BRITISH
COLUMBIA

Ministry of Education

Technical Architecture Standard

Prepared by:	Ministry Architecture Committee (MAC)
Prepared for:	**Program Area IT/IM Initiatives (Projects)**
Version:	1.6
Last Updated:	December 18, 2013
Updated By:	Peter Holland
Creation Date:	August 4, 2009

Table of Contents

1. Introduction

The Technical Architecture (TA) document is provided by the Ministry to project teams to facilitate communications and ensure that project information technology/management needs are supported by the shared Ministry infrastructure.

1.1. Document Purpose

The objectives of the Technical Architecture Document are to provide all program area initiatives with a baseline understanding of the IT/IM standards and guidelines supported by the Ministry, in relation to the project.

For QCIL acceptance of the project, the project manager (program team) must agree to adhere to the standards/guidelines as expressed in the TA. Exceptions may be discussed, but will require a strong business reason, and be explicitly documented in the project charter. The PDO is responsible for coordinating signoff activities associated with the acceptance of the project charter and project plan.

The TA provides the technical underpinnings on which the program area “application architecture” AA can be established. An initial version of the AA must be completed and signed off prior to start of development. The AA is a required deliverable along with the operational support guide. Both documents must be signed off by the Ministry prior to an application being migrated to support user acceptance testing (UAT).

1.2. Intended Audience

The intended audience includes Information Technology Management Branch (ITMB) personnel, Application Maintenance Service (AMS) provider and project development teams.

This document is maintained by the Ministry Architecture Committee (MAC) and operation support staff.

1.3. Assumptions - Business Planning & Requirements Analysis

Prior to the start of any IT/IM program area initiative a level of business planning and associated analysis is required.

The preliminary business planning work and development of a contextual model is essential to ensure the scope of the initiative is well understood. Initial planning and creation of a business case is also mandatory for funding approval and to ensure program area strategic plans are aligned with cross organizational goals and objectives.

Setting up the required business requirement tasks with the expected outcomes (expectations) is part of the master project planning (MPP) process.

The amount of detail required in a Business Requirement Document (BRD) needs to be understood and agreed to. Setting expectations requires agreement among project stakeholders to ensure the details contained in the BRD are sufficient for the intended purpose, i.e. support for an RFP, detail design or code specification exercise. Assumptions regarding the scope of content for a BRD are part of a project planning phase.

Business planning and requirements analysis are required to support the creation of a business requirements document (BRD). The BRD and TA together are the required foundation documents necessary to support the creation of the Application Architecture (AA) document. The details inherit in the BRD may reflect both business level and system level requirements.

This TA document does not contain “legacy technology” options. If legacy (or net new) technologies are being used the following assumptions apply:

a. The application rationalization process which is part of the business planning phase has been used to support the decision to continue applying legacy (or adopt new) practices;

b. The planning phase is the point at which any departure from the Ministry sanctioned TA standards/guidelines should be quantified in the MPP and reflect a capability assessment and associated cost/benefit of the approach; and

c. Standards for the legacy technology/practices do exist and have been accepted as part of the MPP acceptance process.

2. Design Standards

Given the variation in complexity (size) of application systems (business services) some variability in the design approach is expected.

Example design variations may include design/development productivity enhancement patterns based on the use of a particular design/development environment (Eclipse, NetBeans, and JDeveloper) and/or whether web services (WSDL/XSD's) are being designed.

Project teams are required to submit an Application Architecture (AA) document (template provided) prior to or at the start of the design phase, that outlines a strategy for using the available subsystems and API's.. For initiates with a limited scope or represent only a small change with a limited organizational impact the Analysis, Design and Architecture (ADA) template can be used.

Regardless of the size of any funded IM/IT initiative the project plan should clearly outline the activities and expected deliverables,

Confirmation of the final approach for the design may be deferred until acceptance of the AA deliverable, although the project charter along with the MPP should establish basic strategies and expectations regarding timing of deliverables.

The Ministry is transitioning to the use of UML, and utilizes the Sparx Enterprise Architect (EA) product (and associated repository) for UML modeling.

The TA assumes the use of object orientated supported methodologies inclusive of relational modeling practices. The use of Sparx EA for data modeling activities and their dependency on the overall application design approach is a negotiated project detail, and is part of the MPP creation process with tasks/milestones and expectations to be made explicit.

Non negotiable points include the use of the Ministry's configuration management and continuous integration build process which is to be used for managing all design and development deliverables/artifacts and the respective code build and release cycle.

Configuration Management:

Software configuration management (SCM) provisions and maintains information about all the software components related to the delivery of business services.

Continuous Integration Process Supports:

A single source repository and an automated build and deployment process that is identical across all Ministry technical landscapes. Continuous integration also supports effective operational governance by enabling a quick status assessment of any codebase (application or service).

Security Management:

All aspects of Security must be developed and built within the application and be based on Best Practices and meet Government policy (Information Security Policy – Chapter 8).

Security must (unless technology prohibits):

- model role based privileges and be fully documented
- be based on the "least privilege" principle
- have the ability to report system privileges
- have documented a registration and deregistration process
- comply with the Government Information Security Policy for application passwords
- utilize IDIR/BCeID authentication
- data must be classified according to the Information Security Classification Framework

Developers must review Sans Top 25 software errors and OWASP Top 10 to ensure common vulnerabilities are avoided during application development.

3. Technical Infrastructure & Development Standards

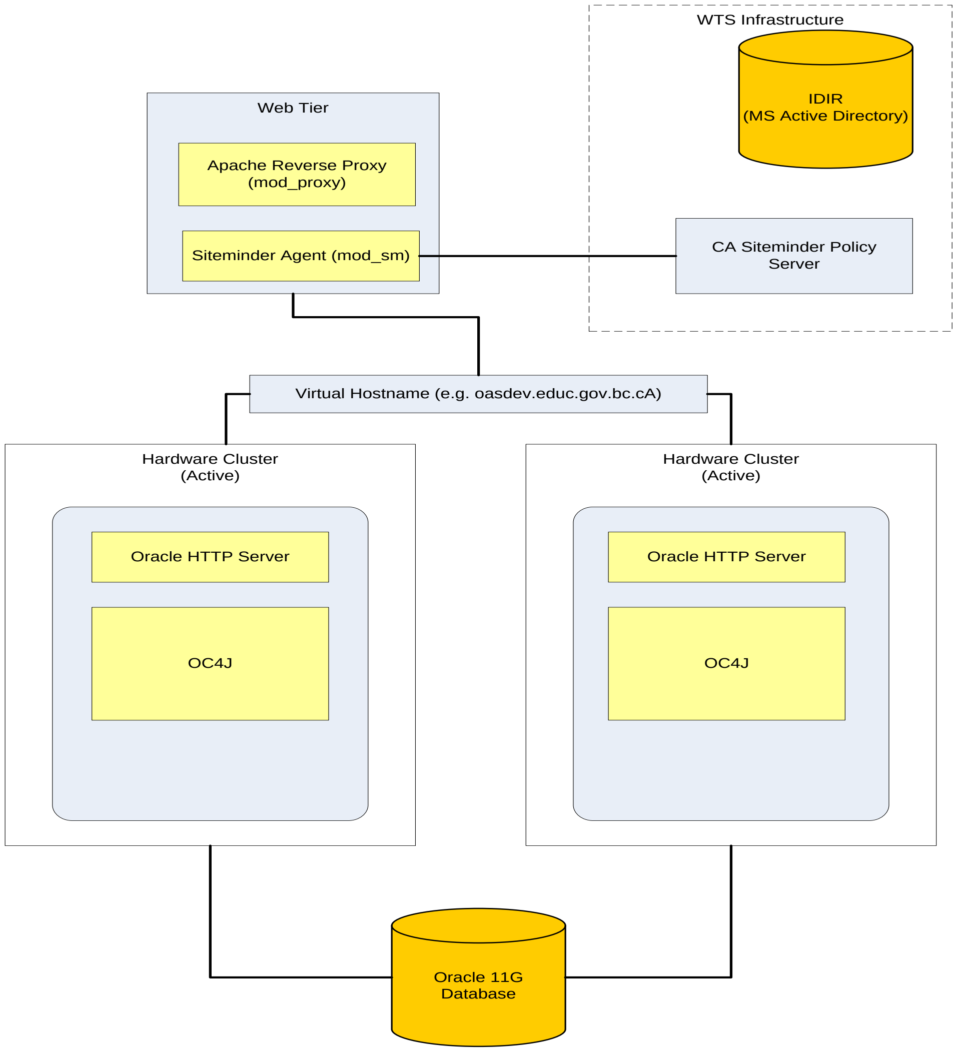
3.1. Java EE Environment

Platform

The Ministry uses Oracle Application Server 10G Release 3 as their platform of choice for hosting Java Enterprise Edition Applications. The application server is installed on Red Hat Enterprise Linux Server infrastructure configured using Oracle Clusterware to run in active-active clustering mode.

Components

The architectural baseline for typical Java EE application deployed into the Ministry environment is depicted on the following diagram:


WTS Infrastructure
IDIR (MS Active Directory)
Web Tier
Apache Reverse Proxy (mod_proxy)
Siteminder Agent (mod_sm)
CA Siteminder Policy Server
Virtual Hostname (e.g. oasdev.educ.gov.bc.cA)
Hardware Cluster (Active)
Oracle HTTP Server
OC4J
Hardware Cluster (Active)
Oracle HTTP Server
OC4J
Oracle 11G Database

Oracle 11G Databases – The ministry uses Oracle to store all application data.

Oracle Application Server (OAS) 10G Release 3 – is used to host Java EE applications. Applications get deployed into OC4J containers that are accessible via Oracle HTTP Server (OHS) interface.

Oracle Clusterware is used to enable clustering capabilities on the Java EE middle tier. The application server is configured to use active-active clustering so it is mandatory to access it by virtual host name that is shared across cluster nodes.

Web tier is implemented with Apache HTTP Server working in reverse proxy mode. The web tier serves the following purposes:

1. Virtualizes application URLs;

2. May act as termination point for HTTPS based traffic; and

3. Provides authentication/authorization services acting as enforcement point for CA Siteminder product.

CA Sitemeinder is used as authentication/authorization framework that protects the Ministry's applications. Siteminder uses BC Government wide Active Directory called IDIR as the source of users account information and is driven by policies defined in Siteminder Policy Server. The actual enforcement of the policies is done by Siteminder Agent component that runs as a module (mod_sm) in Apache HTTP Server.

Note: The description above uses IDIR as an authentication mechanism. Applications wishing to use BCeIDs for authentication need to be established as an eService in BCeID. The details associated with consuming BCeID web services to manage application level authentication and authorization is a project specific design detail that would be described in the AA document.

3.2. PHP Development

The Ministry supports PHP[1] as a technology option for small to medium size projects that do not require long lived or heavy weight database transaction volumes.

When implementing the project with PHP the following rules must be observed:

- Development should be done with PHP version 5 so it can run in the Ministry's environment;
- The Software Configuration Management process will be the same as for JEE applications; and
- An assessment of the value proposition between using PHP versus a JEE approach is required.

Please contact EDUC DL Middle Tier for PHP configuration information.

3.3. Recommended Developer's Toolkit

Java Development Kit (JDK)

All Java code must be compiled and tested against JDK 1.6, this allows the Ministry to build and deploy generated application into an OC4J container running as a part of Oracle Application Server infrastructure.

Relational Persistence for Java - Object-relational Mapping (ORM)

The Ministry's ORM tool of choice is Hibernate ORM framework. Hibernate is a powerful, high performance object/relational persistence and query service. Hibernate lets you develop persistent classes following an object-oriented approach using associations, inheritance, polymorphism, composition, and collections. Hibernate allows you to express queries in its own portable SQL extension (HQL), as well as in native SQL, or with an object-oriented Criteria and

[1] As of May 2010 the PHP Version is 5.2.6 – Note: contact "EDUC DL Middle Tier" to confirm.

Example API. Hibernate version 2.1 or later supports Java Persistence API (JPA) and can be easily integrated with EJB 3.0 based java beans. http://www.hibernate.org/

Enterprise Java Beans (EJB)

EJB 3.0 is supported by the Ministry. This includes session and message driven beans. This version of the specification is much lighter than previous releases and provides more productivity options for developers.

Spring framework

It is recommended to use spring framework to support the following design patterns/practices:

- Use inversion of control pattern, (dependency injection) to configuration application components and lifecycle management of Java objects in a flexible way;
- Adopt aspect-oriented programming (AOP) techniques to support common practices for implementing a separation of concerns (modularity) to help expose cross-cutting routines and enhance their usability.
- Enhance data access by using of ORM tools in conjunction with the Hibernate framework;
- Adopt common Authentication/Authorization patterns using the spring security (former ACEGI framework) to implement application authentication and authorization processes. Spring security is recommend to provide fine grained authorization in addition to Siteminder based protection.

Model-View-Controller Framework

Java web development initiatives are required to use JavaServer Faces (JSF) technology. JSF is a Java-based Web application framework intended to simplify development of user interfaces for Java EE applications. There are several implementations of JSF available and the Ministry doesn't restrict the use of any particular one although there are some successful examples of applications deployment into the Ministry environment that based on ICEfaces framework. ICEfaces is an open source Ajax framework that enables Java EE application developers to easily create and deploy server-based rich Internet application (RIA) in pure Java. http://www.icefaces.org/main/home/open-source-ajax.jsp

Web Services

The Ministry supports web services generated in accordance to JAX-WS 2.1 specifications. JAX-WS specification uses Java annotations to define web services interfaces. Oracle Application Server has built-in support for such services.

In looking forward, consideration may also be given to alternative web service infrastructure components such as project Metro that is web services stack from Sun Glassfish Application Server. Metro project is based on JAX-WS specification and provides a high-performance, extensible, easy-to-use web service stack.

Enterprise Service Bus

The Ministry currently has not deployed any service via an ESB, and may consider a number of product options if a Ministry wide Enterprise Service Bus solution is required.

Reporting

All new Java based reporting development is based on JasperReports. JasperReports is the world's most widely used embedded Java reporting library. It provides accelerated report development, support for web and print-ready production reports, high-performance, and massive scalability. http://www.jaspersoft.com/JasperSoft_JasperReports.html

Logging

The Ministry mandates the use of Log4J framework version 1.2 or later to generate application log records. http://logging.apache.org/log4j/1.2/index.html

Unit Testing

The Ministry supports the use of JUnit framework to implement unit testing for the application. JUnit 4.5 or later is recommended.

3.4. Java Coding Standards

Any attempt to document code is a good thing – there is an expectation (contractual obligation) on developers involved in creating/changing code to highlight the business rationale driving the change and any insight regarding implementation/system complexities/dependencies.

The expectation for code documentation exists and is as pragmatics as it can be. Business changes that are considered conceptually "simple" may have more complex implementation details which need to be made explicit as part of the design or specification for a change or at the very least is expressed as part of the code change.

This section focuses on Ministry specific areas of interest in regards to Java coding standards and Ministry specific sever side Java application/service (component) configuration practices.

Note: Code designers and developers are expected to adhere to industry accepted best practices regarding Java Coding Standards and know what they are.

The Ministry follows Oracle's (formerly SUN) Java coding conventions for Java development you can find at: http://java.sun.com/docs/codeconv

The following minimal standards (best practices) are mandated to ensure the appearance and organization of Java source code is consistent among the work produced by different developers.

Code Comments

Expected to follow "javadoc" commenting standards:

- Comments should add clarity. Documentation exists to enhance your understanding and the understanding of any other developer who comes after you;
- Keep comments simple. Point-form notes are useful to provide enough information so that others can understand your code;
- Each source file should begin with comment block as depicted below:

```
/**
* Description: (description of source file content)
* @author
* @version
*
*
* Copyright (c) 2009, Ministry of Education
*
* All rights reserved.
* This information contained herein may not be used in whole
* or in part without the express written consent of the
* Government of British Columbia, Canada.
*
*/
```

- Additional in-line comment blocks for methods and constructors should include a brief description as well as the following tag comments:

```
/**
* Description: describe purpose of method or constructor
* @param
* @return
* @exception
*/
```

How to Write Doc Comments for the Javadoc Tool:

http://java.sun.com/j2se/javadoc/writingdoccomments/

Naming Conventions

Refer to: http://www.ibm.com/developerworks/library/ws-tip-namingconv.html

Adherence to a consistent naming convention enhances the ability to produce understandable and maintainable code.

All Java packages should start with **ca.bc.gov.educ.<application name>** (Ministry of Education) or **ca.bc.gov.almd.<application name>** (Ministry of Advanced Education and Labour Market Development) where <application name> is official short name of the project. This can be also expanded to application component name and so on.

Exception Handling

The Java Language Specification (Second Edition) contains a complete reference regarding standard exception handling practices and can be found at:
http://java.sun.com/docs/books/jls/second_edition/html/exceptions.doc.html#44122

In addition the following rules must be also observed:

- At no time it is allowed to have catch block without either Log4J logging statement or raise clause. The worst case is when an exception happens and gets lost without being acknowledged;
- It is mandatory to implement finally block to free up resources taken in the code that resides within try-catch block. Following this rule can prevent resource leakages. One example of this is database Connection, PreparedStatement, ResultSet, FileInputStream and so on; and
- Every effort should be made to prevent exception stack trace to show up in application screen. Instead of displaying the stack trace on the screen, exception should be properly logged and user friendly message should appear on the screen.

Logging

Developers are required to use Log4J libraries to implement logging capabilities in the application.

Documentation regarding the standard practices and use of log4j can be and an FAQ can found on the following Jakarta site:

http://logging.apache.org/log4j/1.2/manual.html

The following rules must be observed:

- Developers are required to log every checked exception. The only exclusion from this rule is when it makes more sense to re-throw the exception and log it on a higher level in stack trace. The logging level must be set to ERROR if this is recoverable situation or FATAL otherwise;

- It is necessary to log a DEBUG level message on successful initialization of application major components. This can help to narrow down the problem when an application hangs due to resource allocation or race condition;
- Security events such as successful login/logout/granting access to resource must be logged as INFO message. All unsuccessful attempts to login or gain access to some resource must be logged as WARN message;
- At no time it is allowed to log any kind of sensitive information such as passwords, SIN numbers, credit card numbers and so on;
- The Log4J configuration file should not be included in deployment package. The application code must assume the location of log4j.properties file on the classpath of the application. The Middle Tier support group is responsible to create/copy and maintain the original Log4J configuration file and make it available for the application classpath.

Thread Safe practices

The following principals must be observed when dealing with thread issues:

- Java EE platform technologies such as EJB support multithread environments out of the box. Developers should not create user threads in the code that will be deployed into Java EE container;
- Make sure that the code that is invoked by Java EE container is thread safe:
  - Use synchronize blocks when deal with static code or singleton pattern and execution is shared across server threads;
  - If for any reason it's not feasible to modify the code to make it thread safe make your servlets implement SingleThreadModel interface. Please note that this approach may impact performance and resource consumption since the servlet engine creates a pool of servlets to make sure that any time each instance is dedicated to process one request at a time;

Use of HTTP Session objects

- If your application makes use of session objects, you must ensure that session tracking is enabled by having the application rewrite URLs whenever the client turns off cookies.
- Ensure that all your session variables are serializable. Make every effort to limit the overall size of objects that are kept in session scope, free up memory as soon as objects become not needed;

Use of JUnit

JUnit is a de facto standard in unit testing. The Ministry insists on the use of the JUnit tool for unit testing of Java code and encourages the developers to follow JUnit testing best practices.

Application/Service Configuration Information

The Ministry continuous integration and build process mandates the same deployment package be promoted from one environment to another with no modifications or post installation adjustments for **configuration files**. All environment specific configurations must survive application redeployment. The following rules must be observed to make this process succeed:

- Application must use the OC4J container provided Java EE resources such as database connection pool, JMS resources and so on. The application can leverage JNDI lookups to find resources provided on container level. <u>It must not use locally defined database connections or data sources.</u> The actual configuration of Java EE resources is done by Middle Tier support group before initial deployment of the application;

- For new applications do not manage environment specific configurations in an applications database table. Where this state is unavoidable make this “application feature” clear in the system documentation and ensure business stakeholders are aware of the dependencies and/or potential complexities this can add to implementing certain types of changes.
- The application configuration values that are environment specific should be extracted into a properties file that will be put into target OC4J applib directory by Middle Tier group. The Middle Tier group is responsible for updating the values to the target environment. The application can get access to this file via application classpath. Here is an example on how to do this:

  ```
  private Properties getPropertiesFromClasspath(String propFileName)
  throws IOException {
          Properties props = new Properties();
          InputStream inputStream = this.getClass().getClassLoader()
              .getResourceAsStream(propFileName);

          if (inputStream == null) {
              throw new FileNotFoundException("property file '" +
  propFileName
                  + "' not found in the classpath");
          }

          props.load(inputStream);

          return props;
      }
  ```

  Note: Environment neutral configuration can live in properties file that is packaged in an ear file or alternately be placed in web.xml file. Environment specific properties file should be named *<application>.properties* file, but environment neutral one will be named as *<application>-common.properties*. The same rule applies to xml based configuration files;

Application Build Process and Versioning with Ant

Developers will be provided with initial Ant build script they can extend and maintain so it will be capable of generating deployment packages and running JUnit unit tests. The following rules must be followed when dealing with Ant build script:

- The Ant build script should be named build.xml and it must reside under *build* directory starting from the project root;
- The script must have at least the following targets in it:
  - *clean* – to delete all generated artifacts such as class files, deployment packages, unit tests results;
  - *compile* – to compile java sources. All java classes must reside under *bin* directory starting from project root;
  - *test* – to run JUnit tests. The script must generate unit test report under *test* directory starting from project root;
  - *package* – to build deployment packages. All deployment packages must reside under *dist* directory starting from project root. Application deployment package should be named as *<application>.ear* and shouldn’t have version number in its name. This target should be set as default;
- The Ant script must be able to generate deployment packages in generic environment without any modifications in build properties. Please consult with the Ministry staff if you think it’s not achievable;

- The Ant script must not make any assumptions on the local system directory structure or platform it runs on. The use of absolute path or relative path that points outside of the project root directory is not allowed;
- It is not allowed for the Ant script to make any modifications in the source code during the build. This includes configuration files but doesn't include generated artifacts;
- The Ant script must run on default Ant installation, every extra library that is required for the build to succeed must be explicitly defined in the build script and all extra libraries must reside within project tree;
- Every effort should be made to avoid hardcoding of property values such as classpath, directory names and so on. Use *propertyfile, property, pathelement* instead.
- There are situations when the application will consist of multiple layers that can be built separately but they depend on each other. For example the ear file has a web module and an EJB module in it. Each of these modules must be built separately before the final ear file can be assembled. The web module uses jar files that are also built by the same Ant script. Ant build files must be designed to compile large projects in stages:
  - Compile shared utility code, placing the results into a JAR file;
  - Compile a higher level portion of the project against the JAR file(s) created in the first step;
  - Repeat this process up to the highest level of the system.
- The Ant build script must define an effective dependency graph so every logical task such as compiling the code or running unit test can be done by invoking just one target. These dependencies must be reviewed periodically to make sure that it still makes sense and there is no need to invoke a series of target to accomplish some task;

3.5. Web Application Development Guidelines

The Ministry has supplied the following guidelines to help to make web application more robust, secure, and accessible:

- Use well established framework that implements MVC design pattern. The outcome will be an application with well defined layers delivered in a shorter timeframe. The Ministry recommends using ICEFaces framework that implements Java Server Faces (JSF) technology;
- HTML code generated by application must comply with W3C HTML 4.01 Standard. This can be verified by W3C Validator service that is available here: http://validator.w3.org/
- The use of technologies that require browser plug-in install such as Flash, Java Applets is not currently supported.
- The use of Java Script must be limited to what is absolutely necessary to make application functioning. Java Script technology has a bad record of cross browser interoperability so the Ministry encourages developers to leverage frameworks that abstract the use of Java Script, with an Ajax enabling framework.
- To achieve better consistency, prevent creation of redundant code and reduce development effort when implementing presentation part of the application the Ministry recommends to take the following steps:
  - Use Cascading Style Sheets (CSS) technology to separate the content from document presentation. Things like document layout, elements style, colour and so on should be implemented in CSS files that are shared across application;

  - Use templates to represent common parts of the page such as menus, page headers, footers and so on. Struts Tiles, Spring MVC templating, ICEFaces Facelets for example;
  - Use available Custom Tag Libraries. There are numerous resources for accessing custom tags. Avoid re-inventing the wheel.

- Use HTTP GET for read only operations. Note: State or data modifications as a result of a processing HTTP GET request is not allowed;
- No business logic execution should be done from presentation layer. This includes direct database access, manipulations with business objects and so on;
- Prefer forward to HTTP redirect since it is more efficient and transparent to the client. HTTP redirect requires a roundtrip to the client and extra care needs to be taken on reverse proxy side;
- Use JSP include directive to enable reuse of some common page components. This can be superseded with a templating framework;
- As part of the planning phase for any changes to ensure FOIPOP guidelines have been addressed. Ensure no sensitive/private information is being cached (for example).
- Each internal application must be officially tested against MS Internet Explorer versions 6 and 7. All public facing applications must be also tested with other major browsers such as Mozilla Firefox, Apple Safari, and Opera.

  Follow HTML best practices. Contact the Ministry’s Web Tier operational personnel for additional information regarding current ministry and B.C. Government web standards. The contact email is “DL Middle Tier” available on the B.C. Government exchange global access list (GAL).

4. Technology Landscapes & Release Management

This part of the document describes in details the process of managing the code changes as well as code delivery to the Ministry's supported environments.

4.1. Software Configuration Management (SCM)

Purpose

The goals of SCM are:

- Configuration identification - What code are we working with?
- Configuration control - Controlling the release of a product and its changes.
- Status accounting - Recording and reporting the status of components.
- Review - Ensuring completeness and consistency among components.
- Build management - Managing the process and tools used for builds.
- Process management - Ensuring adherence to the organization's development process.
- Environment management - Managing the software and hardware that host our system.
- Teamwork - Facilitate team interactions related to the process.
- Defect tracking - Making sure every defect has traceability back to the source

Versioning Control

The Ministry uses Subversion as a tool of choice to perform versioning control for applications source code. The Ministry provides a repository for every project that is accessible using HTTPS protocol and requires IDIR authentication.

The Ministry acknowledges the fact that some SCM related terminology may have different meaning in different tools so please use these documents as a reference:

- Version Control with Subversion - http://svnbook.red-bean.com/
- Subversion Wiki - http://www.orcaware.com/svn/wiki/Main_Page

Versioning would be three sets of numerals (e.g. 1.04.1978) where the first set indicates major new functionality, the second set indicates minor new functionality and the third set indicates bug fixes. I.e. :
<major>.<minor>.<build>
The first two sets are changed manually in Anthill by the MT team at the request of the BA. This should occur before the first build and deploy process is run by the development team. The third set is incremented automatically by Anthill at each build. This build number should never be reset as it represents the chronological continuity of the application.

While the ministry would prefer that no 2-phase development take place, this process and our tools can support this. Phase 2 development would continue to use the trunk code base. In the case of an emergency fix request to the Phase 1 code base, the BA would request that the MT set up a new Anthill project. This would require that a branch be created in the source code (using the last Phase 1 code) and the new Anthill project (Phase 1 EFX) would point to that branch in the SVN repository. This Phase 1 EFX project would be able to deploy to EFX (by DEV team) and to PRD (by MT team). It is beyond the MT team's scope to manage the branch and how a code merge is handled. Once the Phase 2 code has been deployed to PRD, the Phase 1 EFX Project would be disabled.

The following list outlines the principals and processes that must take place during project development:

- The development is done in trunk version of repository;

- To support for additional environments, production at very minimum, it is required to create a branch in repository so any additional development done would not affect the code deployed in production and at the same time any bug fix introduced in production and committed to production branch can be easily merged back to the trunk;
- Developers are responsible to obtain the latest copy of the source code from repository on daily basis;
- Developers are responsible to commit their changes as soon as possible without compromising the quality or ability to build the project.
- Developers are required to provide meaningful description of the change on every commit to repository. If the change is driven by some issue from issue tracking system, developers are responsible to provide a reference to the issue number. That will allow the Ministry's tools to link the file changes to the issue in issue tracking system and generate reports that show relationship between issues and Subversion changes;
- All of the application source code including, Oracle forms (fmb) Java classes, XML configuration files, JEE artefacts, SOA artefacts, additional libraries, SQL scripts and so on must be kept in repository.
- All migration time documentation such as Middle Tier Service Request (MTSR), any additional instructions, and release note must be committed to repository. C2 manages the "incident period" (i.e. start and end notification) the MTSR contains the details regarding those activities of the how the change is implemented and provides a useful and reusable information which is relevant to the particular changes. Any of these files get unique http based URL that can be used as a reference to the document. This eliminates the need of attaching files in an email correspondence or ticket tracking system.
- The Ministry follows recommended layout for Subversion repository. Each repository will have at minimum the following directories:
  - /trunk – represents the trunk version of the code;
  - /branches – all branches created must live under this directory. The branch name should clearly indicate why the branch was created. For example "rdms-prd-20090318" may be created to support the version of rdms application deployed to production on March 18th, 2009;
  - /tags – The tagged sources should live under this directory. Please note that Anthill Server tags the files with build version number each time a new build life is initiated;
- Developers are free to use any Subversion client. The Ministry recommends using Subversion support that is integrated with all modern Java IDEs such as NetBeans, Eclipse and so on. This can help to increase productivity of development team and avoid situations when the code in IDE becomes stale due to Subversion update done outside IDE.
- The Ministry made every effort to make Subversion repository accessible to development teams working offsite so any use of other repository within development team is strongly discouraged. Situation when developers work on their own repository and at some point synchronize their changes with what is in the Ministry repository can lead to missing updates and misleading commit messages.

Code Review Process

The Ministry reserves the right to perform occasional code reviews to verify the source code quality and conformance to the Ministry's development standards. Ministry staff will check out the code from Subversion repository to perform code reviews to support requests to confirm adherence to TA standards and guidelines.

Supported Environments (Landscapes)

The following table described all environments that are available to host applications:

Name	Abbreviation	Purpose
Sandbox	SBX	All patches are applied and validated in SBX before rolling out to other environments. This environment is not generally accessible by development team.
Development	DEV	DEV environment is used to support Unit Testing activities.
System / Integration Testing	TST	TST environment is used to support System Testing activities.
User Acceptance Testing	UAT	UAT environment is used to support User Acceptance Testing activities. This includes load testing that can be done as part of UAT process.
Production	PRD	This environment is to support production for the application.
Emergency Fixes	EFX	EFX is used to fix serious problems in that occur in the production environment. Production gets copied to EFX. Developers work in the EFX environment to fix the problem. The fix gets promoted directly to production bypassing the regular route of going through DEV-TST-UAT-PRD environments. Serious performance problem resolution may also require the use of EFX environment.

For middle tier inquiries contact “**EDUC DL Middle Tier**” available on the B.C. Govt. exchange global access list (GAL).

For data tier inquiries contact “**EDUC DL DBA**” available on the B.C. Govt. exchange global access list (GAL).

Release Management

The build and deploy/release process is automated via Anthill Pro Application Lifecycle Management Server. The source code is checked out from defined repository and Ant based build script is invoked by Anthill. Anthill can also collect build artifacts such as ear files and store it in its own repository. At the same time Anthill can generate some reports such as a change log since the last successful build. The deployment process can also activated by Anthill via its web interface.

The development team will be able to start the build and do deployment to development (DEV), test (TST) and emergency fix (EFX) environments where they can test the application code changes. Any subsequent deployments to other environments will be done by the Ministry Middle Tier group using Anthill.

Anthill uses the concept of the build life where any activity such as deployment into any environment can be traced back to the source code that was used. The major benefit is a guarantee that the same exact artifacts are being used for each additional process (abbreviated test suite, exhaustive test suite, release, etc.). And since the same exact artifacts are being used, then there is a guarantee that the same exact sources were used to generate those artifacts.

Each build life can be initiated manually via Anthill Web Console or be triggered by commit to repository or be scheduled to run periodically.

Not all of the builds can end up in production. A build can be withdrawn at any point along development to test and further. The following summarizes the respective release path and associated responsible role(s).

Build EAR ---> **Deploy to DEV** ---> **Deploy to TEST** ---> **Deploy to UAT** -----> **Deploy to PROD**

|------------------*by Developer Team* ---------------------||-------------*by Ministry Operations Team* -----|

Standard Directory Structure

This section documents the application source code directory layout expected by the Ministry. Please make sure that you follow this layout as close as possible.

Directory	Purpose
src/main/java	Application/Library sources
src/main/resources	Application/Library resources
src/main/filters	Resource filter files
src/main/assembly	Assembly descriptors
src/main/config	Configuration files
src/main/webapp	Web application sources
src/test/java	Test sources
src/test/resources	Test resources
src/test/filters	Test resource filter files
build/	Ant related files such as build.xml
README.txt	Project's readme

This layout mimics default layout that is used by Maven framework (http://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html).

Currently the Ministry does not support the use of Maven to build applications.

Build Process

The Ministry will be responsible for enabling the integration of the application into the AntHill process and deployment management system and giving access to run the build/deployment features to development teams. This is done by configuring the invocation of the Ant build script provided by development team during the initial Anthill build execution.

It is understood that initial deployment to some environments may require additional configuration steps that are quite unique so they cannot be automated. All subsequent deployments can be automated via Anthill interface and containers that are already configured will be utilized.

Deployment Process

All deployments into environments are automated via Anthill Server. The deployment process is implemented as a secondary workflow in Anthill and uses artifacts collected during the build workflow. Developers will be given access to deploy the application into development (DEV), test (TST) and emergency fix (EFX) environments. All subsequent deployments into other environments will be done with a use of Anthill, by the Middle Tier support group.

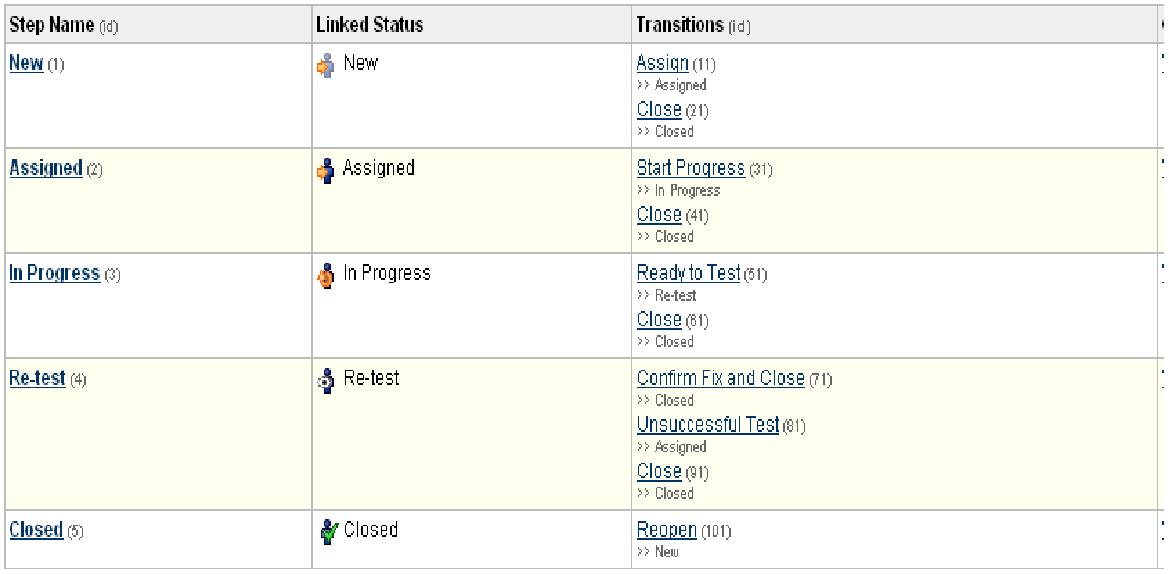
Promoting to Environment

Promoting the build life into the next environment must be accompanied with proper documentation as outlined in other Ministry's standard documents. Specifically the "**Operational Support Guide", available from this locationhttp://www.bced.gov.bc.ca/imb/downloads/cdmdt/dt_doosg.doc ,** is required to be signed off prior to a production release. These documents must reside in Subversion repository. At least one Subversion reference to the Middle Tier Service Request Form (MTSR) must be provided. This form should include Anthill Build Life reference pointing to the version of the application that needs to be migrated to environment. The actual deployment is done by Middle Tier Support group staff.

Issue Tracking

The Ministry standard issue tracking system is Atlassian JIRA. JIRA application uses IDIR authentication and is available from this location: https://gww.jira.educ.gov.bc.ca . Note that this is only for UAT defects. Incident support management (that does not include a UAT phase) is managed through C2, where an immediate fix is required

The following is a snapshot of the workflow describing transitions and statuses of the issue:



Step Name (id)	**Linked Status**	**Transitions** (id)
New (1)	New	Assign (11) >> Assigned Close (21) >> Closed
Assigned (2)	Assigned	Start Progress (31) >> In Progress Close (41) >> Closed
In Progress (3)	In Progress	Ready to Test (51) >> Re-test Close (61) >> Closed
Re-test (4)	Re-test	Confirm Fix and Close (71) >> Closed Unsuccessful Test (81) >> Assigned Close (91) >> Closed
Closed (5)	Closed	Reopen (101) >> New

A new issue can be raised by project team member during the project lifetime. A new issue gets assigned automatically to the project lead. The project lead plays a role of coordinator who prioritizes and assigns the work to development team.

Note: This workflow is provided as an example and can be adjusted to better fit the project needs.

After a developer is assigned to work on the issue, he or she can start progress on the issue, then mark it as ready for testing, or reassign it back to coordinator if developer thinks that this is not a real issue.

The QA team can test the fix and mark it as fixed and closed or pass it back to coordinator as unsuccessful test.

At any time privileged users such as projects leads can close the bug and mark it as Fixed, Won't Fix, Duplicate, Incomplete, and Cannot Reproduce. Any bug that is closed can be reopened if more work is required on this bug.

4.2. Project Startup Kit

During the project initiation the Ministry will be responsible for providing the following to the Project Development team:

- Subversion repository seeded with startup application called "Hello Shirley" will be provided to the development team. This application has a couple of servlets, JSP pages, deployment descriptors and includes an initial Ant build script that can be used to build the application deployment package (ear file);
- An OC4J container will be created in every landscape supported by the Ministry and the startup application will be deployed at least in DEV to ensure that created container is healthy;
- A new project will be created in Jira and configured to point to the applications newly created repository;
- A new Anthill project will be created and configured to support the applications delivery process. The project will be linked to the Subversion repository and the JIRA project created for the application;

The development team will be able to start development activities by extending the "Hello Shirley" application, committing the changes to Subversion repository and running the build and deployment processes in Anthill.

5. Testing Strategy

Defining a test strategy is performed in the Business Design phase. This strategy will include the development of test scenarios, test cases and a detailed test plan.

This part of the document describes various types of testing that needs to be done for a project to verify the quality of the code.

Need to provide single TEST reference document and amalgamate what used to be separate test strategy documents – part of ongoing BPP document review and consolidation.

5.1. Automated Unit Testing

The Ministry may ask developers to accompany the source code delivery with a suite of automated JUnit tests. These tests may be run by Anthill Server during the build process to ensure the quality of the source code and enable regression testing capabilities.

5.2. System Integration Testing

The Ministry provides TST environment so developers can get a chance to deploy the application into this environment to exercise all integration points such as Database Connectivity, Siteminder protection, JMS messaging, LDAP integration and so on. The TST environment is created to be as close to PRD as possible. For more information on Ministry standards on System testing, please refer http://www.bced.gov.bc.ca/imb/downloads/system-test-process.pdf

Note: The project plan is expected to outline who is tasked with performing testing.

5.3. User Acceptance Testing

Official User Acceptance Testing is done in the Ministry's UAT environment. For more information about UAT please see this document

UAT is performed by the business area.

5.4. Security Testing

The Ministry requires a Security Threat and Risk Assessment (STRA), the outcome of the STRA will indicate the level of security testing required.

5.5. Load Testing

The application architecture deliverable includes a capacity plan assessment that is mandatory to ensure the capacity of the shared Ministry infrastructure is capable of supporting transaction loads beyond those estimated to occur during peak business activity periods. Load testing is the responsibility of the AMS provider, with Ministry Operations team (Middle Tier and DBA teams) monitoring the resource utilization while the load testing is in progress.

Note: Load/Stress testing is mandatory for all new application/services and for changes to existing applications where large volumes of database transactions are involved.

6. Project Specific Technical Architecture Details/Exceptions

The MPP process should highlight any TA questions/comments or exceptions early in the project. If a project specific technical detail is not covered in the Ministry TA document but is accepted as a new supported technical infrastructure component, the TA document will be updated by the Ministry to reflect the change after the successful deployment and production migration of the respective application/service.

Note: Applying a technology to meet a business need (requirement) is not a TA item, details on how a technology is being applied (utilised) is an application architecture (AA) item.

7. Revision Log

Date	Version	Change Reference	Reviewed by
Aug. 4, 2009	V1.1	Final Review	MAC
July. 2, 2009	V1.2	Revised content to change outdated link information	MAC
Feb. 5, 2010	V1.3	Address CGI comments and refine content	MAC
June. 10, 2010	V1.4	Adjusted document to be less JEE centric but recognize the OAS stack (JEE centric) is the primary infrastructure supported.	MAC
Oct. 7, 2010	V1.5	Added “Security Management” sub-section to Section 2 Design Standards.	MAC
Dec 18, 2013	V1.6	Removed reference to the TRN1 environment	MAC

8. Appendices