Smarter Balanced Reporting (RFP 15) Developer Guide

Prepared for:

by:

Approvals

Representing	Date	Author	Status
Consortium		Joe Willhoft	
Consortium	2014.09.24	Brandt Redd	Approved for Milestone 5
PMP	2014.09.24	Kevin King	Endorsed for Milestone 5
Workgroup	2014.09.23	Henry King	Endorsed for Milestone 5

Revision History

Revision Description	Author/Modifier	Date
Initial Release (DRAFT)	Anna Grebneva (Amplify)	2014.07.09

Table of Contents

1 Summary

Smarter Balanced Data Warehouse and Reporting system supports ingestion of student assessment and student registration data, authentication with a SAML identity provider, and visual display different representation of the ingested data through a web application in HTML, CSV and PDF data formats.
The implementation is substantially in Python

1.1 Audience

This document is designed for developers that are familiar and have experience with:

- client-server web applications
- REST requests and responses in JSON data format
- authentication with SAML

1.2 Assumptions

This document is tailored towards Mac OS development environment. Many of our 3rd party dependencies can be installed via brew on a Mac OS environment.

2 Setting up Development Environment

1. Install Python 3.3.0
   You can download the installer from https://www.python.org/download/releases/3.3.0

   If an earlier version of Python is already installed (2.6 or 2.7), you may also be able to install Python 3.3 via “brew” (for Mac), “yum” (for Linux/CentOS), or “apt-get” (for ubuntu/debian).

   If installing from source, you must use “make alt-install”

   After installation, please verify that the version is correct by running this command

   ```
   python3 --version
   ```

   Make sure that pip and virtualenv are installed with python. You can verify this by checking that they exist in the bin directory of your python installation.

   If they don’t exist, you can download and install them manually. virtualenv is available at https://pypi.python.org/pypi/virtualenv/1.11 and pip is available at http://pypi.python.org/pypi/pip

2. Create and activate virtualenv

   Python 3.3 doesn't usually have virtualenv included in it, because it has a built-in version called pyenv

Important Note: if your machine has both Python 2.6/2.7 & Python 3.x , it is critical that you don't use pip or ez_setup to install virtualenv because they will use the default System Python in the process and thus implement a binding to the Python 2.x interpreter. Instead, curl or wget the source package from PyPa for virtualenv and run the virtualenv.py with the same syntax statement you have below, but using full local paths, not relative paths to both the python and the destdir

```
virtualenv -p <python-exe-path> --distribute <dest-dir>

. <dest-dir>/bin/activate
```

You can verify pip and python is installed properly inside your virtual environment by running these two commands:

```
(virtualenv) python --version

(virtualenv) pip freeze
```

3. Install Node.js and coffeescript (Only required for web application)

   You can download and install node.js from http://nodejs.org/download/.
   Make sure that `npm` command works after the installation.

   You can now install coffeescript:
   ```
   npm install -g coffee-script
   ```

4. Install xmlsec1 for SAML security verification (Only required for web application)

   On a Mac, you can install from brew:

   ```
   brew upgrade pkg-config

   brew install xmlsec1
   ```

5. Install wkhtmltopdf and poppler for pdf generation (Only required for web application)

   You can download and install from here: http://wkhtmltopdf.org/downloads.html

   On a mac, you can install from brew:
   ```
   brew install wkhtmltopdf
   ```

```
brew install poppler
```

6. Install gpg

   You can download and install gpg1.4 from here: https://www.gnupg.org/download/

   On a Mac, you can install from brew:
   ```
   brew install gnupg
   ```

7. Install PostgreSQL

   You can download PostgreSQL 9.2 from here: http://www.postgresql.org/download/

   You can also download pgadmin, a graphical management tool for PostgresSQL, from here: http://www.pgadmin.org/

8. Install RabbitMQ

   You can download and install from here: http://www.erlang.org/download.html

   On a Mac, you can install from brew:
   ```
   brew install rabbitmq
   ```

9. Install Aptana IDE (optional)

   You can download Aptana from here: http://www.aptana.com/products/studio3/download

   You'll need to set up your Python Interpreter in Aptana. Navigate to Preference → PyDev → Interpreter - Python → New …

   Enter *edware* as the interpreter name and set the Interpreter Executable to the path of your python3 executable within your virtual environment, *<dest-dir>/bin/python3*

   Python Pyramid is automatically installed when you run python setup-developer.py in the smarter directory.

   To configure to run Python Pyramid based web applications, Click on String Substitution Variables → Add Variable

   Enter name as *pyramid_run* and value of *<dest-dir>/bin/pserve*

Apply and Click on OK to save your changes.

You'll also need to add a new Run Configuration to run Pyramid applications.

Navigate to Run → Run Configurations. Add a new configuration called smarter. In Main Module, enter *${pyramid_run}* and Click on Arguments tab and add the following to the program arguments, *../config/development.ini*

3 Running Smarter Web Application

Smarter is a web application written in Python and Coffeescript. It is written using Pyramid web framework and can be run locally via pserve. The application requires an INI configuration file to run. Smarter needs to authenticate with an Identity Provider, such a OpenAM, using SAML 2.0. Client side source code is written in Coffeescript, which requires to be pre-compiled to Javascript. We use node.js libraries to pre-compile, minify, and watch for coffeescript and less file changes in the <repo>/smarter/assets directory.

3.1 Smarter Functionalities

- Web-based Reports
  - Comparing Populations at State Level, District Level, School Level
  - List of Students Report
  - Individual Student Report
- Print-Friendly Formatted Reports
  - PDF version of Individual Student Report
  - 508 Compliant csv Extract of Comparing Populations Report
  - Student Assessment Extracts in JSON and csv
  - Student Registration Statistics and Completeness csv Report
- Pre-Generation
  - PDF pre-generation
  - Cache warmer

3.2 Generating INI Configuration File

Smarter reads an INI file when the application starts up. A basic version of this INI configuration can be generated by the following commands, which builds an INI file based on values defined in settings.yaml. First, you should install the required python dependencies.

```
(virtualenv) cd <repo>/config
(virtualenv) python setup.py develop
```

Next, generate the ini file,

```
(virtualenv) python generate_ini.py -e development
```

An INI file, development.ini, will be generated in the config directory. You can edit this file directly if you need to make any configuration changes. Alternatively, you can make permanent changes inside settings.yaml.

3.3 Configurations in INI file

Below is a list of commonly configured settings that should be modified depending on your environment

Configuration	Description
auth.saml.idp_server_login_url	This is the SSO login URL that Smarter will redirect to when user isn't authenticated. This URL should be an URL provided by your SAML Identity Provider.
auth.saml.idp_server_logout_url	This is the logout URL that Smarter will redirect to when an authenticated user wants to log out.
auth.saml.issuer_name	This is your Service Provider identification name used in SAML Requests.
auth.saml.name_qualifier	This is your Identity Provider identification name used in SAML Requests.
edware.db.*[tenant_name].*url	This is the URL to your database server. The format of this URL is: *postgresql+psycopg2://user:password@hostname:port/databaseName* The *tenant_name* can be any string, but this must match the tenant_name that is provided by SSO.
edware.db.*[tenant_name]*.schema_name	This is the name of the schema that hosts the data.
edware.db.*[tenant_name]*.state_code	This is the state code that the tenant is hosting. ex. CA
edware_stats.db.schema_name	The schema name of the stats database server. (This database is used to record statistics of batch jobs that were migrated and is used in Smarter to trigger pre-generation of pdfs and pre-caching)

edware_stats.db.url	The database url of the stats database server. The format of this URL is: *postgresql+psycopg2://user:password@hostname:port/databaseName*

3.4 Creating Edware Schema

You'll need to create a schema to host the data that Smarter will query from. First, you need to install python dependencies.

```
(virtualenv) cd <repo>/edschema
(virtualenv) python setup.py develop
```

Next, create an empty schema into your database.

```
(virtualenv) cd <repo>/edschema/edschema
(virtualenv) python metadata_generator.py -s <schemaName> -m edware -d
<databaseName> --host <hostname:port> -u <user> -p <password>
```

You can now import our test data into your database.

```
(virtualenv) cd <repo>/test_utils
(virtualenv) python import_data.py -c <repo>/config/development.ini -i <tenant> -s
<stateCode> -n <stateName>

ex. python import_data.py -c <repo>/config/development.ini -i cat -s NC - "North Carolina"
```

Note: import_data.py reads the database configuration for your tenant from the INI file that you've generated previously

3.5 Starting Smarter

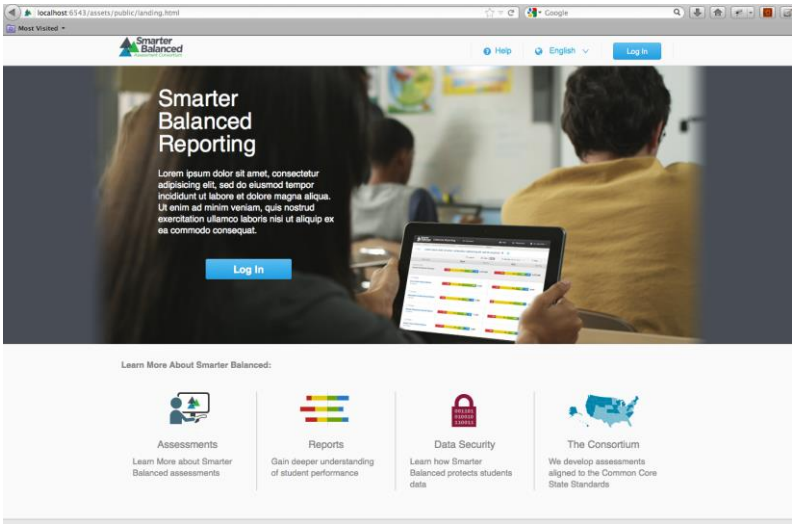
You'll first need to install all python dependencies that Smarter uses.

```
(virtualenv) cd <repo>/smarter
(virtualenv) python setup-developer.py develop
```

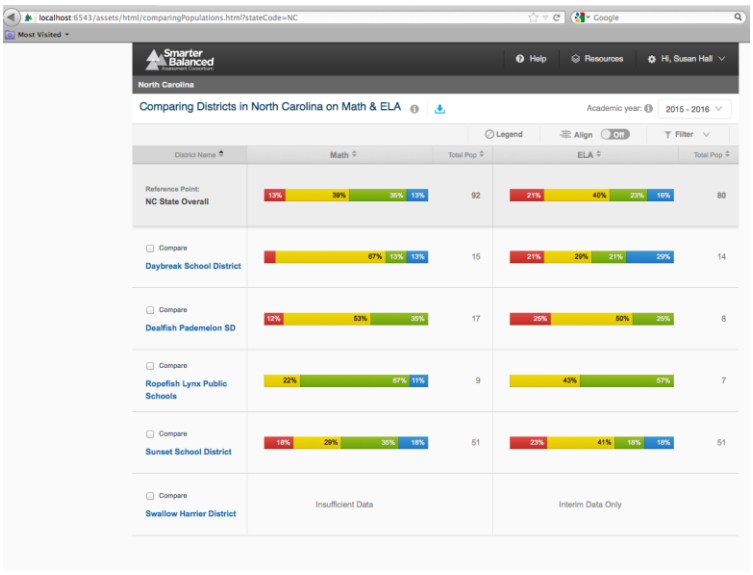
At this point, you're ready to start the server. You can either start it with Aptana, by hitting Run using the Run configuration that you have set up previously, or on a terminal, you can issue the command:

```
(virtualenv) cd <repo>/config
(virtualenv) pserve development.ini
```

At this point, your server should be running on port 6543. You can navigate to this URL on your browser, http://localhost:6543/assets/public/landing.html and should be able to see the login landing page.



Next, you can click on the Login Button and that should redirect you to your Identity Provider's login page. After logging in, you should be able to view the Comparing Population Report.



At this point, you've confirmed that Smarter web application is running.

3.6 Running Smarter Unit Tests

Smarter has a suite of Unit Tests that you can run. First, you'll need to install dependencies in your virtual environment.

```
(virtualenv) pip install nose
(virtualenv) pip install coverage
(virtualenv) pip install nose-cov
```

At this point, you're ready to run the unit tests.

```
(virtualenv) cd <repo>/smarter
(virtualenv) nosetests
```

3.7 Generating Smarter Code Documentation

Smarter uses a Python utility library, Sphinx, to generate its documentation. To generate code documentations, you'll first need to install dependencies in your virtual environment.

```
(virtualenv) cd <repo>/smarter
(virtualenv) python setup.py docs
```

Next, you can generate the html version of the documentation

```
(virtualenv) cd <repo>/smarter/docs
(virtualenv) make html
```

To view the documentation, navigate to
http://localhost/<pathToRepo>/smarter/docs/_build/html/index.html

4 Running Universal Data Loader (UDL)

UDL is responsible for ingesting a CSV/JSON pair of files. Currently, UDL supports two types of ingestion: student assessment and student registration. Please refer to our data dictionary documentation for the formats of these files.

Some basic concepts of UDL:

- Each tenant has its own landing zone for file drop off
- UDL loads CSV into the database via PostgreSQL dblink extension
- Each CSV goes through the pipeline of CSV → staging database → integration database → pre-prod database

4.1 INI Configuration File

UDL reads an INI file during application startup. A basic version of this INI configuration can be generated by the following commands, which constructs an INI file based on values defined in udl2_conf.yaml. You'll first need to install python dependencies (if you haven't done so previously).

```
(virtualenv) cd <repo>/config
(virtualenv) python setup.py develop
```

4.2 Configurations in INI file

Below is a list of commonly configured settings that should be modified depending on your environment

Configuration	Description
udl2_db_conn.url	This is the URL to your UDL staging, integration & stats database server. The format of this URL is: *postgresql+psycopg2://user:password@hostname:port/databaseName* Due to the usage of foreign data wrapper, this database server must reside in the same machine as where UDL celery workers are running from
target_db_conn.*[tenant]*.url	This is the URL to your pre-production database server. Tenant can be any string. When a tenant target database connection is set, a corresponding production database connection must be present in INI file.
prod_db_conn.*[tenant]*.url	This is the URL to your production database server.
prod_db_conn.*[tenant]*.db_schema	This is the schema name for your production database. This schema must exist when UDL starts up.
edware_stats.db.schema_name	The schema name of the stats database server. (This database is used to record statistics of batch jobs in UDL and will be used in EdMigrate to handle migration)
edware_stats.db.url	The database url of the stats database server. The format of this URL is: *postgresql+psycopg2://user:password@hostname:port/databaseName*
logging.audit	This is the path used for logging.

4.3 Copying Decryption Keys

UDL needs to decrypt data files, and we have supplied the corresponding decryption keys that are used to decrypt our test data files.

Three Files are supplied:

File Name	Description
secring.gpg	Lock file for secret keyring (Private keyring)
pubring.gpg	Public keyring
trustdb.gpg	Trust database

You'll need to copy the keys to your gpg home directory:

```
cp <repo>/edudl2/edudl2/tests/data/keys/* ~/.gnupg/
```

If you need to encrypt your own data files, here are some sample commands to tar and encrypt the file.

```
tar -cvzf test_data.tar.gz --disable-copyfile file1 file2

gpg --armor --local-user ca_user@ca.com --recipient sbac_data_provider@sbac.com --
encrypt --sign test_data.tar.gz
```

4.4 Starting UDL

To utilize the generic development environment's configuration, you'll need to create a pre-production database named edware, and a database user named, edware with password of edware2013. As well, you'll need to create a UDL database using the user udl2, and you can also name the database as udl2. These are the default configuration, you may configure your environment with other values, but you'll need to make sure that you update the INI file and/or udl2_conf.yaml.

You can connect to postgres via pgAdmin, and create a database and user through the user interface.

After you've created a database and user, you can begin and set up UDL dependencies.

```
(virtualenv) cd <repo>/edudl2
(virtualenv) python setup-developer.py develop
```

Please make sure that RabbitMQ is already running. You will now generate an INI file, initialize the UDL database, and start celery worker. Note that the script below will always re-generate an INI file, therefore if any configurations need to be overwritten, please make sure that they're done in udl2_conf.yaml.

```
(virtualenv) cd <repo>/edudl2/scripts
(virtualenv) ./start_local_udl.sh
```

You can verify that the workers have started by running:

```
ps -ef|grep celery
```

To initiate UDL pipeline to process the file, you'll first need to copy a .tar.gz.gpg file into the arrival zone of a tenant. You can copy an example from the test data.

```
mkdir -p /opt/edware/zones/landing/arrivals/[tenant]
cp <repo>/edudl2/edudl2/tests/data/test_data_latest/*.tar.gz.gpg
/opt/edware/zones/landing/arrivals/[tenant]
```

Next, you'll have to invoke UDL to process your file

```
(virtualenv) cd <repo>/edudl2/scripts
(virtualenv) python driver.py -a /opt/edware/zones/landing/arrivals/[tenant]/[nameOfFile]
```

There are a few verification that you can check for each ingestion job:

1. Check UDL logs

   The logs are located in /opt/edware/log/udl2.error.log
2. Check udl_batch table inside UDL database

   You should look for the row with UDL_COMPLETE as SUCCESS
3. Check that a new schema has been created in your pre-production database, and that the tables are populated with data.

4.5 Debugging UDL

UDL supports a development mode that allows developers to debug the source code. In your IDE, Aptana, you can add a run configuration for UDL with program arguments of --dev running main module edudl2/scripts/driver.py. Note: you may need to override the default tenant name with -t [tenant]

Under development mode, it will copy a file to the landing zone for you to a default tenant named 'cat' (Note: If you're using the default tenant, you will need to make sure that the database configuration for that tenant exists in your INI file)
You should be able to set a breakpoint in the code base and start debugging.

4.6 Running UDL Tests

UDL has a suite of tests that you can run. First, you'll need to install dependencies in your virtual environment (if you haven't already done so).

```
(virtualenv) pip install nose
(virtualenv) pip install coverage
(virtualenv) pip install nose-cov
```

At this point, you're ready to run the unit tests.

```
(virtualenv) cd <repo>/edudl2
(virtualenv) nosetests
```

4.7 Generating UDL Code Documentation

Smarter uses a Python utility library, Sphinx, to generate its documentation. To generate code documentations, you'll first need to install dependencies in your virtual environment.

```
(virtualenv) cd <repo>/edudl2
(virtualenv) python setup.py docs
```

Next, you can generate the html version of the documentation

```
(virtualenv) cd <repo>/edudl2/docs
(virtualenv) make html
```

To view the documentation, navigate to
http://localhost/<pathToRepo>/edudl/docs/_build/html/index.html

5 Running EdMigrate

When we receive new data through UDL, we want to minimize disturbance and outages to Smarter Web Application. In a production environment, we will migrate data from pre-production to production database on a scheduled basis. During this migration, we will rotate a group of database slaves out to replicate such that we can minimize any performance deterioration. In a local development environment, we can only test the actual data migration. Testing of slaves being detached from PGPool is out of scope in a local environment setting.

5.1 Generating INI Configuration File

EdMigrate reads an INI file on start up. A basic version of this INI configuration can be generated by the following commands, which builds an INI file based on values defined in settings.yaml. First, you should install python dependencies that we need.

```
(virtualenv) cd <repo>/config
(virtualenv) python setup.py develop
```

Next, generate the ini file,

```
(virtualenv) python generate_ini.py -e development
```

An INI file, development.ini, will be generated in the config directory.

5.2 Configurations in INI file

Below is a list of commonly configured settings that should be modified depending on your environment

Configuration	Description
migrate_dest.db.[tenant].schema_name	The schema name of the destination database server (ie. production database server)
migrate_dest.db.[tenant].url	The database url of the destination database server. The format of this URL is: *postgresql+psycopg2://user:password@hostname:port/databaseName*
migrate_source.db.[tenant].url	The database url of the source database server. (ie. pre-production database server)
edware_stats.db.schema_name	The schema name of the stats database server. (This database is used to record statistics of batch jobs in UDL, EdMigrate, and Smarter)
edware_stats.db.url	The database url of the stats database server. The format of this URL is: *postgresql+psycopg2://user:password@hostname:port/databaseName*

5.3 Starting EdMigrate

You'll first need to install all python dependencies that EdMigrate uses.

```
(virtualenv) cd <repo>/edmigrate
```

```
(virtualenv) python setup-developer.py develop
```

Next, you'll need to start main.py

```
(virtualenv) cd <repo>/edmigrate/edmigrate
(virtualenv) python main.py --migrateOnly -i <repo>/config/development.ini
```

The migration script will look for migration candidates inside udl_stats table of edware_stats database for any rows that have a status of 'udl.ingested'. EdMigrate will move data from destination database (pre-prod) using the schema name of the corresponding batch_guid value into source database (prod).

5.4 Running EdMigrate Unit Tests

EdMigrater has a suite of unit tests that you can run. First, you'll need to install dependencies in your virtual environment (if you haven't already done so).

```
(virtualenv) pip install nose
(virtualenv) pip install coverage
(virtualenv) pip install nose-cov
```

At this point, you're ready to run the unit tests.

```
(virtualenv) cd <repo>/edmigrate
(virtualenv) nosetests
```

5.5 Generating Edmigrate Code Documentation

Smarter uses a Python utility library, Sphinx, to generate its documentation. To generate code documentations, you'll first need to install dependencies in your virtual environment.

```
(virtualenv) cd <repo>/edmigrate
(virtualenv) python setup.py docs
```

Next, you can generate the html version of the documentation

```
(virtualenv) cd <repo>/edmigrate/docs
(virtualenv) make html
```

To view the documentation, navigate to
http://localhost/<pathToRepo>/edmigrate/docs/_build/html/index.html