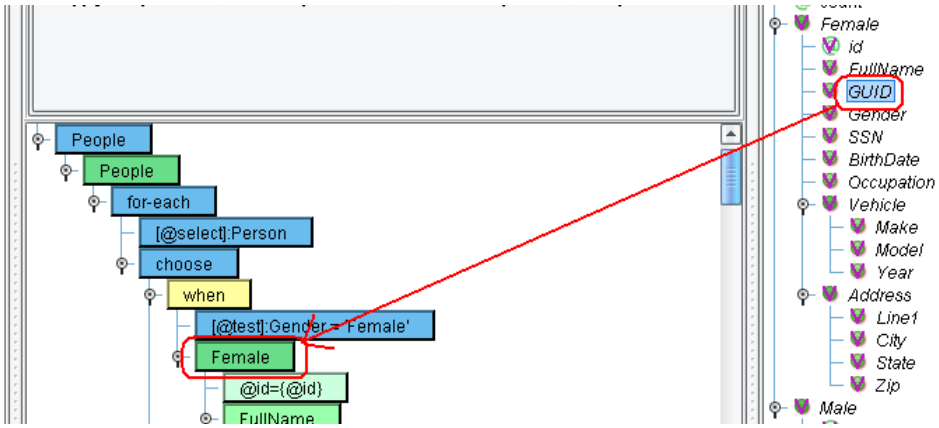
Repeat the mapping exercise from the previous tutorial or copy the resulting XSLT via the “XSLT View” tab:



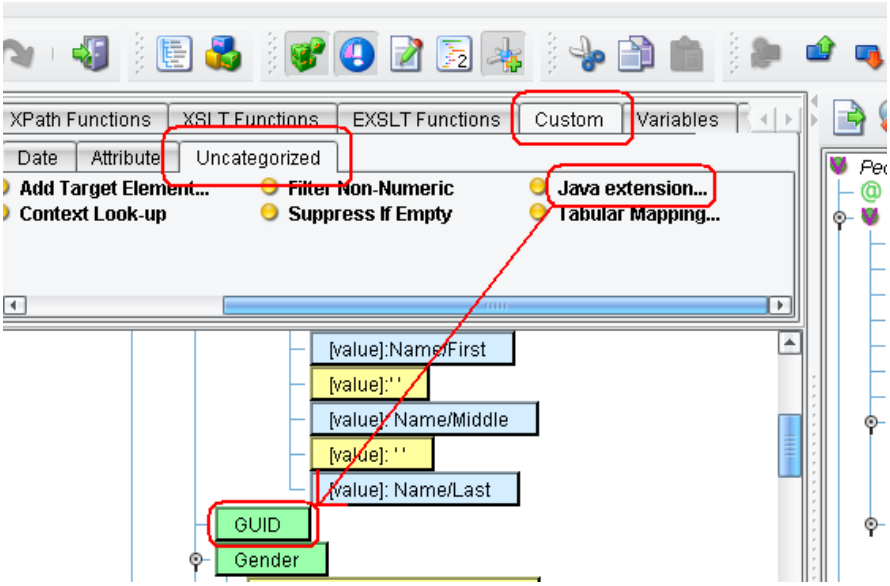
Once again, the structural differences between our previous mapping exercise and this one are minor: we've added a "GUID" field to each "Male" and "Female" element. Each Male and Female already has an "@id" attribute, but this has largely been used to represent a sequence such as 1, 2, 3, etc. We'll want to populate GUID with a value that is reasonably unique even across different documents and systems.

While XSLT is considered functionally complete, generating a UID can be staggeringly difficult and would be less than ideal for an exercise (it would, in point of fact, probably cover dozens if not hundreds of pages). Fortunately, many XSLT engines can make use of their host environments by making calls to existing code. Within the Data Mapper, that language is Java, and the user has access to all of the standard JDK classes as well as the classes and libraries utilized by the eiConsole.

First, map "GUID" from the Target format onto "Male" and "Female":



Next, select and drag Custom → Uncategorized → Java Extension from the tool palette onto "GUID":

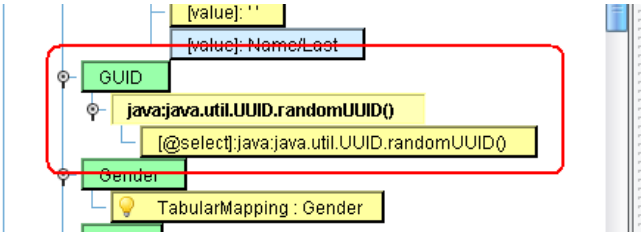


This will raise a dialog where you can provide a Java call string; that is, a Java class, method, and any number of parameters you wish later to pass it:

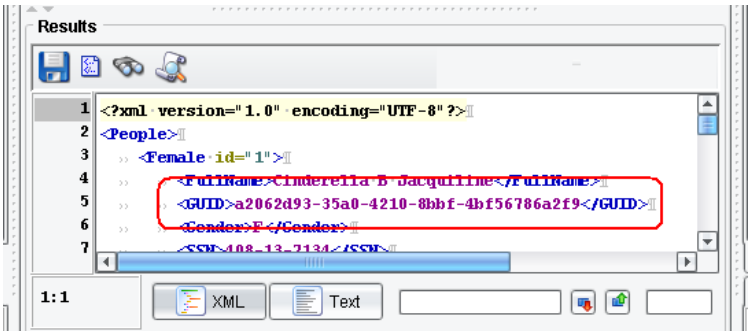
The Java API provides an extremely useful class for our case called "UUID," located in the "java.util" package. It has a static method called "randomUUID()" which creates and returns a new UUID. To invoke this class and method, provide the following for the Java call string:

java.util.UUID.randomUUID

Then click "OK." The GUID element should now have the call string inside it:



Switch to the Testing tab and execute the transformation to see the results:



If you instead receive an error, ensure that the XSLT engine is set to "Xalan (Interpreted)":