SmarterApp Open Source Online Assessment System Deployment and Configuration Guide

Table of Contents

1 Revision History

Version	Comments	Author	Date
0.1 - **Draft**	Initial release - work in progress	Rami Levy Ajay Kumar	30 Oct 2014
0.2 - Draft	Updated Bootstrapping (Section 7). Updated PM config (6.1.3).	Rami Levy	06 Nov 2014
0.3 - Draft	Updated ProgMan parameter lists for ART, TSB, TIB, Progman, and Test Authoring to account for the newly protected MNA REST interface.	Rami Levy	20 Nov 2014
0.4 - Draft	Updated Portal configuration information (6.1.11). Updated Network firewall configuration (4.2.2).	Rami Levy	03 Dec 2014
0.5 - Draft	Added loadbalancer configuration information for PM, TA, ART, TSB, and MNA.	Rami Levy	10 Dec 2014

2 References

Ref.	Reference	Author	Version
1	Assessment System Architecture and Technology Report http://www.smarterapp.org/architecture.html	Smarter Balanced	11 July 2014
2	Smarter Balanced Test Delivery Hosting Requirements http://www.smarterapp.org/specifications.html	Smarter Balanced	V2 (01 May 2014)
3	Smarter Balanced Test Delivery Hosting Cost Calculator http://www.smarterapp.org/specifications.html	Smarter Balanced	V2 (01 May 2014)
4	Smarter Balanced Component Dependencies Matrix (*Administrative* repository in Bitbucket: guide/Component Runtime Dependencies.pdf)	Smarter Balanced	Latest
5	SBAC SSO Component Design Document http://www.smarterapp.org/documents/SmarterBalanced_ArchitectureReport_07112014.pdf	Identity Fusion, Inc.	1.1 (11 Feb 2014)
6	Firewall Network Matrix reference (*Administrative* repository in Bitbucket: guide/firewall-network-matrix.pdf)	Smarter Balanced	Latest

Ref.	Repository Reference	URL
R1	Administrative	https://bitbucket.org/sbacoss/administrative_release
R2	Program Management (PM)	https://bitbucket.org/sbacoss/programmanagement_release
R3	Permissions	https://bitbucket.org/sbacoss/permissions_release
R4	Monitoring and Alerting (MNA)	https://bitbucket.org/sbacoss/monitoringandalerting_release
R5	Core Standards (CS)	https://bitbucket.org/sbacoss/corestandards_release
R6	Test Authoring	https://bitbucket.org/sbacoss/testauthoring_release
R7	Test Spec Bank (TSB)	https://bitbucket.org/sbacoss/testspecbank_release

R8	Test item Bank (TIB)	https://bitbucket.org/sbacoss/testitembank_release
R9	Administration and Registration Tools (ART)	https://bitbucket.org/sbacoss/adminandregtools_release
R10	Test Delivery (proctor)	https://bitbucket.org/sbacoss/tds_release
R11	Student	https://bitbucket.org/sbacoss/student_release
R12	Portal	https://bitbucket.org/sbacoss/portal_release
R13	OpenAM	https://bitbucket.org/sbacoss/openam_release
R14	OpenDJ	https://bitbucket.org/sbacoss/opendj_release
R15	Shared Security	https://bitbucket.org/sbacoss/sharedsecurity_release

3 Introduction

This guide provides instructions on how to deploy the Smarter Balanced Open Source Online Assessment System. Where appropriate, instructions are made to be hosting-environment independent.

3.1 Target Audience

This guide is primarily intended for system administrators.

3.2 How to Use This Guide

If you are new to the Smarter Balanced system, begin with the Architectural Overview below. If you know which components you intend to deploy, begin at *Section 4 (Initial Considerations)* of this document and follow each step there. Continue to *Section 5 (Server Creation)*. For most component deployments, Shared Service components will be required. Use the chart in *Section 6 (Component-Specific Deployments)* to determine which components are needed and follow the instructions in those sections, which will take you through post-deployment scenarios. Each server you create follows the steps outlined in Section 5, and is then customized by the steps in Section 6 (component-specific deployment).

3.3 Architectural Overview

The system's overall architecture is depicted below. Additional details are available at [1]. This Guide will reference the groupings depicted in this Figure that relate to Smarter Balanced Contract 11. This ***excludes*** the following components:

- Digital Library
- Item Authoring and Item Pool (IAIP), also known as Item Bank
- Reporting
- Data Warehouse

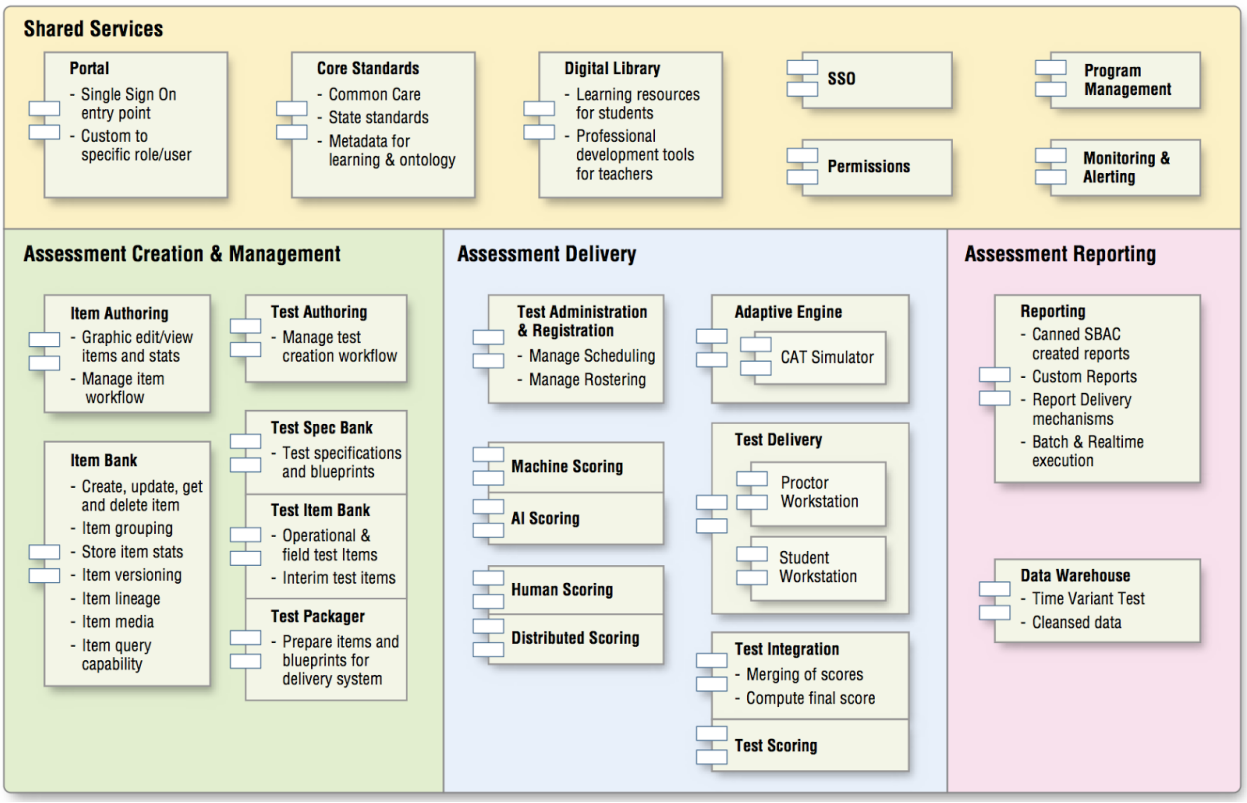


Figure 1: Smarter Balanced Architecture

4 Initial Considerations

4.1 Infrastructure Selection

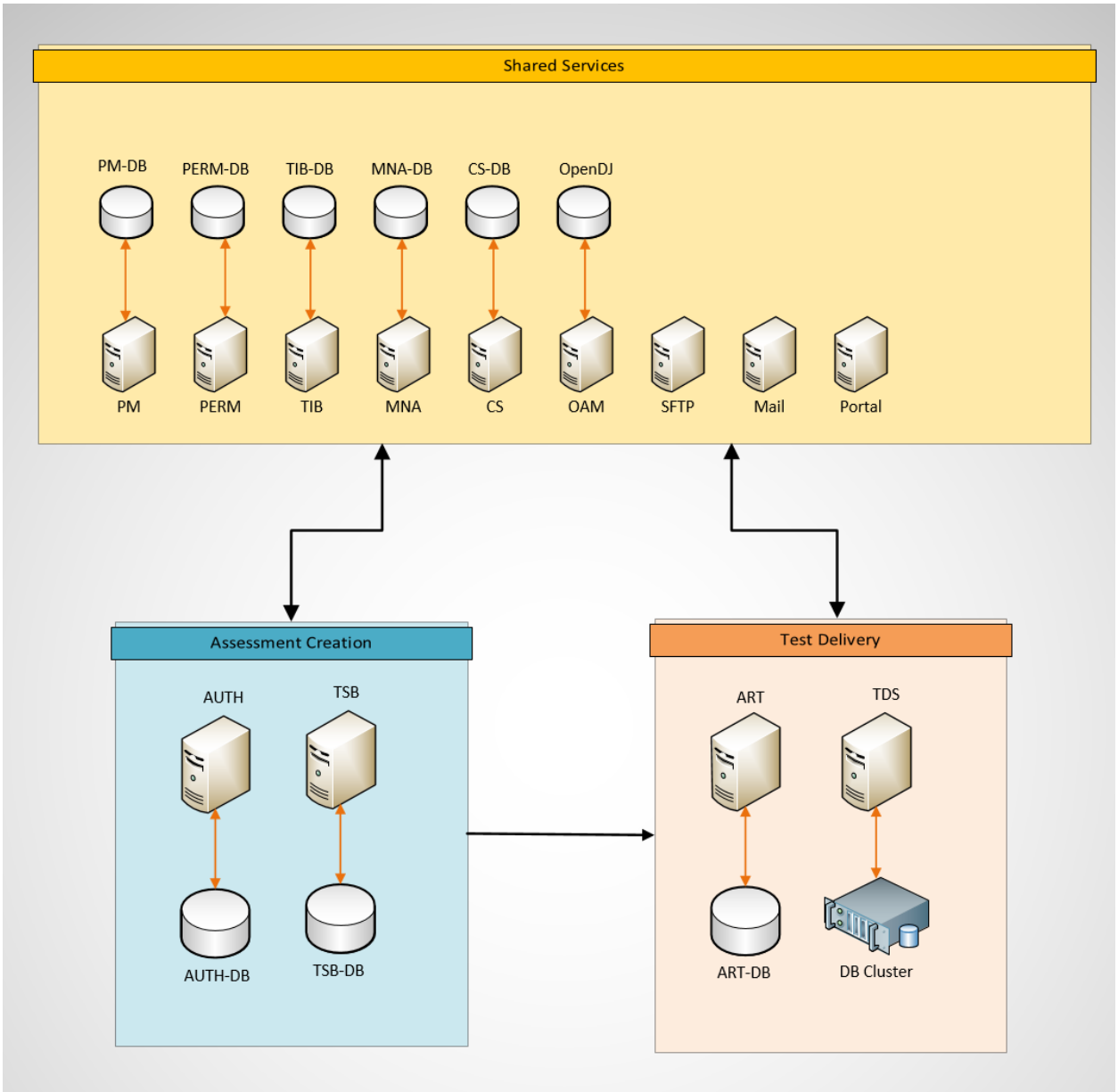
When selecting a target deployment infrastructure, there are a number of considerations to make, including performance requirements, cost, support, availability, and so forth. Examples of cloud-based infrastructure options are Amazon Web Services (AWS) and Rackspace. Alternatively, it would be possible to deploy to an in-house data center with physical rather than virtual hardware. Hybrid solutions are also possible; however for performance reasons it is strongly advised to co-locate all components within the same network wherever possible. Please refer to the *Deployment Assumptions* and *Deployment Configurations* sections in [2] for more information. Cost considerations for an AWS-based deployment are described in [3].

4.2 Server Instances and Network Topology Selection

Once the infrastructure is selected, the machine instance types, database configurations, and deployment node structures (e.g. load balancing) must be defined - again based on the deployer's specific requirements. Please refer to the *Test Delivery Unit* section in [2] for more information.

4.2.1 Example Topology

In one example implementation shown below, each service is deployed onto its own server, and connected with a dedicated database server. The TDS web server is attached to a DB cluster service (e.g. RDS). The TDS server could also be placed behind a load balancer, as can any other service.



4.2.2 Network Security / Access

For security purposes, it is recommended that a single machine be used as the entry point (via SSH) for administrators to manage all machines. The security groups on the remaining machines should be configured such that external access via ssh is blocked. Specifically, the configuration provided in [6] is recommended.

4.3 Domain Name Selection

It is recommended that all deployments in a single environment use the same domain name, to reduce complexity. The selection of a domain name (e.g. my.state.edu) also allows for the selection of a proper SSL wildcard certificate (see next section).

4.4 CA-Signed SSL Certificate

A Certificate Authority issues signed digital certificates which certify the ownership of a public key by the named subject of the certificate. Once a domain name is selected, it is strongly advised to purchase a *wildcard certificate* that will encompass your domain name.

5 Server Creation / Configuration / Provisioning Guide

Server creation is specific to the environment in which the components are being deployed. The instructions here will use Amazon (AWS) as an example; other environments should generally follow the same steps.

5.1 Assumptions

This guide assumes the administrator has full access rights to the target deployment environment.

5.2 Checklist

At this point, the following choices should have been made:

1. Deployment infrastructure (e.g. AWS);
2. Network topology (e.g. load balancers / HA servers);
3. Wildcard SSL certificates for the selected domain name;
4. Instance types for each component being deployed (e.g. AWS EC2 m3.large instance type for Test Authoring component) including databases;
5. Required (dependent) components for all components being deployed (see [4]).

For each of the components being deployed, follow the appropriate deployment guide.

5.3 Server Creation

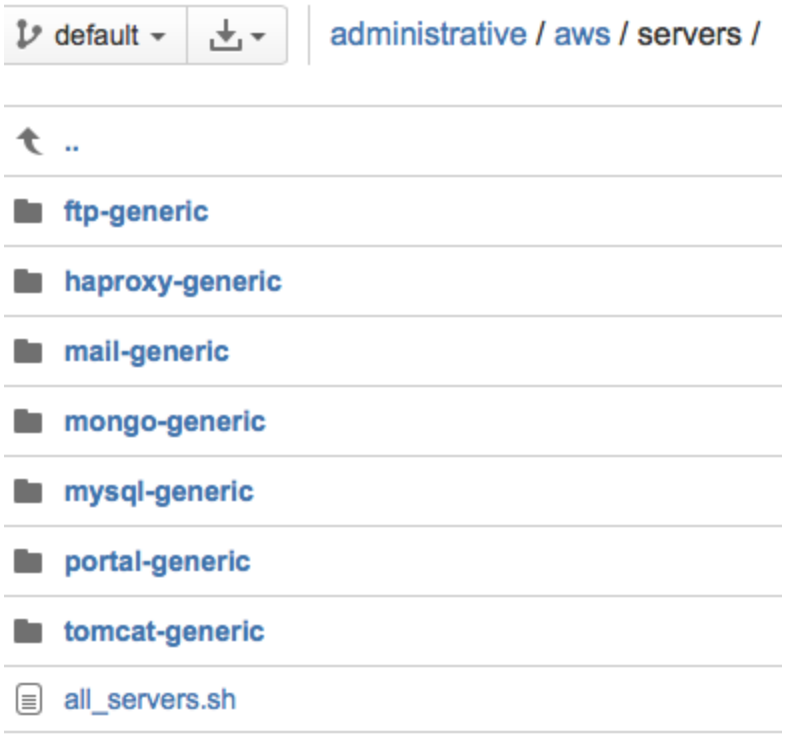
To create a server, follow the following steps:

1. Select an AMI from the publicly available AMIs. Select a 64-bit, Ubuntu Linux 14.04+ AMI using Paravirtual virtualization and EBS root device.
2. After selecting an appropriate instance type, select an availability zone (best is to keep the servers in the same AZ). It's recommended to enable termination protection as well.
3. Then select an appropriately-sized root partition, and add additional storage if needed.
4. Tag the instance by setting the `Name` key to the exact DNS name required for the instance you're creating, making sure it follows the wildcard pattern for your certificate.
5. Finally, create a new (or select an existing) security group for this server.

Once your server is created and launched, clone the *Administrative* repository [R1] locally in order to execute the appropriate scripts (see next section).

5.4 Server Configuration

The *Administrative* repository [R1] is set up in such a way as to properly configure servers of a particular type.



Within each of the subdirectories, there is a script called `transform_machine.sh`. which can be used to create that particular type of server. Prior to executing this script, security credentials related to your current configuration must be added to a few files under the administrative directory.

- `aws/root/usr/local/etc/ec2.env`:
  - Replace CHANGEME with your user's unique AWS credentials (keys).
- `aws/root/usr/local/nagios/etc/nrpe.cfg:`
  - Replace CHANGEME with a password for Tomcat manager, which you will set later
- `aws/servers/tomcat-generic/tomcat-users.xml:`
  - Replace CHANGETHISPASSWORD with the Tomcat manager password you set above.

Next, execute this script as follows:

```
$ transform_machine.sh /path/to/root.pem public_dns_name
```

Where *`root.pem`* is the complete path to the location of the PEM file that would be used to access the new machine, and *`public_dns_name`* represents the contents of the field *Public DNS Name* of the newly created instance in the EC2 dashboard. After the script completes, the machine will be available for root login via ssh with the root PEM. It will also be addressable via its new DNS name (according to the value specified in the *Name* tag during instance creation). The script performs the following functions, among others:

- Updates all the Ubuntu default installed software
- Installs a variety of software that are used in the environment - java, python, cpp, mercurial, etc.
- Turn on nightly security auto updates
- Installs files that are modified for the environment - `/etc/bash.bashrc`, `/etc/environment`, `/etc/timezone`, etc.
- Installs and runs convenience scripts:
    - `update_ssmtp_fqdn.sh` - runs at bootup to put the current FQDN in /etc/ssmtp/ssmtp.conf. This is the basis for all programs to find their hostname.
    - `update_route53_cname.sh` - runs at bootup update AWS Route53 (DNS server) to update FQDN to hostname mapping.
    - `update_authorized_keys.sh` - runs at bootup to remove AWS's change to the /root/.ssh/authorized_keys file to prevent root login.

5.5 Component Software Provisioning and Deployment

Each system component may either be configured manually, or via a provisioning script provided in the *Administrative* [R1] source repository. Provisioning a server essentially comprises these steps:

1. Deploying the component software package itself.
2. Configuring the system and specific installed components. For example, connecting the component to any number of shared services; setting server startup parameters, and setting component initialization parameters via Program Management.
3. Creating, installing, and populating databases for components where applicable.

The last two items will be covered in the Component-Specific Deployments section of this document.

5.6 Provisioning - General Information

Every directory in the *Administrative* [R1] repository follows the same general structure. Most importantly, it has a script named *install.sh*, along with its required supporting files. To provision a particular component, its associated *install.sh* script must be executed.

5.7 Tomcat Webserver SSL certificate installation

> ***NOTE:*** *SSL certificates may either be installed on either a load balancer (if used), or on the individual web servers. The information below describes how to do this on individual servers.*

The procedure for SSL certificate installation is common for all web server application components. The following steps are used to create and install SSL certs once they have been issued by a CA.

5.7.1 Installing SSL certificate to keystore

Add the originally created private key to a JKS keystore, which will be used for all servers. Place this keystore in a secure directory (usually `$CATALINA_BASE`/.ssh) with the appropriate

restrictions for private keys (e.g. chmod 600, ownership by tomcat only). For testing purposes, a self-signed certificate may be added, and replaced later by the CA-signed cert. Run the script `create_tomcat_certs.sh --help` (found at [R1]) for more information. Essentially this is what needs to happen:

1. Create a keystore using the (self-signed or CA-signed) private key
2. Import the trusted cert from the keystore into a trust store
3. Save both of these to the secure directory

5.7.2 Configuring the SSL Connector

> ***NOTE**: This step is automatically performed by the transform_server.sh script discussed above. You should only need to modify the values shown in red below to match your specific configuration.*

Tomcat's global connector options are configured in Tomcat's configuration file, located at `$CATALINA_BASE/conf/server.xml`. Modify it by adding a connector element to support for SSL or https connection. Add a new connector port 8443 section by copying the existing commented `Connector port="8443"` or uncomment and use the same section as a new connector port. Finally, add the keystore file, key alias server name and replace key store pass with yours. Save your changes and restart the server to make changes effective. Your connector port should look similar to the section below:

```
<Connector
        protocol="org.apache.coyote.http11.Http11NioProtocol"
        port="8443" maxThreads="200" maxHttpHeaderSize="16384"
        scheme="https" secure="true" SSLEnabled="true"
        keystoreFile="/path/to/keystore/file" keystorePass="your_keystore_password"
        keyAlias="your_server_key_name" clientAuth="false" sslProtocol="TLS"/>
```

5.7.3 Forcing HTTPS-only Connections in Tomcat

- In $CATALINA_HOME/webapps/application/WEB-INF/web.xml, add this after </servlet-mapping>, add this :

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Everything in the webapp</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

- Enable the load balancer to recognize server side HTTP / HTTPS request or redirects.

- In server.xml add this at the end, before </Host> :

```
<Valve className="org.apache.catalina.valves.RemoteIpValve"
        remoteIPHeader="X-Forwarded-For"
        remoteIPProxiesHeader="X-Forwarded-By"
        protocolHeader="X-Forwarded-Proto" />
```

- NOTE: Make sure the connector is redirecting 8080 to 8443. Example:

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" maxHttpHeaderSize="16384" />
```

6 Component-Specific Deployments

6.1 Shared Services Deployment

These components support the other components and create a cohesive system for the end users. Most components require the installation of the shared services components. Refer to [4] and the shared services chart below for details.

Deployed Component → / Dependent component ↓	Single Sign On (SSO)	Program Management (PM)	Permissions	Monitoring & Alerting (MNA)	Core Standards (CS)	Portal
Single Sign On (SSO)	n/a	Y	Y	Y	Y	Y
Program Management (PM)		n/a	Y	Y	Y	Y
Permissions			n/a			Y
Monitoring & Alerting (MNA)		optional	optional	n/a	optional	
Core Standards (CS)					n/a	
Portal						n/a

Table 1: Shared Services Component Startup Dependencies

Clearly from this chart, every shared service is dependent on SSO, and most are dependent on Program Management. These two should be the first components deployed in any system. Following those, it's reasonable to deploy the remaining shared service components shown above. Integration with MNA is optional but strongly encouraged.

6.1.1 SSO Deployment and Provisioning

There are three types of authentication that must be supported by this system.

- *User authentication*, where an application's user interface prompts a user to authenticate via a username and password (SAML).
- *Coordinated web services*, where the application itself makes calls to other applications on behalf of the logged in user, and the remote application must be able to validate the user's credentials in the background (SAML Bearer).

- *Machine to machine authentication*, where an application makes a call to another application on behalf of no user, but must still be validated. The SSO component is responsible for managing all three types of authentication, as well as facilitating authorization, using tokens as appropriate (OAuth).

For additional information on SAML and OAuth, please refer to the documentation in *Shared Security repository* [R15].
For creation and installation information, please refer to the OpenAM [R13] and OpenDJ [R14] repositories, which contain server creation and installation instructions and scripts.

6.1.2 Program Management Deployment

The Program Management component (also referred to as PM or ProgMan) is the master data repository and service. It is a set of services to provide data that crosses component concerns, as well as holding configuration and branding information for other components.

6.1.3 Program Management Provisioning

Please refer to the *Configuration/Provisioning Guide* section in this document to set up a new generic Tomcat server. The most up-to-date installation instructions are available in the *Program Management source repository* [R2].

6.1.3.1 Property Configuration

The Program Management application has its own set of application-specific configurations to bring it up. Since Program Management is the master data repository which maintains the configurations of all other applications, it is necessary to configure Program Management with the required properties files. These are placed in what is termed the `SB11_CONFIG_DIR`, located under `$CATALINA_HOME/resources`.
A default set of configurations are provided in the code repository. Below is a description of these properties:

Properties file **`progman-bootstrap.properties`** contains all of the following key/value pairs:

MnA properties

- `progman.mna.description` - {a descriptive name for the component used as a display in MnA}
- `mna.mnaUrl` - {url to the base context of the MNA REST application}
- `oauth.access.url` - {url to OAuth URL to OAM instance to allow client calls to POST to get an OAuth token for any 'machine to machine' calls requiring OAUTH}
- `mna.oauth.client.id` - {OAuth Client id configured in OAM to allow get an OAuth token for the ‘batch' web service call to MnA}
- `mna.oauth.client.secret` - {OAuth Client secret/password configured in OAM to allow get an OAuth token for the ‘batch' web service call to core standard}
- `mna.oauth.batch.account=` - {Username (email address) of MNA client user used for authenticating into MNA and logging metrics information}
- `mna.oauth.batch.password=` - {Password of MNA client user}

Mongo Properties

- `pm.mongo.hostname` - {hostname of the mongodb instance}
- `pm.mongo.user` - {mongodb username}
- `pm.mongo.password` - {mongodb password}
- `pm.mongo.dbname` - {mongodb name}

PBE properties

Data encryption requires an externally defined property to hold a secret password. Password-Based Encryption (PBE) is used to encrypt any sensitive values in the key value configurations. A salt is recommended but not required (the key must exist but the value may be left blank):

- pm.pbe.pass=

In order for encryption of data to work, the unlimited JCE security policy must be installed. Copy `encryption/UnlimitedJCEPolicy/local_policy.jar` and `encryption/UnlimitedJCEPolicy/US_export_policy.jar` into your JDK's lib/security folder, replacing the existing files (please back up existing files). See the README.txt in that directory for more details.

PM config properties

- `pm.rest.service.endpoint` - {fully qualified URL to base context of the rest webservice}
- `pm.rest.context.root` - {relative path to base context of the rest webservice}
- `pm.minJs` - {whether to use minimized javascript in the browser}
- `progman_resource_check_token_url` - {URL of OpenAM OAuth token check}
- `mna.logger.level` - {level of logging that will be sent to the monitoring and alerting. it defaults to ERROR if not set}

PM security properties

- `pm.security.saml.keystore.user` - {the cert for accessing SSO server via https)
- `pm.security.saml.keystore.pass` - {password for cert to access SSO server via https}
- `pm.security.dir` - {file:///opt/... (fully qualified path to location of saml metatdata files)}
- `pm.rest.saml.metadata.filename` - {name of the saml metadata file for the REST app contained in the security.dir configured above}
- `pm.webapp.saml.metadata.filename` - {name of the saml metadata file for the WEBAPP app contained in the security.dir configured above}
- `pm.security.idp` - {fully qualified path to the SAML 2 IDP}

Clustered Environment properties

These are *optional* properties which are used when configuring a clustered environment behind a load balancer (LB). To activate clustered environment support, simply change the

active profile setenv as follows: change spring.profiles.active from server.singleinstance to server.loadbalanced. Furthermore, you will need to set these key/value pairs appropriately:

- `progman.loadbalanced.url.scheme` - {this should be http or https}
- `progman.loadbalanced.server.name` - {the loadbalancer's name}
- `progman.loadbalanced.server.port` - {if your server requires a port, include it here, otherwise put 80 in as the default}
- `progman.loadbalanced.includeServerPortInRequestURL` - {boolean true/false value which indicates if the port should be included to resolve the server}
- `progman.loadbalanced.server.rest.contextpath` - {REST context name. e.g.: "/progman.rest"}
- `progman.loadbalanced.server.webapp.contextpath` - {webapp context name. e.g.: "/progman.webapp". Leave this blank if you are using ROOT as webapp context name}

Logback configs

logback configurations can also be placed in this file. These are optional and can be configured in the application's existing logback.xml file instead:

- `logfile.path` - {/path/to/wherever/logs/should/go/; e.g. /usr/local/tomcat/logs}
- `app.base.package.name` - program-management
- `app.context.name` - org.opentestsystem.shared.progman
- `app.base.package.loglevel` - debug

6.1.3.2 Database Configuration

Program Management uses a MongoDB database. Follow the steps in Section 5 of this document to create a generic MongoDB server. Progman will create a database with all required collections upon startup if one doesn't already exist.

6.1.3.3 Server startup configuration settings

Create or edit a file named `setenv.sh` under the `$CATALINA_HOME/bin` directory and define startup environmental variables. Here is a sample setenv.sh file; replace all key values with appropriate values.

```
export CATALINA_OPTS="-DSB11_CONFIG_DIR=$CATALINA_HOME/resources
-DmnaServerName=mna.example.org
-Dspring.profiles.active=mna.client.integration,progman.client.impl.null,special.role.required
-Djavax.net.ssl.trustStore=/trust/store/certs/path/.truststore.aws
-Djavax.net.ssl.trustStorePassword=XXXXXXX"
```

`mna.client.integration:` Integrate Progman with MNA. *This should be set to `mna.client.impl.null` if not using MNA or if MNA is not yet available*.
`progman.client.impl.null:` Use the Program Management null implementation because Progman cannot use itself.

`special.role.required:` This allows progman to use a local implementation of a role to permission binding for a given role. This removes the hard, runtime dependency on the Permissions component if it's unavailable.

Please refer to the *Program Management repository* [R2] for more information, especially about permissions and access.

6.1.4 Permissions Deployment

This is a centralized permissions management for the systems components. It is necessary to require that components share the same permissioning capabilities in order to reduce permissions management complexity, and to allow consistent user experience across multiple components developed by different vendors.

6.1.5 Permissions Provisioning

Please refer to the *Configuration/Provisioning Guide* section in this document to set up a new generic Tomcat server.

6.1.5.1 Property Configuration

Configuration settings for this component are defined via the Program Management application. Add all configuration settings based on the default set provided in the code repository.

Database Properties

The following parameters need to be configured inside program management for database.

- `datasource.url=jdbc:mysql://localhost:3306/schemaname` - The JDBC URL of the database from which Connections can and should be acquired.
- `datasource.username=<db-username>` - Username that will be used for the DataSource's default getConnection() method.
- `encrypt:datasource.password=<db-password>` - Password that will be used for the DataSource's default getConnection() method.
- `datasource.driverClassName=com.mysql.jdbc.Driver` - The fully qualified class name of the JDBC driverClass that is expected to provide Connections.
- `datasource.minPoolSize=5` - Minimum number of Connections a pool will maintain at any given time.
- `datasource.acquireIncrement=5` - Determines how many connections at a time datasource will try to acquire when the pool is exhausted.
- `datasource.maxPoolSize=20` - Maximum number of Connections a pool will maintain at any given time.
- `datasource.checkoutTimeout=60000` - The number of milliseconds a client calling getConnection() will wait for a Connection to be checked in or acquired when the pool is exhausted. Zero means wait indefinitely. Setting any positive value will cause the getConnection() call to time out and break with an SQLException after the specified number of milliseconds.

- datasource.maxConnectionAge=0 - Seconds, effectively a time to live. A Connection older than maxConnectionAge will be destroyed and purged from the pool. This differs from maxIdleTime in that it refers to absolute age. Even a Connection which has not been idle will be purged from the pool if it exceeds maxConnectionAge. Zero means no maximum absolute age is enforced.
- datasource.acquireRetryAttempts=5 - Defines how many times datasource will try to acquire a new Connection from the database before giving up. If this value is less than or equal to zero, datasource will keep trying to fetch a Connection indefinitely.

MNA Properties

The following parameters need to be configured inside Program Management for MNA.

- mna.mnaUrl=http://<mna-context-url>/mna-rest/ - URL of the Monitoring and Alerting client server's rest url.
- mnaServerName=permission_dev - Used by the mna clients to identify which server is sending the log/metrics/alerts.
- mnaNodeName=dev - Used by the mna clients to identify who is sending the log/metrics/alerts. There is a discrete mnaServerName and a node for server name & node1/node2 in a clustered environment giving the ability to search across clustered nodes by server name or for a given spcific node. it's being stored in the DB for metric/log/alert, but is not displayed.
- mna.logger.level=ERROR - Used to control what is logged to the Monitoring and Alerting system. Logging levels: ALL (turn on all logging levels), TRACE, DEBUG, INFO, WARN, ERROR, OFF (turn off logging).

SSO Properties

The following parameters need to be configured inside program management for SSO.

- permission.uri=https://<permission-app-context-url>/rest - The base URL of the REST api for the Permissions application.
- component.name=Permissions - The name of the component that this Permissions deployment represents. This must match the name of the component in Program Management and the name of the component in the Permissions application.
- permission.security.idp=https://<idp-url> - The URL of the SAML-based identity provider (OpenAM).
- permission.webapp.saml.metadata.filename=permissions_local_sp.xml - OpenAM Metadata file name uploaded for environment and placed inside server directory.
- permission.security.dir=file:////<sp-file-location-folder> - Location of the metadata file.
- permission.security.saml.keystore.cert=<cert-name> - Name of the Keystore cert being used.
- permission.security.saml.keystore.pass=<password> - Password for keystore cert.
- permission.security.saml.alias=permissions_webapp - Alias for identifying web application.
- oauth.tsb.client=tsb - OAuth Client id configured in OAM to allow the SAML bearer workflow to convert a SAML assertion into an OAuth token for the 'coordinated web service" call to TSB.

- oauth.access.url=https://<oauth-url> - OAuth URL to OAM to allow the SAML bearer workflow to POST to get an OAuth token for any ‘machine to machine calls requiring OAuth.
- encrypt:oauth.tsb.client.secret=<password> - OAuth Client secret/password configured in OAM (under the client id) to allow the SAML bearer workflow to convert a SAML assertion into an OAuth token for the "coordinated web service” call to TSB.
- encrypt:mna.oauth.client.secret=<password> - OAuth Client secret/password configured in OAM to allow get an OAuth token for the "batch" web service call to MnA.
- mna.oauth.client.id=mna - OAuth Client ID configured in OAM to allow get an OAuth token for the "batch" web service call to MnA.
- encrypt:permission.oauth.resource.client.secret=<password> - OAuth client secret/password configured in OAM to allow get an OAuth token for the "batch" web service call to permissions.
- permission.oauth.resource.client.id=permissions - OAuth Client ID configured in OAM to allow get an OAuth token for the "batch" web service call to Permissions.
- permission.oauth.checktoken.endpoint=http://<oauth-url> - OAuth URL to OAM to allow the SAML bearer workflow to perform a GET to check that an OAuth token is valid.

6.1.5.2 Database Configuration

Permissions uses a MySQL database. Follow the steps in Section 5 of this document to create a generic MySQL server. Use the SQL scripts provided in the repository to create and populate the Permissions database. InsertStartupData.sql adds the default SBAC hierarchy into the database.

6.1.5.3 Server startup configuration settings

Create or edit a file named `setenv.sh` under the `$CATALINA_HOME/bin` directory and define startup environmental variables. Here is a sample setenv.sh file; replace all key values with appropriate values. Refer to the *Permissions code repository* [R3] for the latest information.

- `-Dspring.profiles.active` - Active profiles should be comma separated. Typical profiles for this include:
    - `progman.client.impl.integration` - Integrate with Program Management
    - `progman.client.impl.null` - Use the Program Management null implementation
    - `mna.client.integration` - Integrate with MnA component
    - `mna.client.null` - Use the null MnA component
- `-Dprogman.baseUri` - This URI is the base URI where the Program Management REST module is deployed.
- `-Dprogman.locator` - The locator variable describes which combinations of name and environment (with optional overlay) should be loaded from Program Management. For example: "component1-urls,dev" would look up the name component1-urls for the dev environment at the configured REST endpoint. Multiple lookups can be performed by using a semicolon to delimit the pairs (or triplets with overlay): "component1-urls,dev;component1-other,dev"
- `-DSB11_CONFIG_DIR` - Locator string needed to find the Permissions properties to load.
- `-Djavax.net.ssl.trustStore` - Location of .jks file which contains security certificates for SSO, Program Management and Permission URLs specified inside baseUri and Program Management.
- `-Djavax.net.ssl.trustStorePassword` - Password string for the keystore.jks file.

Example:

```
export CATALINA_OPTS=" -Xms512m -Xmx1024m -XX:PermSize=512m -XX:MaxPermSize=512m
-Dspring.profiles.active=progman.client.impl.integration,mna.client.integration
-Dprogman.baseUri=http://pm.example.org:8080/rest/
-Dprogman.locator="permissions,Dev" -DmnaServerName=perm.example.org
-Djavax.net.ssl.trustStore=/trust/store/certs/path/.truststore.aws
-Djavax.net.ssl.trustStorePassword=XXXXXXX"
```

Please refer to the *Permissions code repository* [R3] for more information.

6.1.6 Monitoring and Alerting Deployment

Monitoring and Alerting, also known as MnA, is a shared set of services that allow components to send alerts in a consistent way. Also those alerts can be monitored and acted upon in a similarly consistent way. It also will allow vendors to develop add on applications and features to use and act on these alerts.

6.1.7 Monitoring and Alerting Provisioning

Please refer to the *Configuration/Provisioning Guide* section in this document to set up a new generic Tomcat server.

6.1.7.1 Property Configuration

Configuration settings for this component are defined via the Program Management application. Add all configuration settings based on the default set provided in the code repository [R4]. Please refer there for the latest information on setup and configuration.

- `mna.mongo.hostname=mongo.host` - The server on which MongoDB is running
- `mna.mongo.port=27017` - MongoDB port
- `mna.mongo.user=` - MongoDB user
- `mna.mongo.password=` MongoDB password
- `mna.mongo.dbname=mna` - Name of the MongoDB database
- `mna.email.active=true` - Enable or disable email from MNA
- `mna.email.address.from=from@somecorp.com` - From address when email from MNA is sent
- `mna.email.subject.prefix=TEST EMAIL ONLY -` - Subject of email from MNA
- `mna.email.host=email.host` - Host address of mail server
- `mna.email.port=465` - Email port
- `mna.email.user=emailuser` - Email username
- `mna.email.password=emailpassword` - Email password
- `mna.email.smtp.starttls.enable=true` - Start TLS enabled
- `mna.email.transport.protocol=smtp` - Email protocol used
- `mna.email.smtp.auth=true` - Use email authentication?
- `mna.email.smtp.ssl.enable=true` - Enable ssl over SMTP
- `mna.rest.context.root=/rest/` - MNA REST context root path
- `mna.clean.days=30` - How long to retain logs before cleaning out. Not required. defaults to 30
- `mna.clean.cron=0 0 0 * * ?` - timing for cron job. Not required. defaults to 0 0 0 * * ?

Clustered Environment properties

These are *optional* properties which are used when configuring a clustered environment behind a load balancer (LB). To activate clustered environment support, simply change the active profile setenv as follows: change `spring.profiles.active` from `server.singleinstance` to `server.loadbalanced`. Furthermore, you will need to set these key/value pairs appropriately:

- `mna.loadbalanced.url.scheme` - {this should be http or https}
- `mna.loadbalanced.server.name` - {the loadbalancer's name}
- `mna.loadbalanced.server.port` - {if your server requires a port, include it here, otherwise put 80 in as the default}
- `mna.loadbalanced.includeServerPortInRequestURL` - {boolean true/false value which indicates if the port should be included to resolve the server}
- `mna.loadbalanced.server.rest.contextpath` - {REST context name. e.g.: "/mna.rest"}
- `mna.loadbalanced.server.webapp.contextpath` - {webapp context name. e.g.: "/mna.webapp". Leave this blank if you are using ROOT as webapp context name}

6.1.7.2 Database Configuration

Monitoring and Alerting uses a MongoDB database. Follow the steps in Section 5 of this document to create a generic MongoDB server. MNA will create a database with all required collections upon startup if one doesn't already exist.

6.1.7.3 Server startup configuration settings

Create or edit a file named `setenv.sh` under the `$CATALINA_HOME/bin` directory and define startup environmental variables. Here is a sample setenv.sh file; replace all key values with appropriate values.

```
export CATALINA_OPTS="-Djava.rmi.server.hostname=mna.example.org
-DSB11_CONFIG_DIR=$CATALINA_HOME/resources
-DmnaServerName=mnaexample.org -Dspring.profiles.active=progman.client.impl.integration
-Dprogman.baseUri=https://pm.example.org/rest/
-Dprogman.locator=MNA,web-dev
-Djavax.net.ssl.trustStore=/trust/store/certs/path/.truststore.aws
-Djavax.net.ssl.trustStorePassword=XXXXXXX"
```

Please refer to the *Monitoring and Alerting repository* [R4] for more information.

6.1.8 Core Standards Deployment

This is the component that needs to manage the Common Core State Standards and learning metadata so that other components can reference and use them in the same manner. It is the single version of truth for these standards.The standards publication must be stored in an OpenOffice Calc spreadsheet with a .ods extension. OpenOffice was selected because it is an open-source spreadsheet application in keeping with the open-source nature of the technologies used in the Smarter Balanced Assessment System.

6.1.9 Core Standards Provisioning

Please refer to the *Configuration/Provisioning Guide* section in this document to set up a new generic Tomcat server.

6.1.9.1 Property Configuration

Configuration settings for this component are defined via the Program Management application. Modify the configuration settings based on the default set provided in the code repository.

Database Properties

The following parameters need to be configured inside Program Management for database.

- `datasource.url=jdbc:mysql://<db.url>:<db.port>/<schemaname>` - The JDBC URL of the database from which Connections can and should be acquired. Can be localhost. Port is usually 3306.
- `datasource.username=<db-username>` - Username that will be used for the DataSource's default getConnection() method.
- `datasource.password=<db-password>` - Password that will be used for the DataSource's default getConnection() method.
- `datasource.driverClassName=com.mysql.jdbc.Driver` - The fully qualified class name of the JDBC driverClass that is expected to provide Connections.
- `datasource.minPoolSize=5` - Minimum number of Connections a pool will maintain at any given time.
- `datasource.acquireIncrement=5` - Determines how many connections at a time datasource will try to acquire when the pool is exhausted.
- `datasource.maxPoolSize=20` - Maximum number of Connections a pool will maintain at any given time.
- `datasource.checkoutTimeout=60000` - The number of milliseconds a client calling getConnection() will wait for a Connection to be checked in or acquired when the pool is exhausted. Zero means wait indefinitely. Setting any positive value will cause the getConnection() call to time out and break with an SQLException after the specified number of milliseconds.
- `datasource.maxConnectionAge=0` - Seconds, effectively a time to live. A Connection older than maxConnectionAge will be destroyed and purged from the pool. This differs from maxIdleTime in that it refers to absolute age. Even a Connection which has not been idle will be purged from the pool if it exceeds maxConnectionAge. Zero means no maximum absolute age is enforced.
- `datasource.acquireRetryAttempts=5` - Defines how many times datasource will try to acquire a new Connection from the database before giving up. If this value is less than or equal to zero, datasource will keep trying to fetch a Connection indefinitely.

MNA (Monitoring and Alerting) properties

The following parameters need to be configured inside program management for MNA.

- `mna.mnaUrl=http://<mna-context-url>/mna-rest/` - URL of the Monitoring and Alerting client server's rest url.
- `mnaServerName=corestandards_dev` - Used by the MNA clients to identify which server is sending the log/metrics/alerts.
- `mnaNodeName=dev` - Used by the MNA clients to identify who is sending the log/metrics/alerts. There is a discrete mnaServerName and a node in case you want to say XXX for server name & node1/node2 in a clustered environment giving you the ability to search across clustered nodes by server name or specifically for a given node. It's being stored in the db for metric/log/alert, but not displayed.
- `mna.logger.level=ERROR` - Used to control what is logged to the Monitoring and Alerting system. Logging Levels (ALL - Turn on all logging levels; TRACE, DEBUG, INFO, WARN, ERROR, OFF - Turn off logging).

SSO properties

The following parameters need to be configured inside Program Management for SSO.

- `permission.uri=https://<permission-app-context-url>/rest` - The base URL of the REST API for the Permissions application.
- `component.name=CoreStandards` - The name of the component that this CoreStandards deployment represents. This must match the name of the component in Program Management and the name of the component in the Permissions application.
- `corestandards.security.idp=https://<idp-url>` - The URL of the SAML-based identity provider (OpenAM).
- `corestandards.webapp.saml.metadata.filename=corestandards_local_sp.xml` - Name of OpenAM SP (Service Provider) Metadata file which has been uploaded for the environment, as well as placed inside the server's filesystem.
- `corestandards.security.dir=file:////<sp-file-location-folder>` - Location of the metadata file.
- `corestandards.security.saml.keystore.cert=<cert-name>` - Name of the Keystore cert being used.
- `corestandards.security.saml.keystore.pass=<password>` - Password for keystore cert.
- `corestandards.security.saml.alias=corestandards_webapp` - Alias for identifying the web application.
- `oauth.tsb.client=tsb` - OAuth Client id configured in OAM to allow the SAML bearer workflow to convert a SAML assertion into an OAuth token for the 'coordinated web service" call to TSB.
- `oauth.access.url=https://<oauth-url>` - OAuth URL to OAM to allow the SAML bearer workflow to POST to get an OAuth token for any "machine to machine" calls requiring OAUTH.
- `encrypt:oauth.tsb.client.secret=<password>` - OAuth Client secret/password configured in OAM (under the client id) to allow the SAML bearer workflow to convert a SAML assertion into an OAuth token for the "coordinated web service" call to TSB.

- encrypt:mna.oauth.client.secret=<password> - OAuth Client secret/password configured in OAM to allow get an OAuth token for the "batch" web service call to MnA.
- mna.oauth.client.id=mna - OAuth Client id configured in OAM to allow get an OAuth token for the "batch" web service call to MnA.
- encrypt:corestandards.oauth.resource.client.secret=<password> - OAuth Client secret/password configured in OAM to allow it to get an OAuth token for the "batch" web service call to Core Standards.
- corestandards.oauth.resource.client.id=corestandards - OAuth Client id configured in OAM to allow it to get an OAuth token for the "batch" web service call to Core Standards.
- corestandards.oauth.checktoken.endpoint=http://<oauth-url> - OAuth URL to OAM to allow the SAML bearer workflow to perform a GET to check that an OAuth token is valid.

6.1.9.2 Database Configuration

Core Standards uses a MySQL database. Follow the steps in Section 5 of this document to create a generic MySQL server. Use the SQL script provided in the repository to create and populate the Core Standards database. Before executing the scripts, create the database by executing the following command:

```
create schema StandardsRepository_Dev default character set=UTF8;
```

Following this, execute the script `Create_StandardsRepository_Tables.sql`, followed by `StandardsRepository_sp_and_functions.sql`. Finally, to insert sample data into the DB, execute the `Sample_Data_for_StandardsRepository_db.sql` script.

6.1.9.3 Open Office (OO) configuration

> ***NOTE**: This step is automatically performed by the transform_server.sh script discussed above. The provisioning script also performs this installation if necessary.*

After completion of the Open Office installation, create a symlink called */usr/bin/soffice* pointing to */install/directory/path/openoffice4/program/soffice*. Finally add OO resource information to tomcat context.xml file.

```
<Parameter name="ooExecPath" value="/install/directory/path/openoffice4/program"
 type="java.lang.String" override="false" />
```

6.1.9.4 Server startup configuration settings

Create or edit a file named `setenv.sh` under the `$CATALINA_HOME/bin` directory and define startup environmental variables. Here is a sample setenv.sh file; replace all key values with appropriate values.

```
export CATALINA_OPTS="-Xms512m -Xmx1024m -XX:PermSize=512m -XX:MaxPermSize=512m
-Djavax.net.ssl.trustStore=/trust/store/certs/path/.truststore.aws
-Djavax.net.ssl.trustStorePassword=XXXXXXXX"
```

Please refer to the *Core Standards source repository* [R5] for more information.

6.1.10 Portal Deployment

This is the entry point where end-users access the components of the Smarter Balanced system. Through its integration with SSO, it provides the logged-in user access to only the components he/she has access to. Portal is a Wordpress-based application and as such its setup differs from most other SmarterApp components.
Please refer to the Portal source code repository [R12] for the latest installation details and documentation.

6.1.11 Portal Provisioning

- Install Wordpress on a Linux box (Ubuntu Linux 12.04+ recommended)
- Configure Wordpress according to the instructions provided in the Documentation folder.
- Log in to the WP admin interface as WP Admin and install the SAML 2.0 plugin code from this Portal repository, not the official saml-20-single-sign-on plugin.

6.1.11.1 Property Configuration

Configuring the Permissions application:

- Create an entry for every component under "Manage Components."
- Portal makes API calls to both *Program Management [R2]* and *Permissions [R3]*. Program Management consumes roles directly, and they are aligned with specific permissions (refer to [R2]). The role required for obtaining Program Management information is **Program Management Read**, which is aligned with the **Progman Read** permission.
- There is no need to assign new permissions to the Portal components.

Create Portal User in ART:

- Create a new user in ART [R9] for Portal's backend processing. The user must have the **Program Management Read** role, assigned at the highest available hierarchy level.

Configuring the Program Management application:

- Create an entry for each component being served by Portal. The "component names" in Program Management must exactly match the component names in Permissions application.
- Create a configuration for Portal containing a set of properties. Program Management properties need to be set for proper functioning of the logged-in portion of the Portal. Sample Portal properties may be found at [config/portal-progman-config.txt]. What follows is an explanation of each Program Management key/value pair.

`iconRelativePath=wp-content/themes/smarterbalanced/images/component_icons/` - Local WP relative path of SmarterApp icon location for display on the logged-in Portal page. For each component defined in the SmarterApp ecosystem, the following four key/value pairs must be defined:

```
COMPONENT NAME=component
COMPONENTNAME.displayname=COMPONENT NAME
COMPONENTNAME.url=https://COMPONENT.URL
COMPONENTNAME.icon=COMPONENT.png
```

As a working example, this is how the Core Standards component would be set up.

- `Core Standards=component` - Defines a component named "Core Standards." This component name, stripped of any spaces, becomes the identifier for the remaining three keys.
- `CoreStandards.displayname=Core Standards` - Defines the name to use in Portal when displaying the icon for this component. The display name does not have to exactly match the component name defined above.
- `CoreStandards.url=https://cs.url/` - Defines the URL for the component itself; when logged-in users click here, they are redirected to this URL.
- `CoreStandards.icon=core_standards.png` - Defines the icon name for this component. The icon must be located on the local filen system, at the location defined in `iconRelativePath`.
- Repeat the above for each component that needs to be accessible from Portal.

Now Configure SSO/SAML on the Portal server

- Go to `/var/www/wordpress/wp-content/plugins` and rename `saml-20-single-sign-on` to, say, `saml-20-single-sign-on-`
- Log into the portal as WP Admin
- Go to `/var/www/wordpress/wp-content/plugins and rename saml-20-single-sign-on-` back to `saml-20-single-sign-on`
- In the WP admin console, go to Settings -> Single Sign-On, and navigate to the Identity Provider tab.
- In the first URL box (IdP URL) enter the IdP config URL (e.g. https://OAM.URL/auth/saml2/jsp/exportmetadata.jsp?realm=/your_realm) and click "Fetch Metadata," then Update Options if all looks correct.
- Log in to the associated OpenAM instance, go to Federation tab, and add Entity. Enter the WP SP URL (e.g. `http://portal.your.portal.domain/wp-content/plugins/saml-20-single-sign-on/saml/www/module.php/saml/sp/metadata.php/1`), and save.
- Click on the realm (e.g., sbac) and add the newly added SP to the Circle of Trust.
- Provision a new user in your SSO system using the Administration and Registration Tools (ART) component named portal.agent@example.com, with a Role of Portal Agent at the Client level.

Configuring the Portal Backend Service

Several services need to run on Portal in order to integrate with SSO, Program Management, and Permissions.

- Copy `dump_component_mapping.pl` to `/usr/local/bin/dump_component_mapping.pl`
- Copy `dump_component_mapping.sh` to `/usr/local/bin/dump_component_mapping.sh`
- Install a crontab for root to execute `/usr/local/bin/dump_component_mapping.sh` periodically - for example, once or twice daily.

6.2 Assessment Creation and Management Components

These components manage the process and workflow of the creation and life cycle of Assessment Artifacts. See the chart below for the components associated with this subsection. Note that *Test Packager* functionality has been combined with the *Test Authoring* component. See [4] for more information on dependencies.

Deployed Component → / Dependent component ↓	Test Authoring	Test Spec Bank	Test Item Bank
Single Sign On (SSO)	Y	Y	
Program Management (PM)	Y	Y	Y
Permissions	Y	Y	
Monitoring & Alerting (MNA)	optional	optional	optional
Core Standards (CS)	Y		
Test Authoring	n/a	Y	
Test Spec Bank (TSB)	Y	n/a	
Test Item Bank (TIB)	Y	Y	n/a
FTP server		Y	Y
Database server	MongoDB	MongoDB	MongoDB

Table 2: Assessment Creation and Management Component Dependencies

6.2.1 Test Authoring Deployment

This component is a graphical interface used for creating test blueprints and specifications and manage the workflow. It will interact with the Test Item Bank component and the Test Spec Bank component. This component includes Test Packaging functionality, which prepares the test items and the test specifications for use by the test delivery system. Test packager pre-processes assessment assets to make them more efficient for the Test Delivery component. The packager creates the assessment instrument that a Test Delivery system can consume and use to deliver the assessment.

6.2.2 Test Authoring Provisioning.

Please refer to the *Configuration/Provisioning Guide* section in this document to set up a new generic Tomcat server.

6.2.2.1 Property Configuration

Configuration settings for this component are defined via the Program Management application. Add all configuration settings based on the default set provided in the code repository [R6]. Please refer there for the latest information on setup and configuration.

- `testauth.item.count.max.limit=40000` - Maximum items per assessment
- `component.name=TestAuthoring` - Component name must match name in Permissions and Program Management
- `permission.uri=http://name.of.permissions.server/rest` - URI of the Permissions application's REST endpoint
- `testauth.security.idp=http://name.of.identity.provider/auth/saml2/jsp/exportmetadata.jsp?metalias=idp&realm=sbac` - SSO Identity Provider metadata URL
- `testauth.mna.description=The Test Authoring Component` - Name of this component, as shown within Monitoring and Alerting
- `testauth.mna.healthMetricIntervalInSeconds=120` - Periodic health metrics for MNA
- `mna.mnaUrl=http://name.of.mna.server/rest/` - URL of MNA REST endpoint
- `mna.logger.level=[OFF | ERROR | WARN | INFO | DEBUG | TRACE | ALL](default:ERROR)` - MNA logging level
- `mna.clean.days=30 (default)` - How long to keep logs before cleanup
- `mna.clean.cron=0 0 0 * * ? (default)` - MNA cron job for cleanup
- `mna.oauth.batch.account=` - Username (email address) of MNA client user used for authenticating into MNA and logging metrics information
- `mna.oauth.batch.password=` - Password of MNA client user
- `testauth.mna.availability.metric.email=abc@example.org` - email to send availability metrics
- `testauth.mongo.hostname=` - Mongo DB Host name
- `testauth.mongo.port=27017` - Mongo DB port
- `testauth.mongo.username=` - Mongo DB usename
- `testauth.mongo.password=` - Mongo DB password
- `testauth.mongo.dbname=testauth-dev` - Mongo DB database name
- `testauth.minJs=true` - Use minJs?
- `testauth.rest.context.root=/rest/` - REST endpoint of Test Authoring
- `testauth.dtd.url=http://xxxxx/rest/resources/dtd/testpackage_v_9_19_2013.dtd` - Test Package DTD path
- `tsb.tsbUrl=http://name.of.test.spec.bank.server/rest/` - Test Spec Bank REST endpoint URL
- `testauth.core.standards.url=http://name.of.corestandards.server/api/` - CoreStandards API endpoint URL
- `tib.baseUri=http://name.of.test.item.bank.server/rest/` - Test Item Bank REST endpoint URL
- `testauth.security.profile=dev` -
- `testauth.languages=Afar|aar|Abkhazian|abk|Achinese|ace...` - Pipe-delimited list of supported languages in Test Authoring, e.g. Name|three-letter-official-abbreviation.

Clustered Environment Properties

These are *optional* properties which are used when configuring a clustered environment behind a load balancer (LB). To activate clustered environment support, simply change the active profile setenv as follows: change `spring.profiles.active` from `server.singleinstance` to `server.loadbalanced`. Furthermore, you will need to set these key/value pairs appropriately:

- `testauth.loadbalanced.url.scheme` - {this should be http or https}
- `testauth.loadbalanced.server.name` - {the loadbalancer's name}
- `testauth.loadbalanced.server.port` - {if your server requires a port, include it here, otherwise put 80 in as the default}
- `testauth.loadbalanced.includeServerPortInRequestURL` - {boolean true/false value which indicates if the port should be included to resolve the server}
- `testauth.loadbalanced.server.rest.contextpath` - {REST context name. e.g.: "/testauth.rest"}
- `testauth.loadbalanced.server.webapp.contextpath` - {webapp context name. e.g.: "/testauth.webapp". Leave this blank if you are using ROOT as webapp context name}

6.2.2.2 Database Configuration

Test Authoring uses a MongoDB database. Follow the steps in Section 5 of this document to create a generic MongoDB server. Test Authoring will create a database with all required collections upon startup if one doesn't already exist.

6.2.2.3 Server startup configuration settings

Create or edit a file named `setenv.sh` under the `$CATALINA_HOME/bin` directory and define startup environmental variables. Here is a sample setenv.sh file; replace all key values with appropriate values.

```
export CATALINA_OPTS="-Xms512m -Xmx1024m -XX:PermSize=512m -XX:MaxPermSize=512m
-Dspring.profiles.active=progman.client.impl.integration,mna.client.integration
-Dprogman.baseUri=https://pm.example.org/rest/
-Dprogman.locator='name,environment' -DmnaServerName=auth.example.org
-Djavax.net.ssl.trustStore=/trust/store/certs/path/.truststore.aws
-Djavax.net.ssl.trustStorePassword=XXXXXXXX"
```

Please refer to the *Test Authoring repository* [R6] for more information.

6.2.3 Test Spec Bank Deployment

Test Specification Bank (a.k.a. TSB or Test Spec Bank) is a repository for test specifications, blueprints and other data about tests such as the adaptive algorithm to be used during the test.

6.2.4 Test Spec Bank Provisioning

Please refer to the *Configuration/Provisioning Guide* section in this document to set up a new generic Tomcat server.

6.2.4.1 Property Configuration

Configuration settings for this component are defined via the Program Management application. Add all configuration settings based on the default set provided in the code repository [R7]. Please refer there for the latest information on setup and configuration.

- `tsb.security.idp=http://name.of.identity.provider.server/auth/saml2/jsp/exportmetadata.jsp?metalias=idp&realm=sbac` - URL of SSO IDP's metadata export file
- `permission.uri=http://name.of.permissions.server/rest` - URI to Permissions REST interface
- `component.name=TestSpecBank` - Must match the name in Permissions and Program Management applications
- `tsb.mna.description=The Test Spec Bank Component` - TSB description for MNA logs
- `mna.mnaUrl=http://name.of.mna.server/rest` - URL to MNA's REST interface
- `mna.logger.level=[OFF | ERROR | WARN | INFO | DEBUG | TRACE | ALL] (default:ERROR)` - Error log items of this level or higher are to be sent to the MNA application.
- `mna.clean.days=30 (default)` - Clear our MNA logs after this many days
- `mna.clean.cron=0 0 0 * * ? (default)` - Cron job entry for MNA log cleaning schedule
- `mna.oauth.batch.account` - Username (email address) of MNA client user used for authenticating into MNA and logging metrics information
- `mna.oauth.batch.password=` - Password of MNA client user
- `tsb.mongo.hostname=` - TSB Mongo DB hostname
- `tsb.mongo.port`=27017 - Mongo DB port
- `tsb.mongo.username=` - Mongo DB username
- `tsb.mongo.password=` - Mongo DB password
- `tsb.mongo.dbname=tsb-dev` - Mongo DB database name
- `tsb.dtd.url=http://name.of.test.authoring.server/rest/resources/dtd/testpackage_v_9_19_2013.dtd` - Test package DTD location
- `tsb.rest.context.root=/rest/` - Root context of the TSB REST service
- `tsb.minJs=true` - Whether to use minimized JavaScript in the browser, true to minify
- `tib.tibUrl=http://name.of.test.item.bank.server/` - URL to TIB server
- `tsb.sftp.host=` - TSB SFTP host
- `tsb.sftp.port=22` - TSB SFTP port
- `tsb.sftp.user=` - TSB SFTP account username
- `tsb.sftp.pass=` - TSB SFTP account password
- `tsb.sftp.dir=` - TSB SFTP account directory
- `tsb.tib.sftp.host=` - TSB's TIB SFTP hostname
- `tsb.tib.sftp.port=22` - TSB's TIB SFTP server port
- `tsb.tib.sftp.user=` - TSB's TIB SFTP account
- `tsb.tib.sftp.pass=` - TSB's TIB SFTP account password
- `tsb.download.directory=` - TSB's SFTP download directory
- `tsb.export.cron.trigger=0,30 * * * * ?` - Cron to configure export frequency of test specifications

Clustered Environment properties

These are *optional* properties which are used when configuring a clustered environment behind a load balancer (LB). To activate clustered environment support, simply change the active profile setenv as follows: change `spring.profiles.active` from `server.singleinstance` to `server.loadbalanced`. Furthermore, you will need to set these key/value pairs appropriately:

- `tsb.loadbalanced.url.scheme` - {this should be http or https}
- `tsb.loadbalanced.server.name` - {the loadbalancer's name}
- `tsb.loadbalanced.server.port` - {if your server requires a port, include it here, otherwise put 80 in as the default}
- `tsb.loadbalanced.includeServerPortInRequestURL` - {boolean true/false value which indicates if the port should be included to resolve the server}
- `tsb.loadbalanced.server.rest.contextpath` - {REST context name. e.g.: "/tsb.rest"}
- `tsb.loadbalanced.server.webapp.contextpath` - {webapp context name. e.g.: "/tsb.webapp". Leave this blank if you are using ROOT as webapp context name}

6.2.4.2 Database Configuration

Test Spec Bank uses a MongoDB database. Follow the steps in Section 5 of this document to create a generic MongoDB server. Test Authoring will create a database with all required collections upon startup if one doesn't already exist.

6.2.4.3 Server startup configuration settings

Create or edit a file named `setenv.sh` under the `$CATALINA_HOME/bin` directory and define startup environmental variables. Here is a sample setenv.sh file; replace all key values with appropriate values.

```
export CATALINA_OPTS=" -Xms512m -Xmx1024m -XX:PermSize=512m -XX:MaxPermSize=512m
-Dspring.profiles.active=progman.client.impl.integration,mna.client.integration
-Dprogman.baseUri=https://pm.example.org/rest/
-Dprogman.locator=Name,Environment -DmnaServerName=tsb.example.org
-Djavax.net.ssl.trustStore=/trust/store/certs/path/.truststore.aws
-Djavax.net.ssl.trustStorePassword=XXXXXXX"
```

Please refer to the *Test Spec Bank repository* [R7] for more information.

6.2.5 Test Item Bank Deployment

This component contains items that are in operational, field, or interim tests.

6.2.6 Test Item Bank Provisioning

Please refer to the *Configuration/Provisioning Guide* section in this document to set up a new generic Tomcat server.

6.2.6.1 Property Configuration

Configuration settings for this component are defined via the Program Management application. Add all configuration settings based on the default set provided in the code repository [R8]. Please refer there for the latest information on setup and configuration.

- `component.name=Test Item Bank` - Component name. Must match name in Permissions and Program Management.
- `mna.mnaUrl=http://name.of.mna.server/rest/` - URL of the MNA server's REST endpoint
- `mna.oauth.client.id=mna` - OAuth MNA client name
- `mna.oauth.client.secret=` - OAuth MNA client secret
- `mna.oauth.batch.account=` - Username (email address) of MNA client user used for authenticating into MNA and logging metrics information
- `mna.oauth.batch.password=` - Password of MNA client user
- `oauth.access.url=https://name.of.sso.server/auth/oauth2/access_token?realm=/sbac` - SSO OAuth service URL including realm
- `permission.uri=https://name.of.permissions.server/rest` - REST endpoint of Permissions server
- `tib.clean.files.age.days=100` - How many days to keep files in TIB's SFTP directory on the sftp server
- `tib.clean.files.cron.trigger=0 0 8 * * *` - Cron config for cleaning TIB files
- `tib.file.pathname=/usr/local/tomcat/uploads` - Location on the TIB server where TIB will place test items it FTPs from the FTP server.
- `tib.mna.description=The Test Item Bank Component` - Name of TIB component for display in MNA
- `tib.mongo.dbname=test` - TIB Mongo DB database name
- `tib.mongo.hostname=dns.name.of.mongodb.host` - DNS name of the Mongo DB host for TIB.
- `tib.mongo.password=` - TIB Mongo DB password
- `tib.mongo.port=27017` - TIB Mongo DB server port
- `tib.mongo.username=` - TIB Mongo DB username
- `tib.oauth.checktoken.endpoint=https://name.of.sso.server/auth/oauth2/tokeninfo?realm=/sbac` - SSO OAuth checktoken service URL including realm
- `tib.oauth.resource.client.id=` - OAuth TIB client ID
- `tib.oauth.resource.client.secret=` - OAuth TIB client secret
- `tib.sftp.hostname=name.of.sftp.server` - Name of SFTP server from which TIB reads items.
- `tib.sftp.import.directory=` - Import directory name in SFTP directory for TIB user
- `tib.sftp.password=` - TIB SFTP password
- `tib.sftp.port=22` - TIB SFTP port
- `tib.sftp.user=` - TIB SFTP username

6.2.6.2 Database Configuration

Test Item Bank (also known as TIB) uses a MongoDB database. Follow the steps in Section 5 of this document to create a generic MongoDB server. TIB will create a database with all required collections upon startup if one doesn’t already exist.

6.2.6.3 Server startup configuration settings

Create or edit a file named `setenv.sh` under the `$CATALINA_HOME/bin` directory and define startup environmental variables. Here is a sample setenv.sh file; replace all key values with appropriate values.

```
export CATALINA_OPTS=" -Xms512m -Xmx1024m -XX:PermSize=512m -XX:MaxPermSize=512m
-Dspring.profiles.active=progman.client.impl.integration,mna.client.integration
-Dprogman.baseUri=https://pm.example.org/rest/
-Dprogman.locator=Name,Environment -DmnaServerName=tib.example.org
-Djavax.net.ssl.trustStore=/trust/store/certs/path/.truststore.aws
-Djavax.net.ssl.trustStorePassword=XXXXXXXX"
```

Please refer to the *Test Item Bank repository* [R8] for more information.

6.3 Assessment Delivery Components

These components deliver the assessment to the students and gather the data and metadata about the assessment. See the chart below for the components associated with this subsection, and [4] for more details.

Deployed Component → **Dependent component ↓**	**Admin and Registration Tool**	**TDS / Student**	**TDS / Proctor**	**Item Selection Shell / Adaptive Engine**	**Machine / AI Scoring**	**Test Integration**
Single Sign On (SSO)	Y		Y			
Program Management (PM)	Y		Y			
Permissions	Y		Y			
Monitoring & Alerting (MNA)	optional		Y			
Admin and Registration Tool	n/a	Y	Y			
TDS / Student		n/a				
TDS / Proctor			n/a			
Item Selection Shell / Adaptive Engine				n/a		

Machine / AI Scoring					n/a	
Test Integration						n/a
Data Warehouse						Y
FTP server	Y					
Database server	MongoDB	MySQL	MySQL	MySQL	MySQL	MySQL

Table 3: Assessment Delivery Component Dependencies

6.3.1 Admin and Registration Tools (ART) Deployment

This component manages the capabilities and methods required for assessment scheduling, test windowing, room scheduling, proctor assignment, student assignment, and student identification methods. This component also registers the student(s) for assessments and allows for uploads of student information and their accessibility profile. It must also manage staff identification for managing assessment events.

6.3.2 Admin and Registration Tools (ART) Provisioning

Please refer to the *Configuration/Provisioning Guide* section in this document to set up a new generic Tomcat server.

6.3.2.1 Property Configuration

Configuration settings for this component are defined via the Program Management application. Add all configuration settings based on the default set provided in the code repository [R9]. Please refer there for the latest information on setup and configuration.

User Change / SSO SFTP Setup

The SSO integration periodically checks to see if any user changes have occurred. If so, it will generate an XML file which is then transferred via SFTP to an OpenAM server where it is consumed for user provisioning or maintenance.

There are a few properties that must be set up in Program Management to enable this integration.

- `testreg.user.export.frequency.milliseconds` - Periodicity of XML generation in milliseconds
- `testreg.sftp.port` - SFTP port of server to connect to
- `testreg.sftp.dir` - Directory on SFTP server to copy file to
- `testreg.sftp.pass` - Password for authentication
- `testreg.sftp.user` - User for authentication
- `testreg.sftp.host` - SFTP host to connect to
- `testreg.sso.filename.suffix` - Suffix to append to file when SFTP'd

Data Warehouse Integration Setup

The data warehouse integration is a periodic export of data to a data warehouse landing zone. It incorporates pulling data out of the ART database, compressing it, encrypting it, and then SFTPing the data. There are numerous items to setup in order for it to all work.

GPG Keys

All files sent to the data warehouse must be encrypted and signed using GPG. In order to do this, each ART deployment must generate its own GPG key pair.

GPG is usually pre-installed on most UNIX systems. If generating keys on a Mac system install GPG Suite [https://gpgtools.org/]. If generating keys on a Windows system GPG4Win [http://www.gpg4win.org/] can be used.

Follow the directions in the software on your system to generate a key pair. In order to generate a key pair on a UNIX system the following can be used:

```
gpg --gen-key
```

You will be asked to choose a type:

```
Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
  Your selection?
```

The default key will work fine.

The next question that is asked is to choose key size. The default key size is ok.

```
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
```

Now choose how long the key is valid for. For testing purposes it is ok to choose that it will never expire.

```
Requested keysize is 2048 bits
Please specify how long the key should be valid.
   0 = key does not expire
   <n>  = key expires in n days
   <n>w = key expires in n weeks
   <n>m = key expires in n months
   <n>y = key expires in n years
 Key is valid for? (0)
```

A name and email address is then requested in order to build a user id to identify the key. For testing purposes these can be fake values.

You need a user ID to identify your key; the software constructs the user ID from the Real Name, Comment and Email Address in this form: "Heinrich Heine (Der Dichter) heinrichh@duesseldorf.de"

```
Real name: Test Tester
Email address: test@test.com
Comment: This is a test account
You selected this USER-ID:
    "Test Tester (This is a test account) <test@test.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit?
```

Choose (O)kay to continue.

Next choose a passphrase to protect your secret key.

```
You need a Passphrase to protect your secret key.

Enter passphrase:
```

After this, GPG needs to gather entropy in order to generate a more random key. On a system with a GUI and mouse entropy is easy to generate just by moving the mouse around the screen. If logged in remotely to a UNIX system, generating entropy will be a little bit tougher. One way to generate the needed entropy is to open up another SSH session to the remote server and running one or both of the following:

```
ls -R /
```

```
find / -type f
```

After the key generates GPG will output something like this:

```
gpg: key 00168441 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   2048R/00168441 2014-03-05
      Key fingerprint = 573D 9B85 618A 1FA8 C61A  B07A 4D57 69AE 0016 8441
uid                  Test Tester (This is a test account) <test@test.com>
sub   2048R/F288019A 2014-03-05
```

The key pair is now created.

The public key that was created needs to be exported so that it can be given to the data warehouse. The export can be done by:

```
gpg --export -a "test@test.com" > public.key
```

The exported key is an ASCII file.

The Data Warehouse should send a public key to ART. This key should be imported into the keychain on the ART server.

```
gpg --import dw.public.key
```

SSH Key

An SSH key pair will need to be generated for SFTP authentication. It is easiest to generate the SSH key on the server that the SFTP will take place on.

To generate a key pair log in to the remote server and change the user to the correct account that will be SFTPing.

Now use:

```
ssh-keygen
```

SSH-keygen will ask for a file to save the key in.

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
```

The default is usually OK unless the user account has multiple ssh keys.

Now ssh-keygen asks for a passphrase. Usually a passphrase is strongly suggested. However, in the case that the application is performing a system to system automated file transfer it is probably ok to just press enter to not have a passphrase.

```
Enter passphrase (empty for no passphrase):
```

You'll see some information about the key and it's now been generated.

```
Your identification has been saved in /home/ubuntu/.ssh/id_rsa.
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub.
The key fingerprint is:
1a:3c:74:09:65:39:8a:b1:c2:48:2e:a3:6b:8d:80:aa ubuntu@jenkinsmaster01
The key's randomart image is:
+--[ RSA 2048]----+
|      ..o.       |
| .  .  oo.       |
|oo   +..o.       |
|+.o oo..         |
|+. .  + S        |
|+      +         |
|o.o   .          |
|oo .             |
|E                |
+-----------------+
```

The public key that was generated needs to be given to the data warehouse so that they can correctly setup security for the landing zone that ART will SFTP to. The path of the generated public key is displayed above.

Password File

A property file needs to be created that contains the passphrase for the secret GPG key that was generated above. This file should contain one line like:

```
testreg.secret.passphrase=testkey
```

The value of that passphrase should be whatever was typed in when the GPG key was generated. Place this file somewhere in the file system that is secure. The only system user that should have access to this file is the user that runs Tomcat.

Tomcat Parameter

Tomcat needs to know where the secret passphrase file is. In order to tell Tomcat about it, a -D parameter needs to be added to the Tomcat command line at startup. It should look something like:

```
-Dtestreg.secret.passphrase.file=/pathto/testregkeypass.properties
```

The parameter name must be testreg.secret.passphrase.file.

Program Management Properties

There are number of other properties that need to be added to Program Management in order to successfully run the data warehouse integration.

- gpgKeyring.public.location - location of GPG public keyring, usually found in .gnupg/pubring.gpg
- gpgKeyring.secret.location - location of GPG secret keyring, usually found in .gnupg/secring.gpg
- testreg.secret.key.userid - email address for test reg key
- landingzone.public.key.userid - email address for landing zone key
- dw.export.callback.url - callback URL for Data Warehouse
- dw.host - SFTP host for landing zone
- dw.private.key.loc - file path of private key for authenticating to landing zone
- dw.port - SFTP port
- dw.user - user to connect to landing zone as
- dw.remote.dir - remote SFTP directory to write to
- dw.gpgfile.prefix - file name prefix for generated file

Clustered Environment properties

These are *optional* properties which are used when configuring a clustered environment behind a load balancer (LB). To activate clustered environment support, simply change the

active profile setenv as follows: change `spring.profiles.active` from `server.singleinstance` to `server.loadbalanced`. Furthermore, you will need to set these key/value pairs appropriately:

- `testreg.loadbalanced.url.scheme` - {this should be http or https}
- `testreg.loadbalanced.server.name` - {the loadbalancer's name}
- `testreg.loadbalanced.server.port` - {if your server requires a port, include it here, otherwise put 80 in as the default}
- `testreg.loadbalanced.includeServerPortInRequestURL` - {boolean true/false value which indicates if the port should be included to resolve the server}
- `testreg.loadbalanced.server.rest.contextpath` - {REST context name. e.g.: "/testreg.rest"}
- `testreg.loadbalanced.server.webapp.contextpath` - {webapp context name. e.g.: "/testreg.webapp". Leave this blank if you are using ROOT as webapp context name}

6.3.2.2 Database Configuration

Administration and Registration Tools (also known as ART) uses a MongoDB database. Follow the steps in Section 5 of this document to create a generic MongoDB server. ART will create a database with all required collections upon startup if one doesn't already exist.

6.3.2.3 Server startup configuration settings

Create or edit a file named `setenv.sh` under the `$CATALINA_HOME/bin` directory and define startup environmental variables. Here is a sample setenv.sh file; replace all key values with appropriate values.

```
export CATALINA_OPTS="-Xms512m -Xmx1024m -XX:PermSize=512m -XX:MaxPermSize=512m
-Dspring.profiles.active=progman.client.impl.integration,mna.client.integration
-Dprogman.baseUri=https://pm.example.org/rest/
-Dprogman.locator=Name,Environment -DmnaServerName=tib.example.org
-Djavax.net.ssl.trustStore=/trust/store/certs/path/.truststore.aws
-Djavax.net.ssl.trustStorePassword=XXXXXXXX"
```

6.3.2.4 Application Bootstrap

Application Settings

In the running ART application there is a link at the top of the screen called Setting. There are two settings on that page that need to be set in order for the Data Warehouse integration to work correctly.

The first is `Test Reg System Id`. If nothing is set in the text field a button should be displayed next to the text field that will generate a random GUID to use as the system ID. Press the button to generate the GUID.

The second setting is `Assessment Eligibility types for export to Data Warehouse`. This is a multi-select list that allows for the setup of only specific types of assessments to control the Data Warehouse export. Select the types of assessments that should influence the export: summative, formative, and/or interim.

Be sure to save any settings changes that are made.

Field Name Crosswalk Setup

Each field that is part of an upload file has an associated field name that is displayed in various locations on the UI. It is possible to change the display names of the fields.

The initial mapping of the field names is kept in the REST module in the `/src/main/resources/spring/entity-crosswalk-context.xml` file. This can be modified and the WAR container restarted to pickup changes.

Organizational Hierarchy Renaming (Entity)

The application allows for the renaming of the levels of the organizational (entity) hierarchy. For example, the lowest level of the hierarchy is an institution. Many deployments may wish to change this to always display "school" rather than "institution."

The renaming of the levels in the hierarchy is part of component branding in the program management component. Two steps are necessary: adding a tenant and then configuring the branding for the tenant and component.

If the deployment of ART is a state level deployment, all users who log into the system should ultimately have tenancy that resolves to the state for which the system has been deployed. If, however, the ART deployment truly is multi-tenanted, then this configuration must be performed for each tenant. Component branding is resolved from lower to higher in the tenant hierarchy.

Add a Tenant

In the Program Management application, choose Manage Tenants to create a tenant if it has not been created. Make sure that the tenant has a subscription to the ART component. If an ART component has not yet been created go to Manage Components to create one.

Configure Component Branding

To set up the names go to Configure Component Branding. All of the entries are of type `Property`. The list of properties to configure are:

- Assessment
- Assessments
- User
- Users
- Institution
- GroupOfDistricts
- GroupOfInstitutions
- GroupOfStates

- Student
- Students
- Institutions
- StudentGroups
- District
- State
- Accommodations
- Eligibility
- Student Group

For each property the value reflects what will be seen in the UI.

Other Miscellaneous Setup

In the ART UI there is a link at the top of the screen called Setting. The settings page has a few items that should be configured before users use the system:

- `Client Name` - The name of the top level Client for this deployment. This is bootstrapped from Program Management properties, but can be changed here.
- `Time Zone` - The effective time zone for this deployment. Choose from the drop down list.
- `Share Student Identity Data` - This checkbox turns on and off the choice to share student identity data. If selected, then student names, birth dates, and SSID will be shared out through external interfaces. If turned off, then this data will not be shared.
- `Hide 'Group Of' Entity Levels` - This must be configured before the system is used. Allows the system to completely hide the "Group of" levels of the organizational hierarchy. Once hidden, the system will not allow any functions to be performed for those levels of the hierarchy.

Other Dependencies and Progman Properties

There are many other properties that need to be set in Program Management so that ART will function correctly.

MNA Properties

- `testreg.mna.description` - Component name string to pass to M&A
- `mna.mnaUrl` - The URL of the M&A REST api (/rest/)
- `mna.oauth.batch.account` - Username (email address) of MNA client user used for authenticating into MNA and logging metrics information
- `mna.oauth.batch.password` - Password of MNA client user

RabbitMq Properties

- `rabbitmq.vhost` - The name of the RabbitMq ghost to connect to. The default vhost is "/", but RabbitMq could be configured to use something different.
- `rabbitmq.password` - Password to authenticate to RabbitMq with
- `rabbitmq.username` - Username to connect to RabbitMq as
- `rabbitmq.host` - RabbitMq server hostname

Security/SSO Properties

- `testreg.security.idp` - The URL of the SAML-based identity provider (OpenAM)
- `testreg.security.profile` - The name of the environment the application is running in. For a production deployment this will most likely be "prod". Must match the profile name used to name metadata files.
- `component.name` - The name of the component that this ART deployment represents. This must match the name of the component in Program Management and the name of the component in the Permissions application
- `permission.uri` - The base URL of the REST api for the Permissions application

Mongo Properties

- `testreg.mongo.hostname` - A comma delimited list of Mongo hostnames
- `testreg.mongo.port` - The Mongo port
- `testreg.mongo.username` - Username to connect to Mongo as
- `testreg.mongo.password` - Password for Mongo authentication
- `testreg.mongo.dbname` - The database name for the ART database

Other Miscellaneous Properties

- `client` - The name of the top-level client entity for this deployment
- `tsb.tsbUrl` - The base URL for the REST api of Test Spec Bank
- `language.codes` - Comma delimited list of valid language codes
- `testreg.minJs` - Flag for JavaScript minification, true to minify
- `testreg.rest.context.root` - The server relative context root for the ART REST WAR. Should start and end with a slash.

6.3.3 Test Delivery System (TDS) Deployment (for both student and proctor)

Follow the steps outlined in Section 5 of this document (Server Creation) to create the TDS server.

6.3.3.1 Property Configuration - Proctor

Configuration settings for this component are defined via the Program Management application. Add all configuration settings based on the default set provided in the code repository [R10]. Please refer there for the latest information on setup and configuration.

Database Properties

The following parameters need to be configured inside program management for database.

- `datasource.url=jdbc:mysql://mysql.db.url:3306/schemaname` - The JDBC URL of the database from which Connections can and should be acquired.
- `datasource.username=<db-username>` - Username that will be used for the DataSource's default getConnection() method.
- `datasource.password=<db-password>` - Password that will be used for the DataSource's default getConnection() method.
- `datasource.driverClassName=com.mysql.jdbc.Driver` - The fully qualified class name of the JDBC driverClass that is expected to provide Connections.
- `datasource.minPoolSize=5` - Minimum number of Connections a pool will maintain at any given time.
- `datasource.acquireIncrement=5` - Determines how many connections at a time datasource will try to acquire when the pool is exhausted.
- `datasource.maxPoolSize=20` - Maximum number of Connections a pool will maintain at any given time.
- `datasource.checkoutTimeout=60000` - The number of milliseconds a client calling getConnection() will wait for a Connection to be checked-in or acquired when the pool is exhausted. Zero means wait indefinitely. Setting any positive value will cause the getConnection() call to timeout and break with an SQLException after the specified number of milliseconds.
- `datasource.maxConnectionAge=0` - Seconds, effectively a time to live. A Connection older than maxConnectionAge will be destroyed and purged from the pool. This differs from maxIdleTime in that it refers to absolute age. Even a Connection which has not had much idle time will be purged from the pool if it exceeds maxConnectionAge. Zero means no maximum absolute age is enforced.
- `datasource.acquireRetryAttempts=5` - Defines how many times datasource will try to acquire a new Connection from the database before giving up. If this value is less than or equal to zero, datasource will keep trying to fetch a Connection indefinitely.

MNA properties

The following parameters need to be configured inside program management for MNA.

- `mna.mnaUrl=http://<mna-context-url>/mna-rest/` - URL of the Monitoring and Alerting client server's rest url
- `mnaServerName=proctor_dev` - Used by the MNA clients to identify which server is sending the log/metrics/alerts.
- `mnaNodeName=dev` - Used by the mna clients to identify who is sending the log/metrics/alerts. There is a discrete mnaServerName and a node in case say XXX for server name & node1/node2 in a clustered environment giving the ability to search

across clustered nodes by server name or specifically for a given node. It's being stored in the db for metric/log/alert, but not displayed.
- mna.logger.level=ERROR - Used to control what is logged to the Monitoring and Alerting system. Logging Levels (ALL - Turn on all logging levels, TRACE, DEBUG, INFO, WARN, ERROR, OFF - Turn off logging).

SSO properties

The following parameters need to be configured inside program management for SSO.

- permission.uri=https://<permission-app-context-url>/rest - The base URL of the REST api for the Permissions application.
- component.name=Proctor - The name of the component that this Proctor deployment represents. This must match the name of the component in Program Management and the Permissions application.
- proctor.security.idp=https://<idp-url> - The URL of the SAML-based identity provider (OpenAM).
- proctor.webapp.saml.metadata.filename=proctor_sp.xml - OpenAM Metadata file name uploaded for environment and placed inside server directory.
- proctor.security.dir=file:////<sp-file-location-folder> - Location of the metadata file.
- proctor.security.saml.keystore.cert=<cert-name> - Name of the Keystore cert being used.
- proctor.security.saml.keystore.pass=<password> - Password for keystore cert.
- proctor.security.saml.alias=proctor_webapp - Alias for identifying web application.
- oauth.tsb.client=tsb - OAuth Client id configured in OAM to allow the SAML bearer workflow to convert a SAML assertion into an OAuth token for the "coordinated web service" call to TSB.
- oauth.access.url=https://<oauth-url> - OAuth URL to OAM to allow the SAML bearer workflow to POST to get an OAuth token for any "machine to machine" calls requiring OAUTH
- encrypt:oauth.tsb.client.secret=<password> - OAuth Client secret/password configured in OAM (under the client id) to allow the SAML bearer workflow to convert a SAML assertion into an OAuth token for the "coordinated web service" call to TSB.
- encrypt:mna.oauth.client.secret=<password> - OAuth Client secret/password configured in OAM to allow get an OAuth token for the "batch" web service call to MnA.
- mna.oauth.client.id=mna - OAuth Client id configured in OAM to allow get an OAuth token for the "batch" web service call to MnA.
- proctor.oauth.resource.client.secret=<password> - OAuth Client secret/password configured in OAM to allow get an OAuth token for the "batch" web service call to core standards.
- proctor.oauth.resource.client.id=proctor - OAuth Client id configured in OAM to allow get an OAuth token for the "batch" web service call to core standards.
- proctor.oauth.checktoken.endpoint=http://<oauth-url> - OAuth URL to OAM to allow the SAML bearer workflow to perform a GET to check that an OAuth token is valid.

Proctor properties

The following parameters need to be configured inside program management for Proctor

- proctor.IsCheckinSite=false
- proctor.ClientQueryString=false
- proctor.Appkey=Proctor
- proctor.EncryptedPassword=true
- proctor.RecordSystemClient=true
- proctor.SqlCommandTimeout=60
- proctor.AppName=Proctor
- proctor.SessionType=0 - Type of the testing supported: 0 is online, 1 is paper-based.
- proctor.DBJndiName=java:/comp/env/jdbc/sessiondb
- proctor.TestRegistrationApplicationUrl=http://url.to.art/ - URL to ART Application
- proctor.TDSArchiveDBName=archive - Name of the archive schema
- proctor.TDSSessionDBName=session - Name of the session schema
- proctor.TDSConfigsDBName=configs - Name of the config schema
- proctor.ItembankDBName=itembank - Name of the itembank schema
- proctor.Debug.AllowFTP=true
- proctor.StateCode=SBAC_PT
- proctor.ClientName=SBAC_PT

6.3.3.2 Property Configuration - Student

Configuration settings for this component are defined via the Program Management application. Add all configuration settings based on the default set provided in the code repository [R11]. Please refer there for the latest information on setup and configuration.

Database Properties

Following parameters need to be configured inside program management for database.

- datasource.url=jdbc:mysql://localhost:3306/schemaname - The JDBC URL of the database from which Connections can and should be acquired.
- datasource.username=<db-username> - Username that will be used for the DataSource's default getConnection() method.
- encrypt:datasource.password=<db-password> - Password that will be used for the DataSource's default getConnection() method.
- datasource.driverClassName=com.mysql.jdbc.Driver - The fully-qualified class name of the JDBC driverClass that is expected to provide Connections
- datasource.minPoolSize=5 - Minimum number of Connections a pool will maintain at any given time.
- datasource.acquireIncrement=5 - Determines how many connections at a time datasource will try to acquire when the pool is exhausted.
- datasource.maxPoolSize=20 - Maximum number of Connections a pool will maintain at any given time.

- datasource.checkoutTimeout=60000 - The number of milliseconds a client calling getConnection() will wait for a Connection to be checked-in or acquired when the pool is exhausted. Zero means wait indefinitely. Setting any positive value will cause the getConnection() call to timeout and break with an SQLException after the specified number of milliseconds.
- datasource.maxConnectionAge=0 - Seconds, effectively a time to live. A Connection older than maxConnectionAge will be destroyed and purged from the pool. This differs from maxIdleTime in that it refers to absolute age. Even a Connection which has not been much idle will be purged from the pool if it exceeds maxConnectionAge. Zero means no maximum absolute age is enforced.
- datasource.acquireRetryAttempts=5 - Defines how many times datasource will try to acquire a new Connection from the database before giving up. If this value is less than or equal to zero, datasource will keep trying to fetch a Connection indefinitely.

MNA properties

The following parameters need to be configured inside program management for MNA.

- mna.mnaUrl=http://<mna-context-url>/mna-rest/ - URL of the Monitoring and Alerting client server's rest URL.
- mnaServerName=student_dev - Used by the mna clients to identify which server is sending the log/metrics/alerts.
- mnaNodeName=dev - Used by the mna clients to identify who is sending the log/metrics/alerts. There is a discrete mnaServerName and a node in case say XXX for server name & node1/node2 in a clustered environment giving the ability to search across clustered nodes by server name or specifically for a given node. It’s being stored in the db for metric/log/alert, but not displayed.
- mna.logger.level=ERROR - Used to control what is logged to the Monitoring and Alerting system. The logging levels are ALL - Turn on all logging levels, TRACE, DEBUG, INFO, WARN, ERROR, OFF - Turn off logging.

Student properties

The following parameters need to be configured inside Program Management for Student.

- student.IsCheckinSite=false
- student.DONOT_Distributed=true
- student.ClientName=SBAC_PT
- student.StateCode=SBAC_PT
- student.ClientQueryString=true
- student.ClientCookie=true - If it is turned on, Student will try to get clientname from cookie
- student.Appkey=Student
- student.EncryptedPassword=true
- student.RecordSystemClient=true
- student.AdminPassword=<password> - Admin Password
- student.AppName=Student - Application name
- student.SessionType=0 - Type of the testing supported: 0 is online, 1 is paper-based
- student.DBDialect=MYSQL - Indicates which database we are using
- student.TestRegistrationApplicationUrl=http://localhost:8083/ - URL to TR(ART) Application

- student.TDSArchiveDBName=archive - Name of the archive schema
- student.TDSSessionDBName=session - Name of the session schema
- student.TDSConfigsDBName=configs - Name of the config schema
- student.ItembankDBName=itembank - Name of the itembank schema
- student.Debug.AllowFTP=true
- student.TDSReportsRootDirectory=/usr/local/tomcat/resources/tds/tdsreports/ - Directory on Student server box where TDS reports generated after student finished the test are located.
- student.StudentMaxOpps=2

6.3.3.3 Database Configuration

The TDS student and proctor applications use the same database. For production systems it is recommended to use large clustered instances such as Amazon's RDS. On the database machine, the following settings must be made.

_key	name	description	homepath
25	SBAC	(NULL)	/tmp/1/
26	SBAC_PT	(NULL)	/path/to/resources/tds/

Create Default Databases

The following steps will create the DBs and all necessary objects within. There are also drop scripts: drop_constraints.sql and drop_tables.sql should there be need to drop all the tables and start over. All scripts mentioned below are located at tdsdlldev project, tds-dll-schemas module.

'configs' Database

- Run the following command on the db server: `CREATE DATABASE 'configs' DEFAULT CHARACTER SET=utf8`
- Create tables by running the SQL script file located at tds-dll-schemas/src/main/resources/sql/MYSQL/configs/create_tables.sql
- Create constraints by running the SQL script file create_tables.sql located at tds-dll-schemas/src/main/resources/sql/MYSQL/configs/create_constraints.sql
- Create indexes by running the SQL script file located at tds-dll-schemas/src/main/resources/sql/MYSQL/configs/create_indexes.sql
- Create stored procedures by running the SQL script file/files located in folder tds-dll-schemas/src/main/resources/sql/MYSQL/configs/StoredProcedures/
- Create functions by running the SQL script file/files located in folder tds-dll-schemas/src/main/resources/sql/MYSQL/configs/Functions/

'itembank' Database

- Run the following command on the db server: `CREATE DATABASE 'itembank' DEFAULT CHARACTER SET=utf8`
- Create tables by running the SQL script file located at tds-dll-schemas/src/main/resources/sql/MYSQL/itembank/create_tables.sql
- Create constraints by running the SQL script file create_tables.sql located at tds-dll-schemas/src/main/resources/sql/MYSQL/itembank/create_constraints.sql
- Create indexes by running the SQL script file located at tds-dll-schemas/src/main/resources/sql/MYSQL/itembank/create_indexes.sql
- Create triggers by running the SQL script file located at tds-dll-schemas/src/main/resources/sql/MYSQL/itembank/Triggers/triggers.sql
- Create stored procedures by running the SQL script file/files located in folder tds-dll-schemas/src/main/resources/sql/MYSQL/itembank/StoredProcedures/
- Create functions by running the SQL script file/files located in folder tds-dll-schemas/src/main/resources/sql/MYSQL/itembank/Functions/

'session' Database

- Run the following command on the db server: `CREATE DATABASE 'session' DEFAULT CHARACTER SET=utf8`
- Create tables by running the SQL script file located at tds-dll-schemas/src/main/resources/sql/MYSQL/session/create_tables.sql
- Create constraints by running the SQL script file create_tables.sql located at tds-dll-schemas/src/main/resources/sql/MYSQL/session/create_constraints.sql
- Create indexes by running the SQL script file located at tds-dll-schemas/src/main/resources/sql/MYSQL/session/create_indexes.sql
- Create triggers by running the SQL script file located at tds-dll-schemas/src/main/resources/sql/MYSQL/session/Triggers/triggers.sql
- Create views by running the SQL script file/files located in folder tds-dll-schemas/src/main/resources/sql/MYSQL/session/Views/
- Create stored procedures by running the SQL script file/files located in folder tds-dll-schemas/src/main/resources/sql/MYSQL/session/StoredProcedures/
- Create functions by running the SQL script file/files located in folder tds-dll-schemas/src/main/resources/sql/MYSQL/session/Functions/

'archive' Database

- Run the following command on the db server: `CREATE DATABASE 'archive' DEFAULT CHARACTER SET=utf8`
- Create tables by running the SQL script file located at tds-dll-schemas/src/main/resources/sql/MYSQL/archive/create_tables.sql
- Create indexes by running the SQL script file located at tds-dll-schemas/src/main/resources/sql/MYSQL/archive/create_indexes.sql

Load Configuration Data

1. Run the below three files on `configs` db to setup generic configs db:
   `/tds-dll-schemas/src/main/resources/import/genericsbacconfig/gen1.sql`
   `/tds-dll-schemas/src/main/resources/import/genericsbacconfig/gen2.sql`
   `/tds-dll-schemas/src/main/resources/import/genericsbacconfig/gen3.sql`
2. Run these statements on `itembank` db to setup generic itembank db:

```
INSERT INTO `itembank`.`tblclient`
(`name`,
`description`,
`homepath`)
VALUES
('SBAC_PT',
NULL,
/*IMP: place root directory for the items path here */);
b. INSERT INTO `itembank`.`tblitembank`
(`_fk_client`,
`homepath`,
`itempath`,
`stimulipath`,
`name`,
`_efk_itembank`,
`_key`,
`contract`)
VALUES
(1, /*should be equal to _key column from tblclient tbl for this client name */
/*IMP: place relative home path for this itembank here */,
/*IMP: place the item path here */,
/*IMP: place the stimuli path here */
NULL,
1,
1,
NULL);
```

Example:

```
Insert into `itembank`.`tblclient` (`name`, `description`,`homepath`)
values ('SBAC', null, '/usr/local/tomcat/resources/tds/');
insert into INTO
`itembank`.`tblitembank`(`_fk_client`,`homepath`,`itempath`,`stimulipath`,`name`,`_efk_itemba
nk`,`_key`,`contract`)
values (1, 'bank/', 'items/', 'stimuli/', null, 1, 1, null);
```

Load Test Package into the Database

Execute/Run stored procedure `loader_main()` in database 'itembank' once per testpackage file. The stored proc takes one input, that is the XML file content. Copy the XML file content and paste it as an input and execute the stored procedure. Sample test packages are available in the **assessmentpackages** project. For example:

```
Call `itembank`.`loader_main` (`<testpackage purpose="administration" publisher="SBAC_PT" publishdate="Aug 15 2014  9:19AM" version="1.0">.. .. .</testpackage>`)
```

6.3.3.4 Server startup configuration settings

Create or edit a file named `setenv.sh` under the `$CATALINA_HOME/bin` directory and define startup environmental variables. Here is a sample setenv.sh file; replace all key values with appropriate values. These Tomcat/JVM values have been shown to be effective in a high performance scenario; modify if needed:

```
# TDS JVM config For Higher Performance
JAVA_OPTS="$JAVA_OPTS -server"
JAVA_OPTS="$JAVA_OPTS -Xms1080m"
JAVA_OPTS="$JAVA_OPTS -Xmx804m"
JAVA_OPTS="$JAVA_OPTS -XX:NewSize=484m"
JAVA_OPTS="$JAVA_OPTS -XX:MaxNewSize=512m"
JAVA_OPTS="$JAVA_OPTS -XX:PermSize=512m"
JAVA_OPTS="$JAVA_OPTS -XX:MaxPermSize=512m"
JAVA_OPTS="$JAVA_OPTS -Dsun.io.useCanonCaches=false"
JAVA_OPTS="$JAVA_OPTS -Dsun.net.client.defaultConnectTimeout=40000"
JAVA_OPTS="$JAVA_OPTS -Dsun.net.client.defaultReadTimeout=60000"

# Garbage Collection
JAVA_OPTS="$JAVA_OPTS -verbose:gc"
JAVA_OPTS="$JAVA_OPTS -Xloggc:/usr/local/tomcat/logs/gc.log"
JAVA_OPTS="$JAVA_OPTS -XX:+PrintClassHistogram"
JAVA_OPTS="$JAVA_OPTS -XX:+PrintGCTimeStamps"
JAVA_OPTS="$JAVA_OPTS -XX:+PrintGCDetails"
JAVA_OPTS="$JAVA_OPTS -XX:+UseParNewGC"
JAVA_OPTS="$JAVA_OPTS -XX:+UseConcMarkSweepGC"
JAVA_OPTS="$JAVA_OPTS -XX:+UseCMSCompactAtFullCollection"
JAVA_OPTS="$JAVA_OPTS -XX:+CMSClassUnloadingEnabled"
JAVA_OPTS="$JAVA_OPTS -XX:+DisableExplicitGC"

# Heap Dumps
JAVA_OPTS="$JAVA_OPTS -XX:+HeapDumpOnOutOfMemoryError"
JAVA_OPTS="$JAVA_OPTS
-XX:HeapDumpPath=/usr/local/apache-tomcat-7.0.41/logs/heapdump.hdprof"
```

Note:

Xms and Xmx sizing should be at least, a one third of the the server's RAM size but not more than 3/4 of the server's RAM size. Assigning a heap/RAM size below 500m to Xms/Xms may not work

very well with this web application. We recommended having at least a total of 1GB RAM on a server for this application.
XX:NewSize, NewRatio, MaxNewSize, PermSize, MaxPermSize should range between a minimum of 20% to a maximum of 50% of the total size of -Xms. Other Parameters can remain as shown.

TDS-specific configurations:

```
export CATALINA_OPTS="-DSB11_CONFIG_DIR=$CATALINA_HOME/resources
-Dorg.apache.commons.logging.Log=org.apache.commons.logging.impl.Log4JLogger
-Dlog4j.configuration=file:///path/to/tomcat/conf/log4j.xml
-Dspring.profiles.active=progman.client.impl.integration,mna.client.integration
-Dprogman.baseUri=http://url.to.progman.rest.interface/
-Dprogman.locator='Proctor,envname;Student,envname'
-DmnaServerName=name-of-this-tds-server
-Djavax.net.ssl.trustStore=/path/to/tomcat/ssh/.truststore.aws
-Djavax.net.ssl.trustStorePassword=XXXXXXXXXXXXXX"
```

- Run the Jenkins deploy script to push a version of student (or proctor) to this machine.

Please refer to the *Test Delivery repository* [R10] for more information.

6.3.4 Adaptive Engine Deployment

This component is an implementation of an adaptive algorithm. Similar to the AI Scoring Engine component, this component will have performance issues if implemented as a network service instead of a Test Delivery component plugin. The Test Spec Bank contains metadata that defines the algorithm or engine to use. The Test Delivery Component is expected to load the correct algorithm or engine when the test is being administered.
This component is automatically deployed when deploying the TDS system (Section 6.3.3.).

6.3.5 Machine / AI Scoring Deployment

Machine scoring is an integral part of the Test Delivery system and is automatically deployed when deploying the TDS system (Section 6.3.3.).

6.3.6 Test Integration Deployment

TBD

6.3.7 Test Integration Provisioning

TBD

6.4 Post-Deployment Configuration

6.4.1 Security

- X509 certificates will need to be installed on each load balancer to secure communications. References:
    - Updating a certificate on the load balancer.

- Using HTTP/HTTPS Protocol with Elastic Load Balancing.

7 Bootstrapping the System

The system consists of components which may be deployed at multiple tenancy levels (e.g., Consortium / top level, state level, etc.). As an integrated system, however, it is important to follow the following sequence when bootstrapping the system.

1. OpenAM or the selected SSO solution
2. Program Management
3. Monitoring and Alerting
4. Permissions
5. Test Item Bank and Core Standards
6. Portal and Digital Library

The remainder of the components may be deployed in any order once the shared services are available.

7.1 Creating the Prime User

While users are generally provisioned in ART, it's not possible to do so without logging into ART with a user - hence a chicken-and-egg problem. In order to provision a "Prime User" for a new system, an XML file must be generated and dropped into the SSO system's dropbox for consumption. A sample XML is available in the *Administrative* repository [R1] under the `pm` directory. An example is below:

```
<?xml version='1.0' encoding='UTF-8'?>
<Users>
<User Action="ADD">
  <UUID>CREATE-UNIQUE_UUID_HERE</UUID>
  <FirstName>CHOOSE-FIRST_NAME</FirstName>
  <LastName>CHOOSE-LAST_NAME</LastName>
  <Email>temp-bootstrap@example.com</Email>
  <Phone/>
  <Role>
    <RoleID></RoleID>
    <Name>Administrator</Name>
    <Level>CLIENT</Level>
    <ClientID>CHOOSE-CLIENT_IDENTIFIER_NUMBER</ClientID>
    <Client>CHOOSE-UNIQUE_CLIENT_NAME</Client>
    <GroupOfStatesID/>
    <GroupOfStates/>
    <StateID/>
    <State/>
    <GroupOfDistrictsID/>
    <GroupOfDistricts/>
    <DistrictID/>
    <District/>
```

```
    <GroupOfInstitutionsID/>
    <GroupOfInstitutions/>
    <InstitutionID/>
    <Institution/>
  </Role>
</User>
</Users>
```

WARNING: The Prime User is a *temporary* administrative user that is pushed directly into SSO. Since the user is not created with ART, it cannot be managed within ART. The idea is to use the Prime User for all necessary logins until ART is up and running, then create a new user via ART with Administrator role. At that point, it is recommended that this Prime User be deleted for security reasons.

7.1.1 Field Explanations and Notes

- **UUID**: This must be a unique identifier within the SSO system. Generate a UUID using any available mechanism.
- **FirstName** and **LastName**: Select this user's first and last names. These fields are essentially irrelevant, but it's a good idea to use a descriptive name for the first and last names.
- **Email**: The email address does not have to be real, but does have to be unique. This serves as the user's username.
- **Role/Name**: This field must say "Administrator".
- **Role/Level**: This field must say "CLIENT".
- **Role/ClientId**: This should be a unique text string identifier for the client, usually numeric.
- **Role/Client**: This should be a unique text string identifier for the client name. Selection of this name is important down the road when performing ART configuration. *This name cannot be changed once the Prime User is inserted into SSO* and configuration of other applications has begun.

7.1.2 Inserting Prime User Into SSO

Once the file is created, name it `prime_user_testfile_.xml`. The `_testfile_` extension ensures that an email will not be sent to the user's email (since it is fake). Place this file into the SSO FTP dropbox site. You will need the URL, username, and password of the SFTP site in order to do this. The prime user's default password is *password1*.

7.1.3 Protected Roles

The Administrator role is usually considered a special ("protected") role ("with great power comes great responsibility"). It's possible to set and unset the "protected" status of any role via ART. It's important to note that only users with protected roles can even view those protected roles; therefore when initially setting up a system, it's more convenient to not set the protected status of any of the roles until the system has been properly configured and regular users are ready to be provisioned.

7.2 Creating the Entity Hierarchy

Users and Students must be associated to an entity in order to exist in the system. Table 7.1 below describes the entity hierarchy levels in the SmarterApp system. The “Group Of…” entities, shown in dotted lines, are optional.

While the prime user is associated with the top-level (Client) entity, users and students are usually associated with lower-level entities (Institutions and Districts, for example). The entity hierarchy levels are fixed as shown (other than the optional elements), but users and students can be added and managed using the ART application (see 6.3). When new users are added to the system, the entity to which they are associated must already exist. For example, if student Johnny Appleseed is to be added to School “Generic High”, an institution with that name must already exist, and be parented in some way to the Client entity. Usually this is done by creating the parent District, its parent State, and parenting that state to the Client.

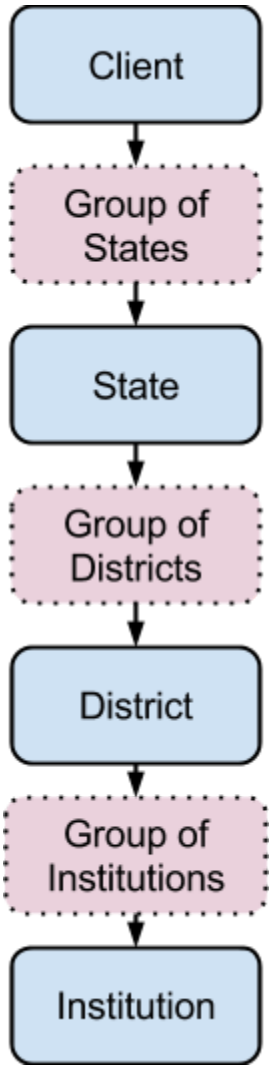


Table 7.1: SmarterApp Entity Level Hierarchy

7.2.1 The Tenancy Mechanism

Client components, such as TDS or ART, use a tenancy mechanism for authorization which depends on three main shared services: SSO, Program Management, and Permissions. This mechanism is used to manage User Attributes, Roles and Tenancy Chains, Roles to component Permissions mappings, and Tenants and Component Subscriptions.

Prior to a user logging on to a component such as Proctor, the component needs to know:

- **What roles matter to me, and what are their mappings to my permissions?**
    - This information is obtained via a REST call to Permissions. It might find out that users with *Test Administrator* roles have permissions to proctor summative tests, while users with *Interim Test Administrator* roles have permission to proctor only non-summative tests.
- **What tenants have a valid subscription to me?**
    - This information is obtained via a REST call to Program Management. It might find out that MN, ID, and CA have valid subscriptions to Proctor, but KS does not.

Next, after the user logs on, SSO returns the following information about the user:

- **Is this user authenticated?**
    - Is the username/password combination valid?
- **Does this user have any roles that matter to me?**
    - Role information is returned in the SAML response
- **Does this user belong to a valid tenant?**
    - Tenancy information is returned in the SAML response
- **Does the tenant have a current subscription to me?**
    - Tenant subscription information was obtained in the previous step from Program Management
- **What permissions does this user have?**
    - The user's roles are mapped against the permissions obtained earlier.
- **What entity at what level is this user's role(s) associated with?**
    - Role and Tenancy information from the user is mapped against the information obtained earlier.
- **What is the lineage of the entity?**
    - See Table 7.1.

Based on these answers, the component:

- Resolves role conflicts if necessary
- Determines associated entity and tenant
- Determines what features the user can access
- Determines what data to show the user
- Optionally brand itself with component assets

7.3 Creating Additional Users

Once the Administrator user has been created, it will be possible to log into any application as long as the proper permissions have been configured (see the Permissions section, under Database configuration). To ensure a newly created user has access to a particular application, several steps must be taken in these running apps.

7.3.1 Program Management

The Program Management component is used to create components,

1. Create *component*(s), for example the Applications in your system.
2. Create *tenant*(s) of which the user will be a member, if not already there (for example, State or District entities).
3. Ensure the *tenants* are subscribed to the *component*(s), are in good standing, and are within the appropriate effective date range (for example, the state of AK is subscribed to the Program Management, Permissions, and ART components).
4. Create a configuration file for the new component, and ensure the component references it properly (including environment name). Refer to the Property Configuration section of each component.

7.3.2 Permissions

1. Create a *component* with the exact same name as the one created in ProgMan, if not already there.
2. Create a new *role*, if not already there
3. Add a *permission* for that component, if not already there
4. Map the *role* to the component *permission*(s).

7.3.3 ART

1. Create a new *user* in ART application (must be configured for provisioning new users into OAM).
2. Assign one or more roles (defined in Permissions) to that user.
3. Associate the user with an entity that is in the hierarchy of the tenant which you defined in ProgMan.

A detailed matrix of component dependencies is available at Reference [3].