

FSA Integration Partner

United States Department of Education
Federal Student Aid



Enterprise Application Integration (EAI)

SAIG – COD

Common and Legacy Record Schools
Interface

Transformation

Internal Design Document

Document Reference Name: SAIG_COD_MSG_TRANSFORM_IDD

September 16, 2003

Document Change Control

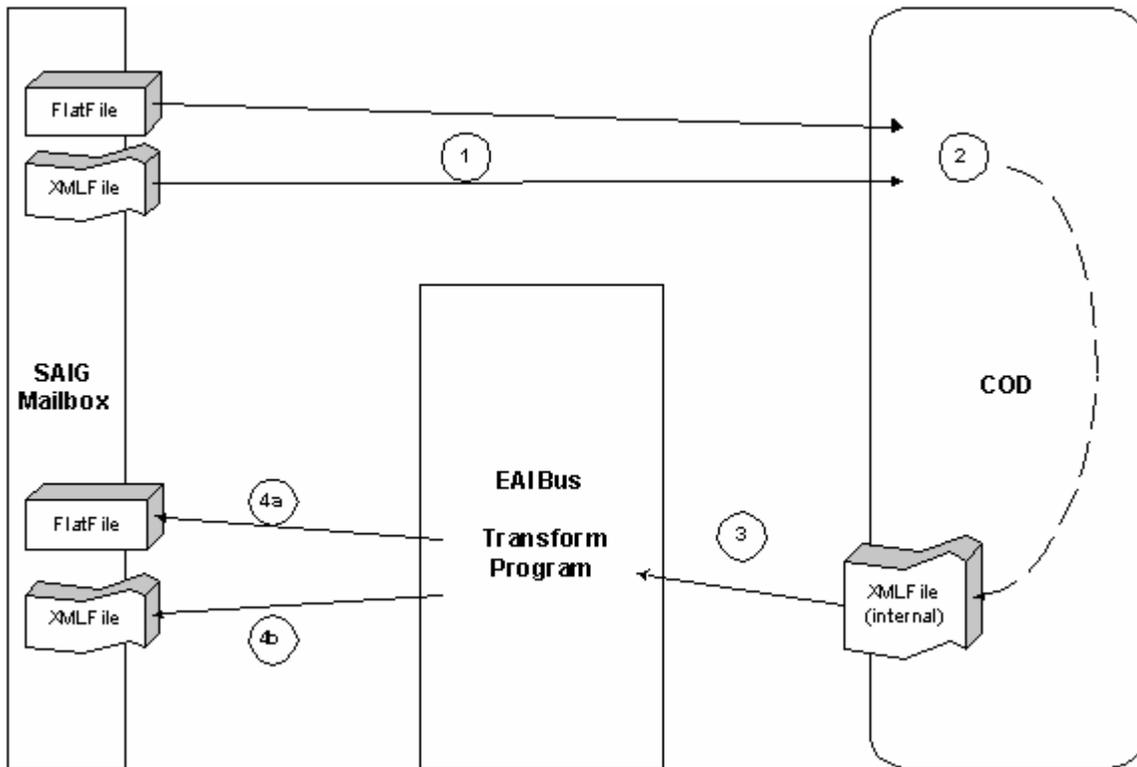
Date	Author	Version	Change Reference
9/16/2003	Scott Van Velsor	1.0	Initial Document Creation

Approval

Created By:	Scott Van Velsor	202.962.0771	Creation Date 9/16/2003
Approved By:	TBD	TBD	Approval Date: TBD
Tech Sign Off:	TBD	TBD	Sign Off Date: TBD

Overview

This section describes the entire COD file submission/response process to provide a general understanding of the COD transformation program and how it fits into the process. The following diagram illustrates the process at a high level:



Steps in the process:

1. The COD front-end system at TSYs accepts student loan Origination and Disbursement data from schools (via SAIG Mailbox) in two formats:
 - XML “Common Record” format from full participant schools
 - Legacy “Flat File” format from phase-in schools (schools that have not yet converted their systems to support the new XML format)

Both the XML and the Legacy file submissions are wrapped with SAIG headers and trailers, which are used in the SAIG mailbox system.

2. COD processes the data and sends out all of its responses in XML format. In other words, responses to both XML and Flat Files come back in XML. The XML file that it sends conforms to the internal version of the schema, which contains additional fields used by the COD transform program for constructing the final output (discussed in Step 3).

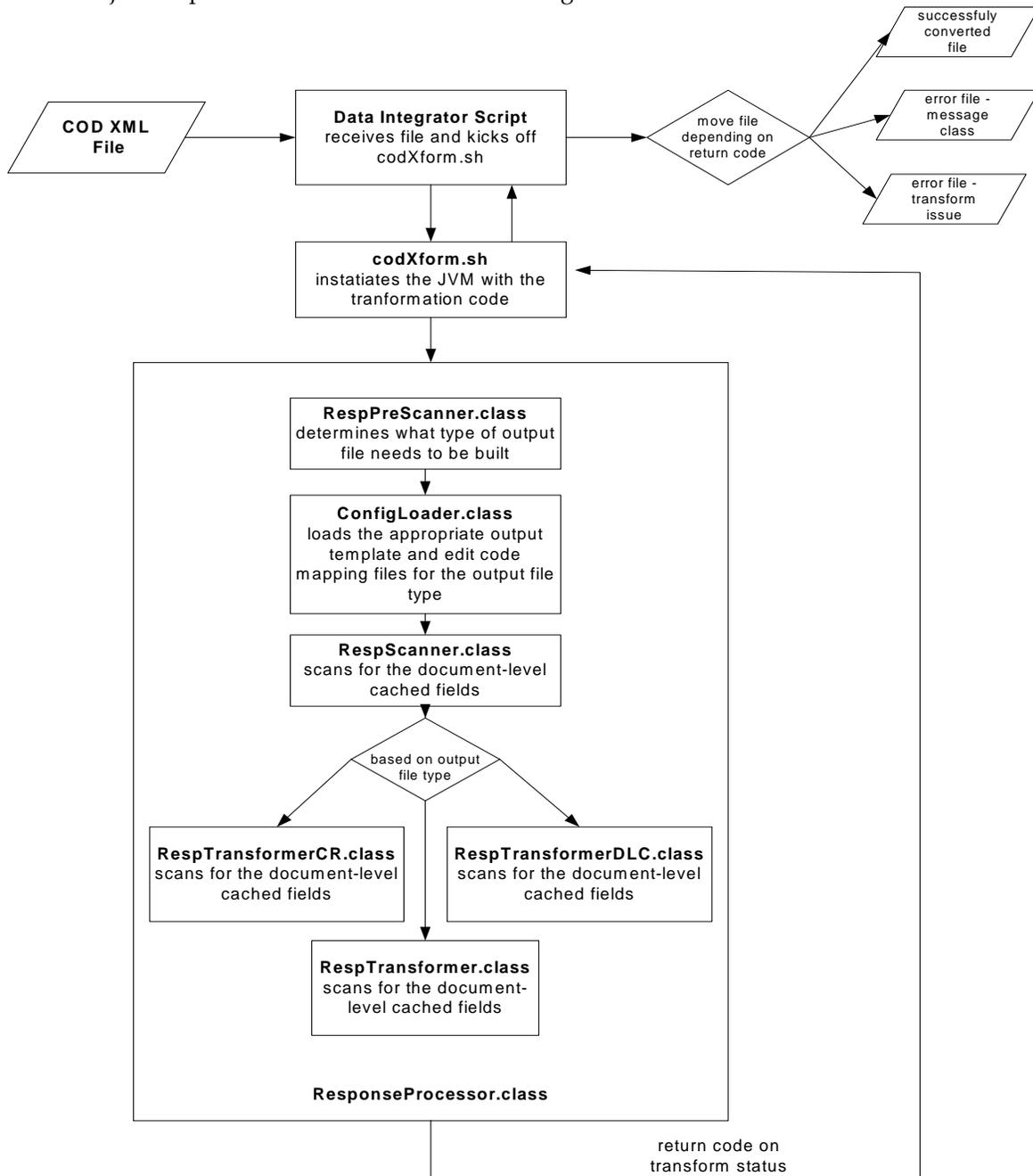
3. The COD transformation program processes the COD front-end acknowledgement output before it gets placed in the SAIG mailbox system and is returned to the schools. All COD front-end XML output goes through the EAI Bus before it is returned to schools via the SAIG mailbox system. The type of processing performed on the file depends on the type of response that is required.
4.
 - a) For XML submissions and responses from full-participant schools, the COD XML output is read, internal tags are stripped, and the SAIG headers and trailers are built around the XML
 - b) For Legacy submissions and acknowledgements from phase-in schools, the COD XML output is converted into a flat file acknowledgement format, the type of which is dependent on the message class

The COD transformation program is located on the EAI “Bus” servers. It is written in Java, and uses XML configuration files. In production, there is a mirror instance on each of the Bus machines. Each instance operates independently. Routing of traffic and load balancing is dependent on the Data Integrator (DI) messaging infrastructure, which actually moves the files from point to point.

Each COD transformation process is in fact a separate UNIX process on the bus server. That is, a separate JVM is launched on the box every time a transformation process occurs. This approach essentially eliminates restart and recovery type concerns – there is no “listener” process or anything similar that needs to be up and running for the transformation functionality to be available.

The Transform Process Flow

This section details the COD transformation process flow and the major components of the transformation module. The major components are detailed in the following sections.



Class Hierarchy

This section details the major COD transformation components and a high-level functional description of each component.

Component Name	Location	Description
transform.jar	\$EAIDIR/bin	All of the executable Java .class files are packaged together in this JAR file for a consolidated deployment of the executable code.
ConfigLoader.class	part of transform.jar	Responsible for processing the configuration files and creating the search-and-build data structures.
EmptyHandler.class	part of transform.jar	A RespSAXHandler-based class that doesn't process any events. Used as a placeholder when an XML file has been processed as much as necessary.
FFRowField.class	part of transform.jar	A data structure that models a field in a flat file row.
FFRowTemplate.class	part of transform.jar	A data structure that models a flat file row.
FFTemplateMgr.class	part of transform.jar	A singleton class that holds references to the flat file templates and the search-and-build data structures.
FormatUtils.class	part of transform.jar	A utility class that holds all of the formatting functions required to format XML data to the corresponding flat file values.
ResponseProcessor.class	part of transform.jar	The main class for the application, contains the main() function for the program entry point.
RespPreScanner.class	part of transform.jar	A RespSAXHandler-based class that handles the pre-scan, which is simply figuring out what type of output document needs to be produced (what template to load).
RespSAXHandler.class	part of transform.jar	The base class for classes that have to parse XML for this application.
RespScanner.class	part of transform.jar	A RespSAXHandler-based class that handles parsing the XML document, looking for cached fields and calculating count/aggregate fields.
RespTransformer.class	part of transform.jar	A RespSAXHandler-based class that handles conversion of COD XML output to all legacy responses (besides DL Change).
RespTransformerCR.class	part of transform.jar	A RespSAXHandler-based class that handles conversion of COD XML output to

		Common Records (with SAIG headers and trailers).
RespTransformerDLC.class	part of transform.jar	A RespSAXHandler-based class that handles conversion of COD XML output to legacy DL Change responses.
Transformer.class	part of transform.jar	An interface implemented by the RespSAXHandler (and the RespTransformer, RespTransformerCR, RespTransformerDLC, in turn), to allow for polymorphism across these classes on the transform method.

Configuration Files

This section describes in detail all configuration files used by the transformation program for all award types and award years. The legacy record layouts change from award year to award year, to accommodate for these changes, the configuration files that are award year dependent are suffixed with “_0x0x.xml” (e.g., _0203 for the 02-03 award year).

Component Name	Location	Description
x_ResponseTemplates.xml	\$EAIDIR/config/transform	A mapping of message class / batch type pairs to file types. This is the main file used when identifying what type of output file needs to be created.
x_LoggingConfig.xml	\$EAIDIR/config/transform	A file containing configuration information for the logging framework.
x_cr_response.xml	\$EAIDIR/config/transform	The mapping template file used for Common Record output files (Note: award year independent)
x_dl_batchEditMappings_0x0x.xml	\$EAIDIR/config/transform	The configuration file for COD-DL Legacy batch edit code mappings.
x_dl_booking_notif_0x0x.xml	\$EAIDIR/config/transform	The mapping template file used for DL Booking Notification legacy output files.
x_dl_change_ack_0x0x.xml	\$EAIDIR/config/transform	The mapping template file used for DL change file legacy output files.
x_dl_changeCodeMappings_0x0x.xml	\$EAIDIR/config/transform	The configuration file for COD-DL Legacy Sub/Unsub/PLUS change file code mappings.
x_dl_changeEditMappings_0x0x.xml	\$EAIDIR/config/transform	The configuration file for COD-DL Legacy Sub/Unsub/PLUS change file edit code mappings.
x_dl_creditdec_ack_0x0x.xml	\$EAIDIR/config/transform	The mapping template file used for DL Credit Decision legacy output files.
x_dl_plusOrig_ack_0x0x.xml	\$EAIDIR/config/transform	The mapping template file used for DL PLUS origination acknowledgement legacy output files.
x_dl_plusOrigEditMappings_0x0x.xml	\$EAIDIR/config/transform	The configuration file for COD-DL Legacy PLUS origination edit code mappings.
x_dl_pmttoserv_notif_0x0x.xml	\$EAIDIR/config/transform	The mapping template file used

1		for DL Payment to Servicing legacy output files.
x_dl_pnote_ack_0x0x.xml	\$EAIDIR/config/transform	The mapping template file used for DL Unsolicited Prom Note legacy output files.
x_dl_subunOrig_ack_0x0x.xml	\$EAIDIR/config/transform	The mapping template file used for DL Sub/Unsub origination acknowledgment legacy output files.
x_dl_subunOrigEditMappings_0x0x.xml	\$EAIDIR/config/transform	The configuration file for COD-DL Legacy Sub/Unsub origination edit code mappings.
x_dl_supDisb_ack_0x0x.xml	\$EAIDIR/config/transform	The mapping template file used for DL Sub/Unsub/PLUS disbursement acknowledgement legacy output files.
x_dl_supDisbEditMappings_0x0x.xml	\$EAIDIR/config/transform	The configuration file for COD-DL Legacy disbursement edit code mappings.
x_pell_batchEditMappings_0x0x.xml	\$EAIDIR/config/transform	The configuration file for COD-Pell Legacy batch edit code mappings.
x_pell_disb_ack_0x0x.xml	\$EAIDIR/config/transform	The mapping template file used for Pell disbursement acknowledgement legacy output files.
x_pell_disbEditMappings_0x0x.xml	\$EAIDIR/config/transform	The configuration file for COD-Pell Legacy disbursement edit code mappings.
x_pell_orig_ack_0x0x.xml	\$EAIDIR/config/transform	The mapping template file used for Pell origination acknowledgement legacy output files.
x_pell_origEditMappings_0x0x.xml	\$EAIDIR/config/transform	The configuration file for COD-Pell Legacy origination edit code mappings.

Log Files

Component Name	Location	Description
transformation.log	\$EAIDIR/logfiles	Status and error log entries for the transformation process are continually written to this log file as it executes.

Program Dependencies

This section lists the dependencies for the transformation program to function properly.

Dependencies	Location	Description
\$EAIDIR	Set for the mqm user.	The root directory of the installed COD EAI code. It is currently /export/home/mqm/eaicodr1
Sun JVM	N/A.	1.3.1_02
xerces.jar	\$EAIDIR/lib	This library provides support for XML processing via the SAX API. The Common Record XML file is processed with SAX. Current version is 1.4.4.
jdom.jar	\$EAIDIR/lib	This library provides support for XML processing via the JDOM API. The configuration files are processed with JDOM. Current version is b7.
protomatter-1.1.5.jar	\$EAIDIR/lib	This library provides support for application logging. Current version is 1.1.5.

Transformation directory structure

This section lists the directory structure that is required for the transformation application to operate.

Production Directory Structure	Dev/Test Directory Structure	Description
\$EAIDIR/config/transform	<home>/transform/config/transform	This is the repository for all transformation related configuration files for testing.
\$EAIDIR /logfiles	<home>/logfiles	The transformation log file is written to this directory
/export/data/mqm	<home>/xDataFiles	The input data files are moved to this directory for processing
/export/data/mqm/saig/errors	<home>/xErrorFiles	The files that resulted in a transformation error are moved to this directory
N/A	<home>/xPendingFiles	The files that resulted in an expect failure during transformation processing are moved to this directory
/export/data/mqm/saig/proc	<home>/xProcessedFiles	The input data files that are successfully transformed are moved to this directory.
/export/data/mqm/saig	<home>/xOutputFiles	The post-transform files that are successfully transformed are moved to this directory.

EAI - COD Transformation Interface

This section describes in detail the interface between the transformation program and the EAI infrastructure. The COD transformation program is located on the EAI “Bus” servers. The transformation program is instantiated through Data Integrator (DI) scripts on the arrival of COD school files. Data Integrator enables bulk file transfer between applications/systems using the MQSeries infrastructure. The DI script deployed on the EAI Bus to initiate the transformation program is detailed in the table below.

Dependencies	Location	Description
codXform.sh	\$EAIDIR/scripts	<p>This script takes the input parameters and calls the transformation java code. It uses the \$EAIDIR variable to determine the location of the executable .jar files. The input parameters to this script are detailed in order below:</p> <ol style="list-style-type: none"> 1. The full path to the input file 2. The full path to the output file 3. The full path to the pending file directory 4. The full path to the processed file directory 5. The full path to the error file directory

Error Handling

The COD transform returns a status code that indicates the process status to the script that called it. Depending on what happened, it will return one of three codes:

- Status code 0: normal, successful processing. The complete transformed file has been created, with the name and location specified in the initial arguments to the script (see description of codXform.sh above, under “Scripts”). The original input (COD XML output) file is copied to the processed file directory as specified in the arguments to codXform.sh.
- Status code 100: the process was in error. The message class / batch type of the COD output file did not correspond to any known combinations; therefore the transformed file type to create could not be determined. The original input (COD XML output) file has the error message appended to it, and is copied to the error file directory as specified in the arguments to codXform.sh (see description of codXform.sh above, under “Scripts”).
- Status code 200: the process was in error. This error condition is anything but the one described for the 100 code. Examples of 200 error code conditions are:
 - Data was too long/short for field
 - Data was missing for a required field
 - XML was malformed and could not be parsed

The original input (COD XML output) file has the error message appended to it, and is copied to the error file directory as specified in the arguments to codXform.sh (see description of codXform.sh above, under “Scripts”).

Currently, the Data Integrator script that calls the COD transformation program will then route files depending on the status code:

- Status code 0: route the transformed output file to the SAIG mailboxes for delivery to schools
- Status code 100: route the COD XML (pre-transformed) output file with error messages to the “Bad message class” GDG <CPQ2396.INH.GP00.ERTOBUS(+1)> at COD for correction
- Status code 200: route the COD XML (pre-transformed) output file with error messages to the “All other transform errors” GDG <CPQ2396.GP00.BUS.TRANSERR(+1)> at COD for correction

Guidelines for new Transformation Development

This section describes the general guidelines for common COD transformation development to accommodate potential changes in legacy record layouts, common record schemas, and mapping rules from award year to award year. The following steps should be followed for common (e.g., future award years) COD transformation development efforts.

1. Review the EAI mapping documents for the message class and batch type that requires a change.
2. Review the new mapping rules document provided by the COD functional team and update the EAI mapping document to reflect the required change for the particular award year. Any discrepancies in the record layout between the EAI mapping document, COD Technical Reference, or COD functional mapping document would require some research. However, if the issue is still unresolved, it is EAI team's responsibility to track down this information. This is a suggested approach:
 - a. Send an email to the COD/transformation development lead with the record type, AY, program, and a brief description of the issue. The development lead will serve as the point-of-contact for all COD mapping related issues.
 - b. Development lead will update the COD R 2.x mappings Issues Log (X:\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.x\XFORM) with the new issue and send an email to the COD functional team with the issue.
 - c. Development lead will distribute the findings to the entire team after the issue is resolved.
3. Create/update the configuration files based on the updated EAI mapping documents
4. Perform all unit tests on the local development environment described in Appendix A - Development Environment.

Appendix A - Development Environment

This section describes the high-level process to create the COD transformation development environment local to the developer's workstation. The steps are detailed below:

- 1) Download and install the Java Development Kit (JDK) most current version 1.3.1_02 from the SUN website (sun.java.com)
- 2) Download and install Textpad, this will be a very useful tool for creating test data and verifying test results.
- 3) Unzip the XForm zip file that includes:
 - a. Java source code
 - b. Libraries
 - c. Configuration files
 - d. Transformation program directory structure
 - e. Batch files to run the transformation program
- 4) Modify the following batch files to setup the environment variables and classpath:
 - a. sethome.bat - This batch file sets up your home directory for your transformation program (e.g., C:\COD\Development\transform)
 - b. setcp.bat - This batch file sets up your class path for the transformation program
- 5) Send an email to the EAI Configuration Lead to obtain the latest version of the following:
 - a. Transformation configuration files - move these configuration files to \config\transform under the designated home directory.
 - b. Transformation source code - move the source (.java) files to \gov\ed\sfa\cod\transform under the home directory specified in the sethome.bat batch files.
 - c. transform.jar - move the jar file to directly under the home directory
- 6) Use FTP to move the files to their local development environment (laptop/desktop)
- 7) The developers will create/modify the XML configuration files on their local development environment (laptop/desktop). It is the developer's responsibility to backup versions of the configuration files during development. The 'work-in-progress' configuration files will be kept on the LAN during development. However, significant changes to the configuration files can be checked into Clear Case for version control.

Appendix B - Test Environment

This section describes the high-level process to create the COD transformation development/test environment on the Dev EAI Bus server SU35e16 (4.20.15.136) or other Unix server. Developers will have their own test environment setup under their home directory. All tests performed on the test environment will be against the same transformation source code. However, each developer has their own set of configuration files for testing new COD transformation developments to prevent overwriting shared configuration files. The steps are detailed below:

- 1) Send an email to the EAI Configuration Lead to obtain the latest version of the COD Xform Test environment setup files listed in the table below (Refer to Appendix E for detail instructions on how to use these utility scripts). These files should be transferred via FTP to your home directory (e.g., /export/home/userID). Execute *the xform_test_env_setup.ksh* script to setup the test environment.

Script Name	Description
codXform.ksh	This script takes the input parameters and calls the transformation java code. It uses the \$EAIDIR variable to determine the location of the executable .jar files.
codxformB.ksh	This script calls codXform.ksh and performs transformation on each file in the <INPUTDIR> directory.
changetext.ksh	This script performs a find and replace of sub-text within files in a specified directory. The find and replace text are specified in the changetext.ex.cmd command file.
changetext.ex.cmd	This is a command file for the changetext.ksh script. Make modification to this script to find a specified text and replacement text in files. For example, %s/<Original Text>/<Replacement Text>/g
renamefiles.ksh	This script renames a file in the specified directory with a '.xf' suffix.
xform_test_env_setup.ksh	This script creates the COD transformation dev/test environment for the developer on the EAI Bus.
chk_diff.ksh	This script takes two input parameters <DIR1> and <DIR2> and performs a diff between the files in the two directories. This script produce a file with the differences between the files in the different directories

- 2) Send an email to the EAI Configuration Lead to obtain the latest version of the following:
 - a. Transformation configuration files – move these configuration files to \config\transform under the designated home directory.
- 3) All unit test conditions must be met before migrating to the test environment on the BUS server (e16) for Regression Testing.

If the test condition fails, change the configuration file or code as necessary in the development environment and repeat the process until all test conditions are met.

Appendix C - COD Transformation References

This section details the list of EAI and COD documents created through the development and test phase. These are living documents and should be updated to reflect any requirements, development changes, or testing results. The deliverable/document name, a high level description, and sample deliverables are detailed in the table below.

Deliverables/Documents	Description	Location/Sample documents
Mapping Document	<p>The purpose of this document is to capture the data mapping between the internal COD Common Record schema and the legacy record layouts. This document combines the three references into a single view:</p> <ul style="list-style-type: none"> • COD Technical Reference document (PDF) • COD Functional Mapping Document (Excel) • Transformation configuration file (XML) 	\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Legacy Record\PL\26.06 - Pell Legacy Origination\AY 02-03\26.06 Pell Legacy Origination 0203 Mapping Document v01.xls
Unit Test Conditions	<p>The purpose of this document is to list all the test conditions for a specific message class and batch type. These test conditions are directly linked to the requirements and includes three type of processing:</p> <ul style="list-style-type: none"> • Normal • Expected • Unexpected 	\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Legacy Record\PL\26.06 - Pell Legacy Origination\AY 02-03\26.06 Pell Legacy Origination 0203 Unit Test Conditions v01.xls
Unit Test Scripts	<p>The purpose of this document is to detail the steps needed to execute the specific unit test conditions.</p>	\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Legacy Record\PL\26.06 - Pell Legacy Origination\AY 02-03\26.06 Pell Legacy Origination 0203 Unit Test Scripts v01.xls
Gap Analysis of COD Pell Edit Mapping document	<p>The Pell edit code gap analysis document captures the following information for all award years:</p>	\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Edits Documents\R2.2 Edit Issues.xls

	<ul style="list-style-type: none"> • Documents legacy edit codes that have a corresponding COD edit and are translated by the EAI Bus. This reflects the configuration files that are currently in production. • Documents legacy edit codes that doesn't have a corresponding COD edit and are not translated by the EAI Bus. • Documents discrepancies between the functional edit mapping document (provided by the COD functional team) and what is currently in production (e.g., new edit, removed edit, etc). 	
<p>Gap Analysis of COD Direct Loan Edit Mapping document</p>	<p>The Direct Loan edit code gap analysis document captures the following information for all award years:</p> <ul style="list-style-type: none"> • Documents legacy edit codes that have a corresponding COD edit and are translated by the EAI Bus. This reflects the configuration files that are currently in production. • Documents legacy edit codes that doesn't have a corresponding COD edit and are not translated by the EAI Bus. • Documents 	<p>\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Edits Documents\ R2.2 Edit Issues.xls</p>

	discrepancies between the functional edit mapping document (provided by the COD functional team) and what is currently in production (e.g., new edit, removed edit, etc).	
COD Mapping Rules documents	The purpose of this document is NOT to provide technical detail but to assist in the articulation of the legacy elements to the common record XML tags as specified by FSA and other partners during the requirement-gathering phase.	\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Functional Team Mapping Docs\2003-04DirectLoanMappingRuleswith2[1].0Schema12-06-02ver1.0.xls
COD Bus Edit Business Rules	This document is written from the perspective of translating from a Common Record edit to a legacy edit. The purpose of this document is not to provide technical detail and solutions but to assist in the translation of edits from a functional perspective.	\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Edits Documents\Updated Edit Mapping Appendix from FDD (CHANGES TRACKED) EAI VERSION.doc
COD Transformation Test Plan	This document defines the EAI Testing Plan for the Common Origination & Disbursement (COD) Transformation program. This document will capture all testing phases and ensure that a sound framework for testing has been established.	\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Test Plan\COD_Transformation_Test_Plan_R2.2_v1.0.doc

Appendix D - COD Transformation Deliverable Location:

This section lists the location of all transformation deliverables for Release 2.0 in the table below.

Directory/Path	Description
\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Common Record	Common Record transformation deliverables for 02-03 and 03-04 are stored under this directory.
\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Legacy Record\DL	Direct Loan legacy transformation deliverables for 02-03 and 03-04 are stored under this parent directory. This includes edit codes and message/record mappings.
\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Legacy Record\PL	Pell legacy transformation deliverables for 02-03 and 03-04 are stored under this parent directory. This includes edit codes and message/record mappings.
\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\COD Common Record XML Schema	COD Common Record schemas for schools are stored under this directory.
\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Functional Team Mapping Docs	COD Mapping Rules document for all legacy files are stored under this directory.
\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Functional Team Mapping Docs\Mapping Document Discrepancies	COD Mapping document discrepancies for all legacy files are stored under this directory
\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Config Files	Latest versions of COD transformation configuration files during the development phase are stored under this directory
\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Templates	This directory contains various templates to aid the development and testing of the COD transformation module. This includes: <ul style="list-style-type: none"> • Direct Loan Header and Trailer Test Data & Expected Results • Direct Loan TVWIN Header and Trailer Test Data & Expected Results • EAI Transformation Mapping Documents • Test Conditions and Scripts
\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Test Plan	Test plan for COD R 2.2 Transformation development
\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.0\XFORM\Environments	This directory contains the following documents: <ul style="list-style-type: none"> • COD-EAI environment configuration • COD transformation migration checklist • Other environment related documents
\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\Utilities	This directory contains various utilities used to support the development and testing of R 2.2 COD Transformation enhancements

\\Acdoe-dc1\CIO\TechArch\EAI Core Release 3\Interface Partners\COD R2.2\XFORM\R2.2 Regression Testing	This directory contains the baseline test conditions for regression testing of R 2.2 COD Transformation enhancements.
---	---

Every COD transformation development of a different message class and batch type follows the same deliverable package. The AY02-03 Direct Loan legacy record format is used as a example to walk through the directory structure and deliverables required for future developments on the different message classes and batch types.

Directory Structure	Description
...\DL\DL Legacy Origination	This is the root directory for the particular message class and batch type. In this example, Direct Loan Origination.
...\DL\DL Legacy Origination \AY 02-03	This is the root directory for the particular message class and batch type for that specific award year. In this example, 02-03
...\DL\DL Legacy Origination \AY 02-03\x_dl_subunOrig_ack_0203.xml	The configuration file used for development and testing is stored under this directory.
...\DL\DL Legacy Origination \AY 02-03\DL Legacy Sub Unsub Origination Acknowledgement 0203 Unit Test Conditions.xls	The purpose of this document is to list all the test conditions for a specific message class and batch type.
...\DL\DL Legacy Origination \AY 02-03\DL Legacy Sub Unsub Origination Acknowledgement 0203 Unit Test Scripts	The purpose of this document is to detail the steps needed to execute the specific unit test conditions.
...\DL\DL Legacy Origination\AY02-03\26.02.02-03.Template.xml	This is a data template with all the data fields populated. This template will be used to create all test data for this particular message type.
...\DL\DL Legacy Origination\AY 02-03\Test Date\Input*.xml	This is the repository for all the test data created for all the test conditions
...\DL\DL Legacy Origination\AY 02-03\Test Date\Output*.xf	This is the repository for the actual results from the initial execution. If the actual results match with the expected results, it can be archived as expected results for regression testing.

Appendix E - COD Test Utility Scripts

The following is a listing of the test utilities developed by the EAI team for the purposes of COD Transformation testing.

Check Differences Utility

This script takes two input parameters <DIR1> and <DIR2> and performs a diff between the files in the two directories. This script will produce a text file that lists the differences between any files that exists in each directory with the same name. Identical files will result in a “No Differences Encountered” message. Before running this script the change text script should be run on both directories that are to be compared, and any previously existing diff_results.txt files should be erased.

Usage: `chk_diff.ksh <DIR1> <DIR2>`

Parameters: `<DIR1>` - Relative path of a list of files (e.g. actual test results)
`<DIR2>` - Relative path of a list of files (e.g. expected test results)

Expected Results: `diff_result.txt`
This output file lists the differences in the files contained in the two directories.
This output file will be created in the directory `<home directory>/transform`

Script Location: `<home directory>/transform`

Change Text Utility

This script will apply the changes detailed in `changetext.ex.cmd` for all files in the directory `<DIR1>`. This script will perform a find-and-replace on each of the files in the directory `<DIR1>`, the find and replace is defined by the contents of the `changetext.ex.cmd` file.

Usage: `changetext.ksh <DIR1>`

Parameters: `<DIR1>` - Relative path of the directory containing the files on which the find-and-replace operation should take place.

Expected Results: Each of the files within the directory `<DIR1>` will have undergone a text find-and-replace operation as defined by the `changetext.ex.cmd` file.

Script Location: `<home directory>/transform`

Changetext.ex.cmd The `changetext.ksh` script enacts the find-and-replace operation that is defined within the `changetext.ex.cmd` file. This file may contain any number of find and replace commands, one on each line, and terminates with a line containing “`wq`” to save the changes to the files. Each command in the file should be structured as follows:
`%s/<Original Text>/<Replacement Text>/g`

Before running the Check Differences Utility the Change File Utility should always be run on both sets of data files with the following command lines in the `changetext.ex.cmd` file:

```
%s/[0123456789]*/#####/g  
wq
```

This command will replace the unique “inproc” number with a generic “#####” string to avoid unnecessary results from the Check Differences Utility.

Batch Transform Utility

This script takes one input parameter `<DIR1>` and performs a transformation operation on each file in the `<DIR1>` directory. Before running this script any previously existing files should be deleted from the `xOutFiles`, `xErrorFiles`, and the `xProcessedFiles` directories. The script uses the `$EAIDIR` variable to determine the location of the executable `.jar` files.

Usage: `codXformB.ksh <DIR1>`

Parameters: `<DIR1>` - Relative path of a list of files to be transformed.

Expected Results: The files in the directory `<DIR1>` (normally the ‘`xDataFiles`’ directory) will be transformed by the transformation application and deleted. The results of the transformation will be in the `xOutFiles` and/or the `xErrorFiles` directories.

Script Location: `<home directory>/transform`

Transform Script

This script takes one input parameter `< pre-xform data filename >` and performs a transformation operation on each file in the `< pre-xform data filename >` directory. Before running this script any previously existing files should be deleted from the `xOutFiles`, `xErrorFiles` and the `xProcessedFiles` directories. The script uses the `$EAIDIR` variable to determine the location of the executable `.jar` files.

Usage: `codXformB.ksh < pre-xform data filename >`

Parameters: `< pre-xform data filename >` - File name of the pre-transform XML file in the `xDataFiles` directory.

Expected Results: The file `< pre-xform data filename >` in the `xDataFiles` directory will be transformed by the transformation application and deleted. The results of the transformation will be in the `xOutFiles` or the `xErrorFiles` directories.

Script Location: `<home directory>/transform`

Appendix F – Contacts

Name	Contact Information	Team	Responsibilities
Debbie Miller	NCS Pearson Debbie_Miller@ncs.com	COD	Owner of the following documents: <ul style="list-style-type: none"> • COD Mapping Rules documents for Pell and Direct Loans • COD Bus Edit Business Rules for Pell and Direct Loans
Bryan Van Note	NCS Pearson Bryan_Van_Note@ncs.com	COD	Owner of the following documents: <ul style="list-style-type: none"> • COD Mapping Rules documents for Pell and Direct Loans • COD Bus Edit Business Rules for Pell and Direct Loans
Andrew Smalera	andrew.smalera@accenture.com	ITA	COD Transformation architect/ developer <ul style="list-style-type: none"> • Developed the transformation program. • Deep functional knowledge of COD
Theresa Pak	theresa.pak@accenture.com	EAI	COD Transformation developer/ tester <ul style="list-style-type: none"> • Understands the overall transformation development and testing process • Working knowledge of transformation code • Good understanding of configuration files
Julie T. McNeil	julie.t.mcneil@accenture.com	COD	COD Transformation developer/ tester <ul style="list-style-type: none"> • Understands the overall transformation development and testing process • Working knowledge of transformation code • Good understanding of configuration files
Lori J. Clemmensen	Lori.j.clemmensen@accenture.com	COD	Good working knowledge of the following documents: <ul style="list-style-type: none"> • COD Mapping Rules documents for Pell and Direct Loans • COD Bus Edit Business Rules for Pell and Direct Loans • COD Technical Reference – one of the authors
Scott Van Velsor	fscott.van.velsor.jr@accenture.com	EAI	EAI Configuration Management Lead <ul style="list-style-type: none"> • Check-in/ check out code/ scripts • Deploy new EAI builds

Appendix F – Lessons Learned

This section provides the transformation developers/testers with a list of “gotchas” from previous developers. The best practices and general guidelines for transformation maintenance and enhancements are documented in this section for future reference and further development. This section will continue to expand as more requirements are implemented. It is each developer’s responsibility to keep this list up-to-date with the current findings.

Problems/Issues	Potential Problem
The award block is missing in the post transform legacy flat file	<ul style="list-style-type: none"> • Misspelling of the Award block common record tag (.e.g. instead of DLPLUS, it is misspelled DPLUS) • Invalid configuration file is specified in the ResponseTemplate.xml for the message type and batch type.
<pre><!-- <ERRORTXT> null </ERRORTXT></pre>	This error text is written to the post transform response file when an invalid batch or specific edits configuration file is specified in the data mapping configuration files. Verify that the edits configuration exists and that it is spelled correctly.
Edit codes not translated in post transform response.	<ul style="list-style-type: none"> • Misspelling of the edits block <EditProcessResult> • Edit code returned in a different block than specified in the edits configuration file • The edit code translation is not specified in the configuration file
Legacy record data field not translated in pos transform response	<ul style="list-style-type: none"> • Misspelling of the tag names • Data field returned in a different block than specified in the configuration file
Direct Loan Change edit code not translated in post transform response	<p>The DL change edit codes are translated only when the following requirements are met:</p> <ul style="list-style-type: none"> • Direct Loan Change tag in the DL change acknowledge record • Edit code associated with the DL change tag • Direct Loan change tag within the edits block <p>Example:</p> <pre>.. .. <Student SSN="500320442" BirthDate="1962-03-20" LastName="COPPER"> <Identifiers> <SSN>123456789</SSN> </Identifiers> <DLPLUS> <FinancialAwardID>500320442S03G02868001</FinancialAwardID> </DLPLUS> <Response> <ResponseCode>R</ResponseCode> <EditProcessResult> <ResponseErrorCode>016</ResponseErrorCode> <ResponseErrorField>SSN</ResponseErrorField> </EditProcessResult> </Response></pre> <p>...</p>

Appendix G - EAI Mapping Document Guide

During Unit Testing of release 2.0 of the COD Transformation code, mapping documents were created for each transformed message class. The purpose of the mapping documents is to provide a clear representation of how the data contained in the legacy flat files is related to the data contained in the COD XML output files. The documents contain related legacy and XML file information side by side for each field in the legacy flat file.

Legacy File Layouts											Common Record Layout					
Legacy Block	Start	End	Length	Type	Field Name	Field Description	Valid Values	Justify	Pad	No Value Pad	Type	Record Block	Data Source	Cache Data Source	Translation Rules	Formatting Rules

The legacy portion of the document contains the following information:

- Legacy Block
 - Indicates which row of data in the Legacy flat file contains this particular field. This value may be either the SAIG Header or trailer, the Batch Header or trailer, or one of the main data rows.
- Start
 - Indicates the first character number of the flat file field.
- End
 - Indicates the last character number of the flat file field.
- Length
 - Indicates the total number of characters in the legacy flat file that are reserved for the use of this field.
- Type
 - Indicates the type of data in the field, 'A' for alpha, 'N' for numeric, 'A/N' for alphanumeric or 'N/A' for not applicable.
- Field Name
 - The name of the particular field in the flat file layout.
- Field Description
 - When applicable, a brief description of the field and /or any additional comments about its use.
- Valid Values
 - As accurate of a description as possible of the values that are valid for the legacy flat file layout. The description may be a single value for static fields or a set of rules that can be applied to a generated value.
- Justify
 - Describes the justification of the legacy flat file field, may be 'none' (the transformation defaults to left), 'left', or 'right'. This parameter is used by the Transformation code to place values into fields when the value is shorter than the field length.
- Pad
 - Defines how the Transformation will pad the extra characters in a legacy flat file field when the value from the common record id shorter than the field. Valid values include 'spaces' (usually used for alpha or alphanumeric fields), 'zeros' (usually used for numerical fields), 'none' (tell the transformation code to fail with an error when a this field is shorter than the required length), or the pad value may be a single character, i.e. placing a value of 'z' in the Pad field will pad the value of the field with the appropriate number of 'z' characters.
- No Value Pad

- Defines how the Transformation will pad the characters of the legacy flat file field when no value was defined in the XML common record. This pad value is only used when the corresponding XML tag did NOT appear in the XML common record. If the tag did appear in the XML common record (even with no value inside the tag) then the Transformation code will pad using the Pad parameter and ignore the No Value Pad parameter.

The common record portion of the document contains the following information:

- Type
 - Can be one of several values,
 - 'static' - meaning that the value to be placed into the legacy flat file field is hard coded in the configuration file and is independent of the XML produced by COD.
 - 'ref' - meaning that the value to be placed into the legacy flat file field should have been cached as one of the cached fields by the transformation code. These values are often used in several places in the legacy flat file, or are used in headers or trailers.
 - 'node' - meaning that the value to be placed into the legacy flat file field will be read from the XML common record when it is reached. These values are often used only once per data line in the legacy flat file and their values often differ from line to line.
 - 'batch edit' or 'specific edit' - meaning that the configuration file contains the value 'EDITCODE' and that the value to be placed into the legacy flat file is a translation of a values contained in the XML common record field. The translation rules are contained in separate batch or specific edit configuration files that are distinct from the mail configuration file for the message class.
- Record Block
 - Indicates the general block within the XML common record that contains the tag from which the XML common record value is read. This values should be blank for and fields with a type of 'static'
- Data Source
 - For fields of Type 'static' - this value is the literal value (or exact description of the value) that will be placed in the legacy flat file.
 - For fields of Types 'node' or 'ref' - indicated the name of the XML tag that contains the value of the field. Any block names or tag names are relative to the general path given in the Record Block column.
 - For fields of Type 'static', 'batch edit' or 'specific edit' - This column is N/A
- Cache Data Source
 - For fields of Type 'ref' - refers to the internal Transformation name of the cached value for a particular field. These values have already been read from the XML common record during a previous scan and have been saved by the Transformation code.
 - For all other field Types - This column is N/A
- Translation Rules
 - Where applicable this field contains specific details of how a particular field is translated from its XML common record format to its legacy flat file format.
- Formatting Rules
 - Where applicable this field indicated the formatting code contained in the configuration file. Each formatting code refers to a specific formatting function that will perform certain formatting operations, i.e. removing of dashes from XML common record date value.