TDS Deployment Evaluation

Prepared for: SMARTER BALANCED

Fairway Technologies, Inc.
April 27, 2016

Table of Contents

Executive Summary

Fairway has deployed and configured a development environment for the Smarter Balanced Open Source Test Delivery System that is capable of loading, delivering and scoring tests. The deployment process proved to be challenging and time-consuming. The system is not yet in a form that could be deployed without analyzing the source code and support from the development vendor.

In order to facilitate reliable deployment by assessment service providers and encourage contributions by other open source software contributors, Fairway recommends the following improvements:

- **Documentation should be organized, reviewed, and validated.** Documentation was difficult to find and details for some key features were missing. In some cases it lacked the level of detail required to deploy the system without vendor support. The documentation should be validated against the operational system to correct errors and fill in gaps.
- **Configuration options should be reduced and concentrated into a consistent location.** Configuration settings are spread across multiple locations including Program Management, local settings, and database values. Some configuration options are only accessible via direct database access. Concentrating configuration options in one location per component and updating the documentation accordingly will facilitate more rapid deployment and reduce the likelihood of errors.
- **Old code and artifacts should be removed to streamline the code for better performance and maintainability.** Legacy code and database objects clutter the system and the software structure follows patterns from the original C# framework that do not always match Java conventions.
- **Platform and technology choices should be more consistent.** The system uses multiple operating systems (Linux and Windows), databases (MySQL, MongoDB, and SQL Server), and programming languages (Java, C#, Python, and Perl). This requires a considerable breadth of knowledge and experience on the part of the development team.

Fairway recommends automating the build and deployment process, removing old code and artifacts, updating and consolidating documentation, and providing more complete seed data and sample configurations.

NOTE: This report is limited to the components Fairway installed to create an environment sufficient for test delivery. The following components were neither installed nor configured by Fairway:

- Monitoring and Alerting
- Test Item Bank / Test Authoring
- Data Warehouse
- Portal

Build Process

Master Build Sequence

The Smarter App Deployment Guide specifies the overall deployment sequence of components (by virtue of the table of contents)[1] but the components must be built in a different order due to software dependencies and shared code. A combination of trial and error, code analysis and support from AIR was required to determine the proper build order of the repositories necessary to produce the files required for deployment.

Fairway recommends creating scripts that will get the latest code from the repositories and build the repositories in the proper order. Having scripts that automate the fetching and building of source code will simplify the build process and minimize the effort required to build the software components.

Unit Tests

The system uses Maven (a build management tool for Java applications) to build the Java projects. Although the projects include unit tests, Maven must be configured to prevent executing those tests during the build process using the `-DskipTests` command-line option. Since many unit and integration tests fail, Maven cannot build the artifacts without omitting all of the tests. The tests should be improved so that they consistently succeed unless there is a defect. Obsolete or inaccurate tests should be updated to be relevant or removed if they are no longer required.

Maven Settings

To properly download dependencies from the correct repositories, a customized `settings.xml` file (shown below) had to be used in order to override the default Maven `pom.xml` file.

```
<settings>
<activeProfiles>
<!--make the profile active all the time -->
<activeProfile>securecentral</activeProfile>
</activeProfiles>
<profiles>
<profile>
    <id>securecentral</id>
    <!--Override the repository (and pluginRepository) "central" from the Maven
Super POM -->
    <repositories>
        <repository>
            <id>central1</id>
```

[1] AIR, Smarter App Deployment Guide, pg. 2

```
            <url>http://repo1.maven.org/maven2</url>
            <releases>
                <enabled>true</enabled>
            </releases>
         </repository>
        <repository>
            <id>central</id>
            <url>http://repo.maven.apache.org/maven2</url>
            <releases>
                <enabled>true</enabled>
            </releases>
        </repository>
    </repositories>
    <pluginRepositories>
        <pluginRepository>
            <id>central1</id>
            <url>http://repo1.maven.org/maven2</url>
            <releases>
                <enabled>true</enabled>
            </releases>
        </pluginRepository>
        <pluginRepository>
            <id>central</id>
            <url>http://repo.maven.apache.org/maven2</url>
            <releases>
                <enabled>true</enabled>
            </releases>
        </pluginRepository>
    </pluginRepositories>
</profile>
</profiles>
</settings>
```

Additional options must be passed to the `mvn` command line before the Java artifacts would successfully build. Specifically, the following options must be passed to `mvn: -DskipTests -DXmx2048m -DXX:MaxPermSize=1024m`.

Shown below is an example of the `mvn` command used by Fairway to build the Java artifacts:

```
mvn -q clean install -DskipTests -DXmx2048m -DXX:MaxPermSize=1024m -f /path/to/pom.xml
```

Deployment Process

Overall

This section describes deployment issues that are not specific to any particular software component. The issues cited below indicate issues with the deployment process as a whole or issues that were common across several software components.

Deployment Verification Process

There is no easy way to determine if a component (or the system as a whole) has been properly deployed and configured. A system administrator or developer must conduct a test by hand to verify that the component can communicate with its dependencies.

Fairway recommends providing documentation and examples of how a user can test a component to verify it has been deployed successfully. For example, the OpenDJ repository includes some scripts to generate sample user data. These scripts can be used to exercise OpenDJ and make sure a user account can be created properly. Similar scripts (e.g. sample `cURL` calls) would be beneficial in instilling confidence that a component is operating as expected.

Environment Definition

There is no indication that environment names are created via the MySQL seed data scripts that create and populate the Test Delivery System (TDS) databases. Knowledge of these values is essential to properly configuring software components in the Program Management (ProgMan) component. If the environment name in the database does not match the environment name in ProgMan, the settings stored in ProgMan will not be applied and the dependent software component will not launch.

For example, the database scripts name the environment “Development”. Section 5.3 of the Smarter App System Deployment Guide does not mention this setting. If the environment name in ProgMan does not match what is defined in the database (specifically in the `session._externs` table), the TDS will not work properly. There is no way to affect the `session._externs` table through any user interface, meaning the environment value must be verified by querying the database.

Additional documentation describing what an environment is and how it relates to ProgMan (and by extension other components) would be extremely helpful for an administrator or developer who is attempting to configure a TDS environment for the first time.

Disparate Technology Stacks

The majority of TDS system components are written Java. There are two components, Teacher Hand-Scoring System (THSS) and Test Integration System (TIS), that are written leveraging the Microsoft.NET framework in C#, thus requiring Windows servers for hosting. While there are methods for hosting Microsoft.NET applications on Linux (via Mono or .NET Core), it can be difficult to find support for such an undertaking. Furthermore, hosting a Smarter Balanced component built with Microsoft.NET on

Linux may not be feasible; the component may leverage a feature in the .NET framework that may not be available in Mono or the newer .NET Core.

Having a variety of servers on disparate technology stacks requires administrators with a variety of skillsets. Additionally, a single administrator may not be able to maintain the entire environment; proficiency in Linux and Windows is required.

Documentation

Incomplete Documentation

Section 5.6.1.1 of the Smarter App System Deployment Guide states "Configure the IIS webserver to access the application"[2]. No additional instructions, suggestions or recommendations for IIS configuration are given. There is no mention of how to configure application pools, virtual directories, etc. for a .NET TDS component.

There are no database diagrams provided for any of the databases. The TDS databases (particularly the `session` and `itembank` databases) contain enough objects to warrant a database diagram for customers/users who do not have the tools to generate one on their own. The `permissions` database is small but column names can be cryptic, making a diagram is useful for gaining a deeper understanding of how the database objects relate to one another.

Dispersed and/or Difficult to Find

In many cases, the documentation for building a server or deploying a component is located in several places. In some cases, (e.g. Single Sign-On setup/deployment) the Smarter App System Deployment Guide refers the reader to the BitBucket repositories. `README.md` which in turn refers the reader to another document stored within the repository. In a few cases (e.g. OpenDJ installation script output reveals login information) the documentation is stored in installation/deployment scripts.

Troubleshooting Assistance

There is very little troubleshooting/support documentation. A forum[3] exists where users attempting to deploy the TDS can review existing topics or ask a question. If an error is encountered during deployment of a component, the user is left to his/her own recognizance to resolve it. In the case of a Smarter App component issue, the user must:

- Review source code of the installation script, database script or in some cases the program being deployed to identify the source of the error
- Set up and configure remote debugging for JVM-based applications and step through the code in a debugger to determine the issue
- Contact the original developer of the component to get troubleshooting assistance

[2] AIR, Smarter App Deployment Guide, pg. 21
[3] Open Source Test Delivery System Forums

Fairway strongly recommends creating a troubleshooting knowledgebase to assist users with issues encountered during deployment. Refer to the Troubleshooting/Knowledgebase Guide section for Fairway's recommendations on how to achieve this.

Configuration

Configuring the Smarter Balanced Open Source System is complicated. There are many configuration options that are not accessible by any user interface. Definitions are not provided for many configuration parameters, leading to difficulty understanding what changes need to be made where. Many configuration changes must be made directly against the database. After these changes are made, it is unclear which applications are affected and if any applications need to be restarted (i.e. because the previous value was cached at startup).

Tomcat and MySQL Server Configuration

Beyond basic installation and configuration instructions, there is no additional information on how to configure/tune/optimize the Tomcat and MySQL servers. Fairway conducted their own tuning/optimization of Tomcat and MySQL servers based on the output of their scalability/load test suite. Fairway also solicited configuration recommendations from AIR based on their experience running the Smarter Balanced Open Source System under load.

While it would not be practical to provide specifications for every possible combination of servers, some general guidelines regarding server configuration would be helpful. For example, provide some configuration advice to support 100,000 students, which might be different than an environment expecting to support 15,000 students.

Tenant Configuration

Tenant configuration is not well-defined in the Smarter App System Deployment Guide. Section 7.2[4] gives a brief overview of the entity hierarchy, but no details are provided on what each "level" means. The "Managing Tenants" section[5] of the Program Management User guide does not describe the tenant hierarchy or what effect it has on the overall system. The "Managing Records" section[6] of the Administration and Registration Tools User Guide makes some mention of the entity hierarchy levels, but does not clearly describe their association.

The `README.md` files for student_release and proctor_release provide misleading configuration information, adding difficulty to the Student and Proctor configuration. Specifically, they do not provide a description of the client name and state code setting, nor do they indicate a change is required for the new environment.

[4] AIR, Smarter App Deployment Guide, pg. 71
[5] AIR, Program Management System User Guide, pg. 12
[6] AIR, Administration and Registration Tools Guide, pg. 41

Student Settings[7]

```
student.ClientName=SBAC_PT
```

```
student.StateCode=SBAC_PT
```

Proctor Settings[8]

```
proctor.StateCode=SBAC_PT
```

```
proctor.ClientName=SBAC_PT
```

The lack of detail led to a great deal of confusion regarding how the Student and Proctor components should be configured with their client/tenant settings. The correct value for the `StateCode` is not identified anywhere in the documentation provided. Fairway was able to discover the following by debugging the tenancy chain code:

- The `ClientName` should be set to whatever CLIENT-level account has been created
- The `StateCode` value should be set to the STATE-level tenant configured in ProgMan

The correlation between the `StateCode` setting and the STATE-level tenant in ProgMan is not defined in any documentation. The name of the `StateCode` setting does not imply that it is related to a tenant in any way. If the `StateCode` is not correctly configured, a user will not see any assessments in the Proctor user interface.

Fairway recommends providing additional documentation that clearly explains the tenancy hierarchy and how the tenant levels relate to one another. A brief sample is explained in the Smarter App System Deployment Guide[9], but additional samples and use cases would be extremely helpful in providing insight into the tenancy hierarchy subsystem. Additionally, a correlation between the `StateCode` setting in the Student and Proctor components and assessment availability should be clearly defined.

SAML Metadata Generation and Configuration

There are several components that use SAML for authentication (e.g. Program Management, Administration and Registration Tools, etc.) thus requiring configuration within the Single Sign-On services, specifically OpenAM. There is some documentation in the Smarter App System Deployment

[7] https://bitbucket.org/sbacoss/tds_release
[8] https://bitbucket.org/sbacoss/tds_release
[9] AIR, Smarter App Deployment Guide, pg. 71

Guide (Sections 5.5[10], 5.6[11] and 6.1.2[12], reference to sharedsecurity_release repository[13]). This documentation is in several places, making the information difficult to digest.

The SAML-setup.md and SAML-OAuth.md files provided in the sharedsecurity_release repository provide some general information on configuring SSO and contain links to the Spring Security documentation for more details. The information provided is not enough to allow someone without experience using OpenAM and Spring's implementation of SAML to configure the system successfully without needing to do further research. By reading the external Spring Security documentation and trying various approaches, the following configuration was used for each component:

- Add a `metadataGenerator` definition to the `securityContext.xml` file as described in the Spring Security documentation:

```
<bean id="metadataGeneratorFilter"
class="org.springframework.security.saml.metadata.MetadataGeneratorFilter">
    <constructor-arg ref="metadataGenerator"/>
</bean>

<bean id="metadataGenerator"
class="org.springframework.security.saml.metadata.MetadataGenerator">
    <property name="bindingsSSO">
        <list>
            <value>redirect</value>
            <value>artifact</value>
        </list>
    </property>
    <property name="entityId" value="[entity id of component]"/>
</bean>
```

- Restart the Tomcat server

After the `metadataGenerator` configuration was added to the `securityContext.xml` file, visiting the component's URL followed by `/saml/metadata` (e.g. `http://progman.sbtds.org/saml/metadata`) produced the proper SAML metadata.

Fairway recommends consolidating the SAML setup and configuration for software components to a single location. There are many steps to properly configure a component for SAML authentication, so a checklist with examples would be extremely helpful in simplifying the deployment and configuration process.

[10] AIR, Smarter App Deployment Guide, pg. 19
[11] AIR, Smarter App Deployment Guide, pg. 21
[12] AIR, Smarter App Deployment Guide, pg. 24
[13] AIR, Smarter App Deployment Guide, pg. 25

Deployment Automation/Provisioning

The Smarter App System Deployment Guide provides some documentation on the provisioning code provided in the administrative_release repository. While there are some good resources in the Smarter App System Deployment Guide, the provisioning software proved difficult to work with.

Complexity

The provisioning code provided within the administrative_release repository brings with it a considerable amount of complexity. There are several files that must be configured (`machine_configs.txt`, `external_vars`, `firewalls.txt`, `ansible.cfg`). Without context (configuration of these files is mentioned before any of the software components), it can be difficult to understand what configuration information is being asked for.

Fairway spent time trying to use the automation/provisioning resources provided by the administrative_release repository without success. Ultimately Fairway abandoned the provisioning code in favor of configuring the servers manually.

Documentation

The Smarter App System Deployment Guide has several pages devoted to the Ansible and shell scripts provided in the administrative_release. Some areas of this documentation (e.g. Section 5.3.6[14]) provide good background information while others (e.g. Section 5.3.5[15]) assume the reader has some knowledge of the system he/she is attempting to configure the provisioning software for (e.g. the `sasl_password` configuration item).

Fairway recommends changing the order of the sections within section 5.3; the instructions will be easier for the reader to follow if section 5.3.6 came before sections 5.3.3, 5.3.4 and 5.3.5. With this change, the tools used for provisioning will be introduced to the reader prior to discussing how to configure said tool.

Deployment.sh Script

The `deployment.sh` script (located here in the administrative_release repository) has an incorrect path for the command to clone the repository necessary to provision a server. The script points to `provisioningdev`, but should be pointed to `provisioning`. If a user attempts to run the `deployment.sh` script to provision a server, he/she will receive an authorization failed error. Fairway contacted AIR to determine the solution for this issue.

Missing Files

The following files are not included in the administrative_release repository:

- `tomcat-juli-adapters.jar`
- `mysql-connector-java-5.1.22-bin.jar`

[14] AIR, Smarter App Deployment Guide, pg. 15
[15] AIR, Smarter App Deployment Guide, pg.14

The Ansible scripts reference the files cited above to provision a Tomcat server. Without these files, the programs that rely on them cannot run. Since Fairway ultimately opted to install Tomcat manually (via a package manager), the files cited above were copied to the servers manually.

Specific Versions of Tomcat, MongoDB and MySQL

The Ansible playbooks refer to specific versions of tomcat and MySQL. While MySQL and MongoDB are installed using the Linux package manager appropriate for the distribution, the Tomcat server is installed from files checked into the administrative_release repository[16]. This means that the Ansible playbooks will only ever install that specific version of tomcat.

Version numbers are also specified for MySQL and MongoDB, but the package manager has the ability to fetch updated packages for those versions (e.g. a patch build of MySQL 5.6 with a security fix/enhancement).

SFTP Server Configuration

The `firewall_dependencies.xlsx` file dictates that several software components communicate to one another over Secure File Transfer Protocol (SFTP). The spreadsheet and Ansible scripts provided in the administrative_release repository imply that a stand-alone server should be provisioned[17]. While configuring the environment, Fairway determined that this was not necessarily the case for all components.

Application Deployment and Configuration

Shared Services

The sections below describe deployment issues that were encountered while following the instructions provided in section 6.1 of the Smarter App System Deployment Guide.

Single Sign-On (OpenDJ and OpenAM)

Documentation

Section 6.1.2 of the Smarter App System Deployment Guide[18] instructs the reader to find documentation for installing OpenDJ and OpenAM in their respective repositories. The installation documentation is mentioned in the `README.md` file under the Content Overview header. However, the location of the installation documentation is not standard between repositories. The OpenDJ installation document is in opendj_release/sbacInstaller along with the installation scripts. The OpenAM installation document is in the root of the repository: openam12_release.

[16] AIR, `/administrative_release/environment/ansible/roles/tomcat/tasks/main.yml`, line 23

[17] `/administrative_release/environment/ansible/sftp_servers.yml`

[18] AIR, Smarter App System Deployment Guide, pg. 24

Version Dependency

The OpenDJ and OpenAM repositories contain a specific version of the software from ForgeRock. These versions are out of date with the current software releases maintained by ForgeRock. As of this writing, the current stable releases of the ForgeRock software are:

- OpenDJ Enterprise: 3.0
- OpenAM Enterprise: 13.0

The versions of OpenDJ and OpenAM stored in the BitBucket repositories (and thus installed as part of the deployment) are:

- OpenDJ: 2.6.0
- OpenAM: 12.0.0 Build 12813

The OpenDJ and OpenAM installation scripts provided in their repositories specify a Java version (`1.6.0_45`[19] and `1.7.0_76`[20] respectively) that is outdated and no longer supported. The OpenAM installation script checks for the specific version of Java, therefore we needed to edit `installOpenAM.sh` and change the java version from `1.7.0_76` to `1.7.0_80`.

Fairway recommends avoiding committing specific versions of OpenDJ and OpenAM to source control. As the specific versions of OpenDJ and OpenAM get older, documentation and support for deploying those packages will be increasingly difficult to find. Additionally, the installation scripts should relax the Java version requirement. The latest update of Java should be sufficient; enforcing a specific update number of Java should not be necessary. Fairway could not find evidence of a specific Java update version on ForgeRock's site. The OpenDJ Installation Guide states that "OpenDJ relies on Java 6 or later,"[21]. The OpenAM Release Notes state Oracle Java 7, 8 and IBM Java (Websphere only) 7[22] are supported.

OpenDJ - Minimum Server Specifications

When deploying OpenDJ to the development environment, Fairway used a t2.micro instance. During the installation process, the following error was encountered:

```
Error occurred during initialization of VM

Could not reserve enough space for object heap

ERROR:  The detected Java version could not be used with the set of
Java
```

[19] `installOpenDJ.sh`, line 19
[20] `installOpenAM.sh`, line 19
[21] ForgeRock, http://opendj.forgerock.org/opendj-server/doc/bootstrap/install-guide/index.html
[22] ForgeRock, https://backstage.forgerock.com/#!/docs/openam/current/release-notes/chap-before-you-install

```
arguments -server -Xms2048M -Xmx2048M -Xmn128M -XX:+UseConcMarkSweepGC
-XX:MaxTenuringThreshold=1 -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -
Xloggc:/opt/opendj/logs/gc.log.
```

The arguments cited in the error above were set during the OpenDJ installation, using values from the `opendj/config/java.properties` file[23]. The `opendj/config/java.properties` file is stored in an archive called `OpenDJ-customizations.zip`, which is used during the installation process.

To resolve this issue, the AWS instance was destroyed and a larger instance was used to support the installation. The JVM options are not specified anywhere in the documentation that is available. Recommended specifications for the OpenDJ server are included in the SBAC OpenDJ Installation Instructions[24]. The recommended server specifications exceed the free tier for AWS and do not indicate the load it expects to support. As such, these specifications may be well beyond what is required for a development or testing environment.

OpenDJ – User Import Process

Managing users is a result of an integration between OpenDJ and ART. When a new user is created in ART, data for that user account is transmitted to the OpenDJ server via SFTP. This process requires an SFTP server to be set up, otherwise new user account cannot be transferred from ART to OpenDJ.

Documentation on configuring the integration between ART and OpenDJ is non-existent. The OpenDJ repository contains a document named SBAC OpenDJ Installation Instructions, which describes creating a user account and directory that receives files via SFTP. Based on the data provided in the `firewall_dependencies.xlsx` file, an administrator could infer that a stand-alone SFTP server should be created.

For the system to work properly, an administrator must install an SFTP server on the OpenDJ server. The `odj_servers.yml` file in the administrative_release repository[25] is not assigned the `sftp` role, meaning that an SFTP server will not be installed on an OpenDJ server provisioned by Ansible. Ultimately, there is no clarity on how or where the SFTP server should be installed. The SFTP server is essential to the integration between ART and OpenDJ. Without the means to transmit user account data from ART to OpenDJ, those changes cannot be recorded.

There are two scripts that facilitate importing new user accounts into OpenDJ: `sbacWatchXMLFolder.pl` and `sbacProcessXML.pl`. These scripts do not actively "pull" data from another source; they monitor the "dropbox" directory for new XML files produced by the Administration and Registration Tools (ART) when user account data arrives. There is no mention of how these scripts should be configured or executed.

After examining the Perl scripts and reviewing deployment notes provided by a third party who had previously deployed the TDS application, Fairway was able to deduce that an SFTP server must be set

[23] `opendj/config/java.properties`, line 106
[24] SBAC OpenDJ Installation Instructions, pg. 1
[25] `/administrative_release/environment/ansible/odj_servers.yml`, line 6

up on OpenDJ. Additionally, Fairway discovered the following steps for setting up the user import scripts:

- Create a "dropbox" directory that will be monitored by the `sbacWatchXMLFolder.pl` script
- Put user XML file into directory created in previous step
    - e.g. `/opt/dropbox/ftp_root/prime_user_testfile_.xml`
- Update `sbacProcessXML.pl` configuration (specific settings used by Fairway shown below):
    - `my $inputXMLFileDir = "/opt/dropbox/ftp_root"; # folder where the XML files are uploaded`
    - `my $processedFileDir = "/opt/scripts/sbacXMLFiles"; # folder where the XML files are stored after processing`
    - `my $httpResponseServer = "https://www.example.com/callback/"; # HTTP server for callback response`
    - `my $ldapHost = "localhost"; # host name of the OpenDJ server`
    - `my $ldapPort = "1389"; # port number of the OpenDJ server`
    - `my $ldapBindDN = "cn=SBAC Admin"; # replace with the bindDN of a service account or rootDN with permissions`
    - `my $ldapBindPass = "cangetin"; # replace with password of the OpenDJ service account`
    - `my $ldapBaseDN = "ou=People,dc=smarterbalanced,dc=org"; # location where the users may be found`
    - `my $ldapTimeout = "10"; # how long to wait for a connection to the LDAP server before timing out`
- Start the `sbacProcessXML.pl` script.
    - **NOTE:** The `sbacProcessXML.pl` is intended to run as a background process/job.
- Start `sbacProcessXML.pl` as a `nohup` job. If the script runs in the foreground, then it will wait indefinitely.
- After the `sbacProcessXML.pl` has been started, `touch` the `prime_user_testfile_.xml` to update the file's timestamp
- Once the timestamp is updated, the `sbacProcessXML.pl` will write results to the console.

- To verify the user has been created, use Apache Directory Studio (a program developed by Apache that provides a user interface to interact with LDAP [Lightweight Directory Access Protocol] servers) to connect to the OpenDJ instance and refresh the user list.

While creating the initial prime user, Fairway developers noticed that if an SMTP server is not configured, the `sbacProcessXML.pl` script will fail without creating the user account. We recommend the documentation be updated to point this out in order to avoid confusion during deployment and an error message shown if SMTP is not configured properly.

Program Management (ProgMan)

The ProgMan component is typically the first Smarter Balanced Open Source System component deployed after setting up OpenDJ and OpenAM. ProgMan requires complex configuration that must be set up properly for other components to function. Because ProgMan is a component upon which the bulk of the system depends, it has a unique configuration process.

Property Definition for Software Components

Part of defining properties for a software component is specifying which environment the properties should apply to. The term "environment" is never clearly defined in any document. Additionally, some environments are defined in the seed data scripts that populate the TDS databases. In particular, an environment named "Development" is created. There is no indication to the reader that this environment exists in the database. Therefore, when a user is logged into the ProgMan user interface, he/she does not know what value to provide for the Environment field. Providing an incorrect value will cause components dependent on ProgMan to throw an exception on startup and fail to run.

Property Value Discovery

It is difficult to discern the proper configuration values for a software component. Some repositories contain sample files with property settings, but these files are not always easily identifiable (i.e. they are not called out in a document or in a `README.md` file). These examples are not consistently located in a repository and are not always called out in documentation, meaning a reader may not be aware the sample files exist.

Permissions

The Permissions component is the first component deployed after ProgMan is properly configured. As such, this is the first component that depends on ProgMan for its configuration. Without a prior model/example to follow, configuring Permissions to interact with ProgMan adds complexity to this component's deployment. Combine ProgMan dependency configuration with the manual configuration of roles, components and permissions, and deploying the Permissions component is a difficult/error-prone process.

Database Scripts

There are two repositories that contain scripts to create the permissions database:

- permissions_release

- tdsdll_release

The Smarter App Deployment Guide does not explicitly indicate which repository contains the correct scripts for creating the Permissions database schema. Section 6.1.6.2 of the Smarter App Deployment Guide states "Use the SQL scripts provided in the repository to create and populate the Permissions database."[26] Fairway contacted AIR to determine which scripts are appropriate for creating the Permissions database schema.

Seed Data

There is a seed data script (`InsertStartupData.sql`) used to define the default client hierarchy. However, there is no seed data for the database or explicit instructions for configuring permissions, components or roles in the Permissions application. In the Documents directory of the permissions_release repository, there is an Excel spreadsheet named `permissions-roles-components.xlsx`. This file is not mentioned in the `README.md` or the Smarter App Deployment Guide. An administrator can follow the `permissions-roles-components.xlsx` file to set up components, roles and permissions, but this process is time-consuming and error-prone as the user navigates the Permissions UI and configures them manually. After populating the Permissions application for the first time, Fairway created a seed data script that loads the permissions matrix as defined in `permissions-roles-components.xlsx`.

Incomplete List of Roles and Permissions

There is at least one role missing from the `permissions-roles-components.xlsx` spreadsheet. The Test Integration Services (TIS) application will not function unless there is a "TIS Admin" role added to the permissions application. Additionally, the Teacher Hand Scoring Service (THSS) requires the permissions "Can View All Items" and "Can View Own Items."

Prime User Cannot Log In

After the Permissions program is deployed and the seed data script has been run, a user must log in and configure the components, roles and permissions (following the `permissions-roles-components.xlsx` spreadsheet cited in Seed Data). The Prime User account should have sufficient access to log into Permissions. However, Fairway found that the prime user account (which had been successfully tested against ProgMan) could not log into Permissions until the following SQL was executed against the Permissions database:

```
START TRANSACTION;
INSERT INTO component(`name`) VALUES('Permissions');
INSERT INTO role(`name`) VALUES('Administrator');
INSERT INTO permission(`name`) VALUES('Permissions Read');
INSERT INTO permission(`name`) VALUES('Permissions Admin');
COMMIT;

/* Associate Administrator Role to Permissions component and Read/Admin privileges */
```

[26] AIR, Smarter App Deployment Guide, pg. 30

```
START TRANSACTION;
INSERT INTO permission_role(`_fk_rid`, `_fk_cid`, `_fk_pid`) VALUES(1, 1, 1);
INSERT INTO permission_role(`_fk_rid`, `_fk_cid`, `_fk_pid`) VALUES(1, 1, 2);
COMMIT;

/* Associate role to entity type */
START TRANSACTION;
INSERT INTO role_entity(_fk_rid, _fk_etuk) VALUES(1, 'CLIENT');
INSERT INTO role_entity(_fk_rid, _fk_etuk) VALUES(1, 'STATE');
COMMIT;
```

After the SQL shown above was executed, the prime user could log into the Permissions application and perform the required tasks. According to AIR, the SQL should not have been necessary. However, Fairway found that the prime user account (which was the only account available at the time) could not properly log into Permissions without the Permissions component configured in its database.

Assessment Delivery Components

The sections below describe deployment issues that were encountered while following the instructions provided in section 6.3 of the Smarter App System Deployment Guide.

Administration and Registration Tools (ART)

Deployment of the ART component was moderately difficult. The `README.md` for the adminandregtools_release repository provide instructions for generating SSH keys (used for the user import process described in the OpenDJ section (pg. 15)). There is some configuration complexity that must be dealt with after ART has been deployed.

Time Zone Limitation

ART can only support a single time zone, which can cause data issues if servers are set up in different time zones. There is no explicit indication in the deployment guide that all servers should be set in the same time zone. A time zone discrepancy amongst servers can cause unexpected behavior and inconsistencies in data.

Another implication of this limitation is a single installation of the Smarter Balanced Open Source System cannot support locations that span time zones. If the system administrator configuring the environment is not aware of the single time zone limitation, they may expect the Smarter Balanced Open Source System to support tenants in multiple time zones. This limitation could lead to unexpected behavior (e.g. assessment time window availability), causing troubleshooting overhead and a poor experience for users. If the system administrator is aware of the limitation, they can work with their business stakeholders to plan accordingly.

Fairway recommends addressing the time zone limitation by storing all data in UTC and displaying the local time in the user interface. If it is not possible to address the single time zone limitation, alerting the reader that all clients must be in the same time zone.

Scheduling Subsystem

There is no indication that the scheduling subsystem provided by ART is purely a reporting tool that has no effect on the availability of assessments. Fairway's understanding is that some customers use the scheduling subsystem to assist in reporting. The ART User Guide does not explicitly state the scheduling subsystem affects assessment availability and could be construed to imply the opposite. While attempting to deploy an assessment to the TDS, Fairway spent a considerable amount of time troubleshooting the scheduling subsystem on the assumption it affected the assessment's availability. After reaching out to AIR for assistance, we learned that the scheduling subsystem does not affect if or when an assessment is available.

File Upload Templates

The file upload templates provided by ART that allow users to upload a collection of records (e.g. students) are not in the correct format, resulting in a failed upload. The correct format is not easily discernable. Fairway had to discover the correct format via trial and error combined with debugging sessions.

Proctor

Missing Seed Data

After deploying the Proctor application and configuring it via ProgMan, Fairway found that the Proctor application would not run. On startup, the following exception was recorded:

```
TDS.Shared.Exceptions.ReturnStatusException: Failed reading application config.
```

After debugging the Proctor program, Fairway was able to determine some seed data was missing from the Session database. The following SQL was used to resolve the issue:

```
/* Add records for SBAC and SBAC_PT to the session._externs table.  This table
is referenced by the externs view */
USE session;
START TRANSACTION;
INSERT _externs (clientname, environment, shiftwindowstart, shiftwindowend,
shiftformstart, shiftformend, shiftftstart, shiftftend)
VALUES('SBAC', 'dev', 0, 0, 0, 0, 0, 0);
INSERT _externs (clientname, environment, shiftwindowstart, shiftwindowend,
shiftformstart, shiftformend, shiftftstart, shiftftend)
VALUES('SBAC_PT', 'dev', 0, 0, 0, 0, 0, 0);
COMMIT;
```

```
/* Change the environment for SBAC and SBAC_PT from "Development" to "dev" (the
value used by Fairway in ProgMan and JAVA_OPTS settings for the tomcat
servers).  The Environment is set to "dev" in ProgMan for all configuration
properties. */
SET SQL_SAFE_UPDATES = 0;
START TRANSACTION;
UPDATE configs.client_externs
SET environment = 'dev'
WHERE environment = 'Development';
COMMIT;
SET SQL_SAFE_UPDATES = 1;
```

Student

Relative to the other components of the system, the Student application is easy to deploy. The Student application does not rely on SAML for authentication, thus there is no need for any integration with OpenAM. Even so, a number of issues were encountered while deploying the Student component.

Case-Sensitive Configuration Values

For localization, there is a parameter in the database (`configs.client_testtool`) that is case-sensitive. The seed data script that populates the `configs.client_testtool` table sets the value to `'language'`. The value should be set to `'Language'`. When this value is not set properly, messages are not translated/localized. There is no apparent way to change this configuration value from any user interface; a database update via SQL must be made.

Fairway created a SQL script that will set this configuration value appropriately and recommends updating the seed data script appropriately.

Item Scoring Service

The Item Scoring Service component is moderately easy to deploy. A Python interpreter and a small number of Python libraries must be installed to properly deploy the Item Scoring Service program.

Python Library Installation

The Item Scoring Service uses a customized version of the SymPy library. Therefore, SymPy cannot be installed using the Ubuntu package manager and the Python package manager `pip` must be used instead. After installing the SymPy library using `pip`, files must be copied from the installation location to the destination directory where the Python interpreter can find it. Instructions on how to install the customized SymPy library are included in the Running Item Scoring Engine document[27], but there is no indication in the `README.md` that this file exists.

CherryPy 3.6.0 was not installed via the Ubuntu or Python package managers. Instead, the library is fetched via a `wget` command and extracted to the destination directory. Fairway opted for fetching

[27] AIR, Running Item Scoring Engine, pg. 2

CherryPy 3.6.0 because the `pip install 'cherrypy==3.6.0'` command for CherryPy 3.6.0 throws an exception.

Moving the content of the Running Item Scoring Engine document to the `README.md` file would make understanding the installation process of the Item Scoring Service much easier.

Database Configuration

The last page of the Running Item Scoring Engine document[28] states the `configs.client_itemscoringconfig` table should be updated to point to the Item Scoring Service endpoint. A SQL script is not included to make this update. Fairway used the following SQL to configure the Item Scoring Service:

```
SET SQL_SAFE_UPDATES = 0;
START TRANSACTION;
UPDATE configs.client_itemscoringconfig
SET serverurl = 'http://localhost:8080/itemscoring/Scoring/ItemScoring'
WHERE serverurl != 'http://localhost:8080/itemscoring/Scoring/ItemScoring';
COMMIT;
SET SQL_SAFE_UPDATES = 1;

SET SQL_SAFE_UPDATES = 0;
START TRANSACTION;
UPDATE configs.client_itemscoringconfig
SET clientname = 'SBAC_PT'
WHERE clientname = 'SBAC';
COMMIT;
SET SQL_SAFE_UPDATES = 1;

SET SQL_SAFE_UPDATES = 0;
START TRANSACTION;
UPDATE configs.client_itemscoringconfig
SET environment = 'dev';
COMMIT;
SET SQL_SAFE_UPDATES = 1;
```

Running the Item Scoring Service

The `README.md` for the Item Scoring Service recommends running the process as a background task. Fairway also recommends creating a startup script so the Item Scoring Service launches when the server is rebooted. Additionally, a program to monitor the health of the service such as supervisord would be beneficial in insuring the Item Scoring Service is available to the rest of the system. Yet

[28] AIR, Running Item Scoring Engine, pg. 5

another consideration is to register the Item Scoring Service as a service within Linux, so it could be controlled like any other service (e.g. `httpd` or `mysql`).

Test Delivery System Database Deployment

Overall, creating and seeding the database schemas for the Test Delivery System (TDS) was simple. Instructions provided on the tds_release `README.md` file are clear and script execution is listed in the correct order.

MySQL Database Scripts

There were three bugs in the database deployment scripts that prevented creating the schemas necessary to support the Student and Proctor applications:

```
script: /tdsdll_release/tds-dll-
schemas/src/main/resources/import/genericsbacconfig/gen2.sql
error: ERROR 1136 (21S01) at line 2912: Column count doesn't match value count
at row 1

script: /tdsdll_release/tds-dll-
schemas/src/main/resources/import/sessionupdates/sb1282_student_proctor_package
_changes.sql
error: ERROR 1091 (42000) at line 3: Can't DROP
'ix_r_studentpackage_skey_client_currt'; check that column/key exists

script: /tdsdll_release/tds-dll-
schemas/src/main/resources/import/genericsbacconfig/sb1277_other accommodation
addition.sql
error: ERROR 1060 (42S21) at line 1: Duplicate column name 'IsEntryControl'
```

Fairway reported these errors to AIR, who quickly resolved them and created a new release with the fixed scripts. The fixes provided by AIR resolved the issues encountered with the scripts, allowing Fairway to continue with their deployment.

To simplify the database deployment even further, Fairway created a shell script that will create the required databases and execute the scripts in the correct order. This shell script eliminates the need to manually execute the SQL scripts to create the database schemas.

Assessment Scoring

The sections below describe deployment issues that were encountered while following the instructions provided in section 6.4 of the Smarter App System Deployment Guide.

Student Report Processor

Non-Standard Configuration

While most other Smarter Balanced components consume configuration information from ProgMan at startup, the Student Report Processor (SRP) configures itself via options passed in as `JAVA_OPTS`. There is no indication in the `README.md` or in code that the SRP can consume configuration information from ProgMan.

Configuration Values

The deployment documentation defines a startup script that should be created in order to run the standalone application. The example shows the arguments that need to be passed in and defines placeholders that you will need to change to fit your specific environment. The placeholders are not defined and there are no examples given so determining some of the values is difficult.

Specifically, the URLs for TIS and TIS status callback were difficult to configure properly. We needed to analyze the source code in order to determine where and how they were being used. It is not clear if the TIS URL is the base URL and the REST endpoint is appended in the code, or if it is the full URL.

Here is the current example from the documentation for setting the arguments:

```
reportingDLL.tisUrl="<TIS URL>"
reportingDLL.tisStatusCallbackUrl="<TIS Status Callback URL>"
```

Changing the above example to the following will make the intent clear and allow administrators to set the values without needing to look in the code:

```
reportingDLL.tisUrl="<TIS BASE URL>/api/testresult"
reportingDLL.tisStatusCallbackUrl="<TDS STUDENT BASE URL>/tisReply"
```

Test Integration System

Build and Publish

The process of building and publishing the TDS Receiver, TIS Service and TIS Scoring Daemon is simple and works as expected without requiring any alterations to the source code. .NET developers familiar with Visual Studio should not have any issues building and publishing the application.

Installing TIS Service

The documentation states: “Run `InstallUtil` for .Net 4.5, 32-bit on `/tis_opentestsystem/TDSQAService.exe` to install the windows service.” Experienced .NET developers will understand what this entails, however, considering most of TDS is installed on Linux, administrators tasked with installing TIS might need to do more research to understand what needs to be done. Providing more details would make the process simpler to follow.

Additionally, the executable that is created when building the project is `TISService.exe` and not `TDSQAService.exe` as noted in the documentation.

After installing the service using `InstallUtil.exe`, two separate services are installed: OSS_TTISService and TISService. This is not documented and therefore we expected installing TISService.exe as the service using `InstallUtil` would result in the TISService and we did not notice the OSS_TISService initially. Starting the TISService results in an error and leads to research and debugging to determine what might have been done incorrectly. In fact, the OSS_TISService is the correct service to use and works as expected once found.

We recommend providing more details on process of installing the TISService and specifically mention the correct service that needs to be run after a successful installation.

Database Deployment

Overall, the database deployment documentation is concise and easy to follow. One correction should be made: the OSS_TIS database configuration script `2_Configuration.sql` has been broken down into two separate scripts: `2_Configuration_IAB Tests.sql` and `2_Configuration_ICA_OP Tests.sql`.

While following the documentation is straightforward, there are 15 individual SQL scripts that must be run. We recommend combining these scripts into a single SQL script that creates all database objects and inserts seed data.

Seed Data

In order for tests to be processed from the queue, the test name must match a record within the `TestNameLookUp` table. Fairway found this when debugging why the tests that were successfully sent to TIS were not being processed. The seed data scripts populate the `TestNameLookUp` table with tests for the 2014-2015 academic year. After an initial installation, the current IRP and practice tests will not be processed by TIS. We did not see any documentation mentioning this fact in the deployment guide.

After manually installing records into `TestNameLookUp` for the IRP tests used in our environment, we were able to get TIS working as expected.

Configuration

TIS restricts access to submit test results to users connecting via OAuth that have the "TIS Admin" role assigned to them. This information was not provided in the deployment documentation for the Student Report Processor (SRP) of TIS. Fairway needed to analyze and debug the code in order to see where it was getting blocked.

Within Permissions, a new role of "TIS Admin" must be created. This role must then be granted to the user defined in the SRP run script argument `oauth.tis.username`. We did this manually in OpenDJ but ART could be used as well.

We recommend clarifying this process in the documentation for both Student Report Processor and TIS to make it clear.

Teacher Hand-Scoring System

Build and Publish

The Visual Studio publishing process is a common approach to deploying applications since it builds the project and only includes assets and resources needed for a production deployment. Visual Studio is not able to publish the current code because there is a project reference to `\images\favicon.ico` that does not exist on disk. This causes the publish process to fail.

Additionally, there is a missing Visual Studio project reference for `\App_Data\OpenAmSites.xml` which causes the file to not be included when the application is deployed using the publish process. This leads to an error that does not allow the application dashboard to be displayed and the application to fail.

We recommended updating the Visual Studio project file to remove the reference to `favicon.ico` and add a reference to `OpenAmSites.xml`, therefore allowing the publish process to work as expected.

Database Deployment

Overall, the database deployment documentation is concise and easy to follow, however, some details are incorrect and should be updated.

The paths listed for the SQL scripts is incorrect and caused a few moments of confusion when they were not found, however, it was simple to determine the correct path. The paths listed start with `<root>\Src\DB\TSS` whereas the correct path in source control is `<root>\DB\TSS`.

The SQL scripts are listed in the order they are supposed to be run but the first three scripts are not ordered correctly due to foreign key constraints. The documentation should be updated with the following order:

- `Items.sql`
- `Dimensions.sql`
- `ConditionCodes.sql`

Additionally, `Districts.sql` and `Schools.sql` are listed but are not files that exist in source control and appear to be unnecessary. These items should be removed from the documentation in order to avoid confusion.

Fairway found that the stored procedure `dbo.sp_UpdateDistrictTeacherRelationship.sql` contains SQL for altering an existing stored procedure instead of creating a new one. The fix is simply changing `ALTER PROCEDURE` to `CREATE PROCEDURE`.

While following the documentation is straight-forward, there are many individual SQL scripts that need to be run (37 in all; one for each table, stored procedure and function). We recommend combining these scripts into a single SQL script that creates all database objects.

Missing Database Function

The SQL function `fn_DecrpytValue` is never created since its source code is not available in source control. Without this function, the dashboard cannot be shown thus the application cannot be used.

SSO Integration

In order to integrate THSS with OpenAM there are five files that need to be created (`fedlet.cot`, `idp-extended.xml`, `idp.xml`, `sp-extended.xml`, and `sp.xml`). No details or examples are

provided. The documentation states, "These are all required to work with Open AM, but will differ with every deployment. Please refer to Open AM documentation for generating and configuring these documents."[29]

Fairway found guidance in creating these five files on Oracle's website documenting the use of OpenSSO and ASP.NET (https://docs.oracle.com/cd/E19575-01/821-1818/gjvia/index.html). The OpenSSO Enterprise download includes template files with placeholders for `fedlet.cot`, `idp-extended.xml`, `sp-extended.xml`, and `sp.xml`.

Since there is no guidance within the THSS deployment documentation related to creating the `sp.xml` file, the proper THSS URL to use is unknown and requires trial and error or careful review of the source code. OpenAM needs to redirect back to /InitiateLogin.aspx in order to function properly.

Fairway recommends including template files for each of the five SSO related files and include more information on implementation specific to OpenAM as it is deployed for TDS.

Configuration

The `Web.config` configuration file includes two additional configuration files (`App_Data/DataDistribution.config` and `App_Data/settings.config`) which are missing from source control, yet included in the Visual Studio project file. This leads to a yellow warning icon when viewing the project and developer confusion. The documentation includes examples of both of these files with placeholders for the developer to replace. We recommend including these files in Visual Studio and source control in order to make it as clear as possible how to configure the application.

Within `settings.config`, there is a specific `AppSetting` named `ART_ENTITIES_DATA_CACHING_DAYS` which is missing. This can only be determined after deploying the application and reviewing the error logs to find out why the application is failing. This setting and a default value should be added to the documentation and the `settings.config` file when it is included in source control.

Additionally, new permissions must be added to the Permissions application for the Teacher Hand Scoring System and mapped to the roles for teachers and proctors in order to use THSS to hand score items. This requirement is not documented and Fairway had to analyze the source code to find these values and configure the permissions properly.

Assessment Deployment and Configuration

Deploying an assessment to the TDS is a complicated process that is not clearly documented. The documentation that does exist is in the `README.md` file of three repositories:

[29] https://bitbucket.org/sbacoss/teacherhandscoresys_release, `README.md`

- testspecbank_release (describes using the `load_reg_package.pl` script to "side load" assessments into ART)
- tds_release (describes using the `loader_main` stored procedure to load assessments into the Itembank database)
- student_release (describes loading content into IRiS, which coincidentally describes the same file path as the insert statements from the tds_release `README.md`)

From the Smarter App System Deployment Guide, it is not clear which components must be deployed in order for a user to load an assessment into the system. It is Fairway's understanding that the open source version of the Assessment Creation and Management Components are not complete. To load assessments into a deployed system, AIR uses their proprietary version of these components.

Terminology

The terms "Assessment" and "Test" are used interchangeably throughout the Smarter App System Deployment Guide[30] and neither term is clearly defined. There are no links to other documentation on the SmarterApp site that might contain definitions/clarification on the terms used to describe an assessment.

Assessment Item Files

Part of the assessment package is the collection of files that support the assessment item ("question" or "test question"). The location of these files is not defined in ProgMan or accessible by any user interface. The path to these files is defined and configured in two database tables in the Itembank database:

- `tblitembank`
- `tblclient`

The combination of several columns from the tables cited above can be used to build the path where the Student application expects the assessment item files to be. SQL Insert statements that write data to the `tblitembank` and `tblclient` tables are displayed in the `README.md` of the tds_release repository, but their significance is not made clear.

After the assessment item files are copied to the file system of the Student server, proper permissions must be applied. These permissions are not explicitly documented anywhere, but the directories must be owned by the Tomcat server user and allow read and execute permission on the assessment item files.

Case Sensitivity

Part of loading an assessment into the `Itembank` database on the MySQL TDS sever is setting the path to the assessment item files. The path to these files is stored in the database with a lower-case "I" (e.g. "item"). When the files are deployed to the file system, the file path has an upper-case "I" (e.g.

[30] AIR, Smarter App Deployment Guide, pg. 38

`/path/to/Item`). The case mismatch causes an error when attempting to view an assessment within the Student application. A database script must be run after an assessment is loaded into the `Itembank` database to update the path to match the file system. Fairway used the following SQL to address the case sensitivity issue:

```
SET SQL_SAFE_UPDATES = 0;
START TRANSACTION;
UPDATE itembank.tblitem
SET filepath = CONCAT("I", SUBSTRING(filepath, 2));
COMMIT;
SET SQL_SAFE_UPDATES = 1;
```

If subsequent tests are loaded into ART, the SQL above must be run again to make sure the file path in the database matches what has been deployed to the file system.

Fairway recommends creating a comprehensive user interface to simplify loading an assessment into an installed system, updating the `loader_main` stored procedure to address the case sensitivity issue, and creating comprehensive documentation for loading assessments into the system that are provided in one location.

Side-Load Script

The `load_reg_package.pl` script is written in Perl, thus requiring a Perl interpreter on whatever machine intends to run it. There is no documentation for what the script does or how to run it beyond the help text within the script itself. The documentation within the script is thorough with respect to what configuration options there are.

Recommendations

In addition to the recommendations that are provided within each specific area described above, Fairway has the following general recommendations in order to make the deployment process simpler and easier for those new to the TDS.

Consolidate Technology Stacks

Consider porting the Smarter Balanced components to a homogenous technology stack. Since the majority of the components are written in Java, a JVM-supported language is preferred. Having the entire system on a single technology stack will make deployment, maintenance and support easier.

Eliminate Version-Specific Software Dependencies

Remove dependencies on specific versions of software (open source or otherwise). In particular, remove dependencies on software that is no longer supported by vendors (e.g. Java 6 and Java 7). Avoid committing specific versions of software to version control unless absolutely necessary. Instead, provide detailed instructions on how to acquire the latest stable version of a software package and how to configure it to work within the Smarter Balanced system. Relying on an older version of a software package may cause significant troubleshooting overhead; documentation/support for the specific version can be difficult to find. By providing instructions on how to get the latest stable version of a dependency, the system administrator will be more likely to find support should he/she encounter an issue during deployment.

Deployment Automation

Containers

A Linux container is "an operating-system-level virtualization environment"[31]. A Linux container is similar to a virtual machine in the sense that multiple containers can run on a single Linux host. These containers are isolated from one another; a process from one container cannot directly affect another container. In effect, each Linux container behaves like a full-fledged Linux server.

Docker is a software package designed to automate the creation, deployment and running of Linux containers. Docker can be used to create containers for the various software components that comprise the Smarter Balanced Open Source System. These containers could be configured with the correct Java artifacts (e.g. .jar and .war files), database and/or application servers (e.g. MongoDB or Tomcat), file system permissions, and firewall access rules for a specific software component.

For example, Docker could be used to create a container for ProgMan, another for ART, one for Permissions and containers for OpenDJ and OpenAM. With these containers, the system administrator would only have to deploy these containers and make environment-specific changes (if any). Deploying the containers and making minor configuration changes will be significantly easier than deploying and configuring each software component by hand.

Server Provisioning

Strides have already been made to automate provisioning of servers by means of Ansible playbooks provided in the administrative_release repository. Unfortunately, the provisioning code provided in the administrative_release is difficult to work with and prone to configuration error, even after following the steps described in the Smarter App System Deployment Guide. Work in this direction should continue; the scripts should be hardened against error conditions and should aim to reduce the number of configuration items.

[31] https://en.wikipedia.org/wiki/LXC

Ansible can be used in conjunction with Docker to provision an environment comprised of Linux containers. The combination of Ansible and Docker will make for a simplified and automated deployment process that allows system administrators to quickly create a Smarter Balanced Open Source System environment. Furthermore, automating the environment creation in this manner makes deployment a repeatable process.

Configuration

Audit

Evaluate each configuration item for each application/component to determine its value. If the configuration item is not necessary (e.g. it was carried over from the port of the proprietary system or it has been made obsolete by feature implementation), take steps to remove it. This includes updating documentation to reflect the changes in the configuration items. Eliminating extraneous configuration, clearly defining the necessary configuration parameters and providing configuration examples will aid a system administrator in:

- Configuring the Smarter Balanced Open Source System correctly with minimal effort
- Understanding what is being configured and how configuration settings affect the system

Streamline

Where possible, consider using a service discovery mechanism to allow components to find the services they need to interact with. Software packages such as Eureka (used by Netflix on AWS) or Apache Zookeeper can be used to provide a service registry to assist with discovery. Furthermore, Apache Curator is a service discovery extension that can be used in conjunction with Zookeeper to ease the service discovery implementation for the JVM-specific components. This will simplify the manual configuration needed and make deployment easier and faster.

Simplify

Provide a user interface for any configuration values that affect the environment. If any of the configuration items affected by the user interface are cached on startup, the user interface should update the cached value so applications do not have to be restarted.

Documentation

Additional Details

Provide a glossary of terms/jargon used within the Smarter App System Deployment Guide. Additional diagrams of how system components interact would be extremely useful. Where possible, the documentation should make no assumptions on the knowledge of the reader. Consider providing a detailed and thorough checklist (either a document or interactive web page(s)) that allows

administrators to easily track the tasks required to deploy the Smarter Balanced Open Source System. Diagrams of the SAML interaction between components would be helpful.

Consolidation

Rather than having a plethora of documents in a variety of formats (pdf, readme files in code or on BitBucket), establish a website where the instructions can be found. A wiki would be ideal; it could be maintained by the community and moderated by Smarter Balanced (or whoever "owns" the source code). Having a single location for all deployment-related documentation would greatly simplify discovering how to deploy a particular component.

Additional Awareness/Availability

Adding links to the documentation contained within a component's repository will make the consumer aware that the documentation exists as well as making it easy to find. Documentation should be located in one area. Ideally, documentation would be in the form of a "living document" (e.g. a wiki) that can be maintained by the community and moderated by an appropriate authority.

Troubleshooting/Knowledgebase Guide

To assist in resolving issues, Fairway recommends collecting the issues encountered during various deployments of the Smarter Balanced Open Source System into one document. A wiki or other form of living document that is easily modifiable should be used. This guide would allow for quicker resolution of common problems without needing to contact support.

Seed Data

Seed data is sorely lacking for some software components (e.g. Permissions). Where possible, seed data should be provided for software components. If seed data needs to be customized to meet a customer's needs (e.g. changing a database name or creating an environment), those items should be called out in the software component's documentation.

Seed Data Scripts

Scripts to pre-populate databases with standard configuration data are provided for the TDS components, but should be provided for other components also (e.g. ProgMan, ART, Permissions). These seed data scripts should include sample data that is representative of what production data would look like. In effect, the seed data scripts could be used to set up a fictitious client, assessments, students, etc. Being able to see the system configured and with sample data will provide much needed clarity on how the Smarter Balanced Open Source System operates.

For example, ProgMan could be seeded with sample tenants and components or ART could be seeded with sample assessments (from the Implementation Readiness Package or the Practice Tests) and students who are eligible to take those assessments. Having the system configured with sample data will be immensely useful for users who are new to the system.

Sample Configuration Files

Many of the BItBucket repositories have sample configuration files, but these files are not readily apparent. Sample configuration files, even with configuration values from a fictitious system will greatly assist in providing clarity on how software components should be configured.