

*FSA Integration Partner*

United States Department of Education

Federal Student Aid



# XML Registry and Repository Product Test Plan

**Version 1.0**

May 28, 2004



## Amendment History

DATE	SECTION/ PAGE	DESCRIPTION	REQUESTED BY	MADE BY
05/27/04	All	Final version submitted for FSA-wide review.	N/A	H.Sibunruang



## Table of Contents

<b>1</b>	<b>INTRODUCTION .....</b>	<b>5</b>
1.1	OVERVIEW .....	5
1.2	PURPOSE .....	5
1.3	SCOPE .....	5
1.4	ASSUMPTIONS .....	5
1.5	TEST OBJECTIVES .....	5
1.6	DOCUMENT OBJECTIVES .....	6
1.7	RESOURCE AND RESPONSIBILITIES .....	6
<b>2</b>	<b>TEST PLAN .....</b>	<b>7</b>
2.1	PRODUCT TEST OVERVIEW .....	7
2.2	TEST CYCLE APPROACH .....	8
2.2.1	<i>Entry and Exit Criteria</i> .....	8
2.2.2	<i>Developer Testing</i> .....	9
2.2.3	<i>Product Testing</i> .....	10
2.2.4	<i>System Investigation Requests (SIRs)</i> .....	11
2.3	TEST PLAN .....	12
2.3.1	<i>Process</i> .....	12
2.3.2	<i>Test Scenarios</i> .....	13
2.3.3	<i>Test Scripts</i> .....	13
2.3.4	<i>Test Script Standards</i> .....	13
2.3.5	<i>Test Conditions</i> .....	14
2.4	REQUIREMENTS TRACEABILITY .....	15
2.4.1	<i>Baseline Test Data</i> .....	15
2.5	TEST ENVIRONMENT .....	15
2.5.1	<i>Test Execution</i> .....	15
2.5.2	<i>Test Passes</i> .....	15
2.5.3	<i>Identification and Resolution of Issues</i> .....	16
2.6	DEVELOPMENT, TEST, AND PRODUCTION ENVIRONMENTS .....	17



## Tables and Figures

Table 1 - Test Passes.....	11
Table 2 - SIR Severity and Definitions.....	11
Table 3 - SIR Status Levels .....	12



## 1 Introduction

### 1.1 Overview

Core Components will define the basic building blocks that can be used across the FSA enterprise and the Postsecondary Financial Aid Community to create eXtensible Markup Language (XML) message specifications.

These core components and supporting XML documents and schemas need to be accessible to interface developers at FSA in order to define new XML message specifications. An XML Registry and Repository is a tool that enables users to access documents and services from one common repository of information.

### 1.2 Purpose

The purpose of the XML Registry and Repository Test Plan is to present the test approach for the first release of the Registry and Repository application. The Test Plan defines the business events, test conditions, test cycles, test scripts and scenarios to be tested. It describes the overall testing approach for the application. The objectives of system testing are to:

- Ensure that a quality product is delivered to the FSA stakeholders.
- Minimize risk during implementation.
- Find and fix problems during the development process.
- Follow FSA guidelines and SLC procedures.

This test plan provides an overall framework for testing the XML Registry and Repository application and is not intended to provide step-by-step instructions (i.e., test scripts) for every aspect of the testing effort.

### 1.3 Scope

The XML Registry and Repository application will undergo component, assembly, product, and user acceptance testing before being made available to end users. Successful completion of these testing phases will ensure that the new application meets the business needs of the end users as well as the functional and technical requirements specified in the XML Registry and Repository – Requirements Traceability Matrix.

### 1.4 Assumptions

- **Performance Testing** – Refer to Appendix F: Performance Test Plan and Results
- **Usability Testing** – Refer to Appendix E: 508 Test Results

### 1.5 Test Objectives

The objectives of the XML Registry and Repository Release 2.0 test effort are to:



- Verify the application satisfies the business requirements and supports the business processes as defined by the Registry and Repository requirements and use cases. Verify the functionality of the application performs as designed.
- Validate the integration of component tested units of code into assembled modules of code.
- Test the functional interaction of individual units and ensure functional usability.
- Ensure that the individual units of code correctly pass the data and session level information between each other.
- Confirm the implemented software modules meet the documented business requirements.
- Verify the modules of code can properly handle (and recover from, if necessary) expected and unexpected error conditions.
- Validate the functional and technical interaction of the graphical user interface layer, architecture middleware layer, and the data access layer.
- Ensure the overall software integrity.

### ***1.6 Document Objectives***

This Master Test Plan for the Registry and Repository Release 2.0 solution supports the following objectives:

- Detail the activities required to prepare for and conduct the testing effort.
- Define the scope of the testing effort, including the items to be tested.
- Document the overall testing approach.
- Define the test environments required to conduct the test.
- Identify and communicate the tasks to be performed and the schedule to be followed to responsible parties.

### ***1.7 Resource and Responsibilities***

The XML Registry and Repository Implementation team will prepare and execute the test scripts for the component, assembly, and product test phases and will facilitate User Acceptance testing.

The XML Registry and Repository Test team will prepare and execute the test scripts for the product and end-to-end tests. The test team will also aid in the facilitation of user acceptance testing, including the preparation of test scripts.



## 2 Test Plan

This section will provide a detailed overview of the Registry and Repository Test Plan. Specifically, this section includes information on the following areas:

- Test Approach
- Test Plan
- Test Environment
- Testing Schedule

### 2.1 *Product Test Overview*

#### **Modules**

The Modules to undergo testing correspond to the Application Features defined in the XML Registry and Repository Document. The following modules will be tested in cycles according to the Test Cycle Approach:

- Administration
- Classifications
- Core Component Management
- Sector Library Management
- Message Specification Management
- Landing Page
- Notes
- Search
- Error Handling

#### **Test Description**

During product testing for each module, any defective code will be corrected in the development environment and retested at the component level. All of the source code for that tested code will be repackaged and versioned prior to additional testing.

The user interface layer interaction with the middleware and data access components will also be thoroughly tested during this phase. Any defects discovered with these shared functions will also be retested at the component test level. Other modules, impacted by these shared functions, will be regression tested to ensure middleware and data access coding fixes do not impact other modules. A product test database will be generated. This common database will be used to test all coding modules. This database will not be refreshed at any point during the testing cycle, nor will it be refreshed during subsequent regression testing.

During the Product Test execution phase, defects will be tracked by test cycle and execution date in a defect tracking spreadsheet. For each coding module, a regression testing plan will exist. Whenever a coding module is re-released into Product Test, the regression test plan must be successfully executed prior to packaging and migration to production.



## 2.2 Test Cycle Approach

The following test cycle approach will be used to conduct the Product Test.

### Cycle 1 - Normal

This cycle is designed to test the application's ability to perform expected business functions under normal conditions. The scope of testing for this module will be limited to business functions which can be performed entirely within the associated module. This cycle will ensure that all of the module's associated business requirements have been met.

### Cycle 2 - Expected Error

This cycle is designed to test all expected application error conditions to ensure the code properly handles all planned exceptions. As an example, a representative test condition would ensure that an alphanumeric character could not be entered into a numeric-only telephone field.

### Cycle 3 - Unexpected Error

This cycle is designed to test unexpected error conditions and ensure that the system can properly detect and recover from unexpected system errors. As an example, the application should properly roll back uncommitted data upon a database failure. Upon system restart, there should be no system wide impacts.

### Cycle 4 - Regression Testing

This cycle is designed to provide a regression testing approach at the module level. This cycle will be executed prior to migrating new versions of code to the production environment. This cycle is essentially a subset of the overall product test.

#### 2.2.1 Entry and Exit Criteria

For each test cycle, entry and exit threshold criteria will be established. The purpose of entry and exit criteria is to improve the quality of the testing effort by acting as an agreement between the Implementation and System Test teams. The criteria will be used to judge software quality levels and readiness for system testing. Entrance criteria will define the conditions that must exist prior to the start of a test execution phase. Exit criteria will define the conditions that must exist prior to the software being released from the testing environment.

The entry and exit threshold criteria for relevant test phases will be reviewed before terminating a test cycle and proceeding into the next one. Any deliverable failing to meet its criteria will be returned to its current test cycle.

#### Entry Criteria

Entry Criteria are standards used to determine when a software development project is ready to enter a given phase. The entry criteria for each phase of the XML Registry and Repository test effort are:

- Code modules completed – unit and component tested.
- Test preparation activities for the particular test phase are complete.
- Resources are available to execute tests.



- Test environment and test data are in place.

### **Exit Criteria**

Exit criteria are standards used to understand what is required of a software development project to exit a particular phase. Each test phase must complete the following items:

- Relevant test cases and conditions have been executed.
- Identified issues have been properly documented and worked through the resolution process.
- All defects have been logged as System Investigations Requests (SIRs) in eProject.

If the number and severity of the open SIRs meets the exit criteria, the system will move to the next test phase and the remaining open SIRs will be resolved then. If the product has not met the exit criteria, SIRs will be resolved and tested within the current phase until the exit criteria are met.

## **2.2.2 Developer Testing**

The system test phase contains the following two types of testing:

- Component Testing
- Assembly Testing

### *2.2.2.1 Component Testing*

The Registry and Repository Implementation Team will conduct component testing to ensure that each developed code module meets its particular business needs and requirements. Component testing will be conducted in the development environment.

A component test is the test of an individual component, or module, of the solution. The objective of a component test is to verify that the component correctly implements the designed specifications.

Component testing is based on knowledge of how the logical [code] unit is designed to work. The test conditions for the component test are unique to this effort and will not be repeated in other efforts once the exit criteria have been achieved. Component testing will include tests for field ranges, values and lengths, functions, data validation, data dependencies, and any special processing contained in the module.

Modules will be developed and become available for testing independently of other modules. The developer responsible for the module will perform the component test and identify and document the errors related to the independent operation of the program.

### *2.2.2.2 Assembly Testing*

The XML Registry and Repository Implementation Team will conduct assembly testing, in addition to component testing, to ensure that the developed code modules meet their particular



business need and the requirements. Assembly testing will also be conducted in the development environment.

Assembly test is a string of individual components, or modules, of the application that when combined together test an entire scenario end-to-end. Assembly test scripts are based on knowledge of how the code modules are designed to work together and will mimic the system test scripts. The assembly test scripts will be updated if changes occur to the requirements and/or the detailed design specifications.

All Level 1 SIRs will be fixed and closed prior to progressing to system test. In addition, there should be a minimal number of open Level 2 SIRs per module. No defined limit will be set for Level 3 SIRs while Level 4 SIRs are system enhancements and will not count towards the exit criteria for this phase.

### 2.2.3 Product Testing

The product test phase consists of End-to-End System Testing. At the end of the product test, relevant issues should be identified and documented. All Level 1 SIRs will be fixed and closed prior to progressing to the user acceptance test. A determination on the number of open Level 2 and 3 SIRs will be at the discretion of the delivery manager. Level 4 SIRs are system enhancements and will not count towards the exit criteria for this phase.

#### 2.2.3.1 End-to-End System Testing

System testing concentrates on requirements and business processes, and ensures that the system requirements have been met. System testing will test the components and interfaces working together as a whole, integrated solution.

The XML Registry and Repository Test team will handle the creation and execution of test scripts. The test scripts for this phase will be developed using the high-level scenarios developed during the Design phase.

Ideally, system testing cannot begin until all modules have passed both the component and assembly test phases because the modules must be fully integrated in the test environment for system testing. However system testing may begin prior to the acceptance of all modules within the prior phases, but the impacted modules must be moved to the end of the test phase. This decision will be at the discretion of the XML Registry and Repository Test Team Lead.

The Test Team will conduct system testing on the Registry and Repository application to ensure that all developed code modules work together to meet the intended business needs and requirements. The XML Registry and Repository Test team will conduct end-to-end testing to ensure that the components of each system work together as intended. Both end-to-end and system testing will be conducted in the test environment.

#### 2.2.3.2 Passes for Each Level

Pass	Test Cycle	Description
First Pass	Full Test Cycle	



Second Pass	Full test cycle	Performed to re-test failed items from first pass and confirm previous successes from first pass after failed items pass successfully.
Third Pass	Partial test cycle and Regression Test	Performed to re-test failed items from the second pass. Anything more than 3 passes probably indicates a serious quality problem, and will be reported to the Test Team Lead. The Test Team will continue to add additional test passes until all test conditions are met successfully.

**Table 1 - Test Passes**

### 2.2.4 System Investigation Requests (SIRs)

Whenever a test condition or test script step does not return the expected result, the tester will log the discrepancy between the expected and actual result as a SIR. SIRs will be logged and tracked in a central repository, or SIR database, using eProject. The SIR database will include a long and short description of the issue (including the actual and expected results), its severity and status, the affected test area or module, the test phase, test pass, and tester.

#### 2.2.4.1 SIR Severity

The following table contains the SIR severities and definitions. The tester will use these definitions to provide an initial severity, but the XML Registry and Repository Test Team Lead will have the authority to make a final decision on SIR severity levels.

ID	Label	Description
1	High	The anomaly results in a failure of the complete software system, a subsystem, or a software module within the system. There is no way to make the failed component(s) work; however there is an acceptable processing alternative that will achieve the desired results (an acceptable work around exists).
2	Medium	The anomaly does not result in a failure, but causes the system to produce incorrect, incomplete, or inconsistent results, or the anomaly impairs system usability.
3	Low	The anomaly does not cause a failure, does not impair usability, and the desired processing results are easily obtained by working around the anomaly. The anomaly may also be the result of non-conformance to a standard or related to the aesthetics of the system.
4	Enhancement	The anomaly is a request for an enhancement. Anomalies at this level may be deferred to a future release.

**Table 2 - SIR Severity and Definitions**



### 2.2.4.2 SIR Status

The following table contains the SIR Status levels that will be used during this effort and a definition of each:

SIR Status	Definition
Opened	The SIR was created and is awaiting assignment or review
In-Progress	The SIR was assigned and that resource is currently working on its resolution
Resolved	Corrective action was completed and the SIR is awaiting validation
Closed	Corrective action has been completed and validated
Postponed	The team agrees to postpone the corrective action on this SIR to a later release
Duplicate	The issue identified in this SIR is identical to another open SIR, and the resolution to the duplicate SIR will be tracked through the original open SIR

Table 3 - SIR Status Levels

### 2.2.4.3 Re-test SIR Fixes

As SIR fixes are completed, the script in which the SIR was encountered will be re-tested. These SIRs may have originated from an earlier pass, the same pass, or from another script that has one or more components in common. The System Test Team Lead will coordinate with the Implementation Team Lead to group related issues together, thus minimizing the delay of executing the next pass of the test script.

SIR fixes will be re-tested using local copies of the original input files, data, etc. After conducting a re-test, SIRs that fail are reopened and newly identified SIRs are logged in the SIR database. Additional passes will continue to be conducted until the exit criteria are achieved for each phase.

Whenever a test condition or test script step does not return the expected result, the tester will log the discrepancy between the expected and actual result as a SIR. SIRs will be logged and tracked in a central repository, or SIR database, using eProject. The SIR database will include a long and short description of the issue (including the actual and expected results), its severity and status, the affected test area or module, the test phase, test pass, and tester.

## 2.3 Test Plan

### 2.3.1 Process

The process for developing system test scripts will be as follows:



- 1) Identify Test Scenarios from Requirements Definition documents (e.g., Requirements Traceability Matrix Design and Use Case Specifications).
- 2) Write Test Scripts for test execution.
- 3) Identify Test Conditions.
- 4) Map Test Conditions to Requirements.
- 5) Create baseline Test Data.
- 6) Conduct Test Readiness Review session.

### 2.3.2 Test Scenarios

Using the Use Case Specifications and the Requirements Traceability Matrix, test scenarios will be identified to demonstrate how users perform specific tasks in the XML Registry and Repository. Use Case specifications describe functionality in terms of user behaviors to be satisfied by the system. Therefore, the test scenarios will describe the user interaction with the system required to perform a specific business function.

### 2.3.3 Test Scripts

The Test Team Lead will work with the test team to create test scripts. Reports can then be generated and given to the test team showing the conditions that a script should test, and the tester will write or modify a script to cover those conditions. Scripts should be consolidated, wherever possible, to maximize test scenarios and minimize staff hours.

Test scripts are executed within each test cycle. The scripts test the basic system functionality derived by the system level requirements. Test scripts are documented in terms of a business function or technical requirement and serve as the basis for creating the test conditions.

### 2.3.4 Test Script Standards

Please see Appendix A: Sample Test Script for an example of the XML Registry and Repository system test scripts. Each test script contains the following information:

- 7) Script Name and Number - assigned by script writer.
- 8) Description - describes the test script.
- 9) Created By - name of script writer.
- 10) Tested By - person who is performing the actions for a test condition.
- 11) Test Date - written on the printed script by the tester.
- 12) Prerequisites - reference data or transactions that must exist prior to script execution.
- 13) Use Case(s) - references use cases covered.

For each step within the script, the following information is shown in table format:

- 14) Step number - automatically assigned by the script template.
- 15) Step Description - describes what the tester needs to do for each step of the condition.



- 16) Action – describes detailed instructions for how to accomplish the step.
- 17) Input – describes the data that needs to be entered into the user interface at each step.
- 18) Expected Results – describes how the system should respond to the action.
- 19) Actual Results – describes what happens when the tester performs the action.
- 20) Pass/Fail – describes whether the expected and actual results match for a condition. Pass means that the expected and actual results match. Fail indicates the expected and actual results did not pass for a condition.
- 21) Priority – describes the conditions that fail, the severity of the defect and the priority for development to fix the defect.
- 22) Comments – provides an area for notes about the test condition.
- 23) SIR # - identifies the related SIR # for a defect logged against the failed test condition. If the condition fails for more than one reason, multiple SIRs will be logged.
- 24) Requirement # – traces the test condition to a system requirement.

### 2.3.5 Test Conditions

Test conditions provide the ability to test the system functionality under specific circumstances to be performed in production. The test conditions will be developed so that they are repeatable and reusable. Test conditions must be repeatable so that errors can be reproduced and retested under the same circumstances.

Test conditions will be written using the detailed functional and technical requirements from the Requirements Definition phase. These conditions will be grouped into scenarios. The scenarios will contain a related set of conditions, with the condition mapped to the Use Cases and requirements.

Finally, test scripts will be prepared during the Application Development phase that define the steps, input data, and expected result to test the scenarios. A single scenario may be split across multiple test scripts, or multiple scenarios may be included within a single test script.



## 2.4 Requirements Traceability

The test scripts and test conditions will then be mapped to the Requirements Matrix and Use Case Specifications.

### 2.4.1 Baseline Test Data

Baseline data must be defined and loaded into the system prior to testing. This data includes:

- Core Component Test Data – search, upload, download, etc.
- User Setup Data – user roles and responsibilities, etc.

Test data will be created as representative sample data to use when executing the test scripts. Before test execution can begin, baseline test data will be created to enable the testers to execute the test scripts. The 'Input' column will represent data entered into the XML Registry and Repository while executing the test scripts.

It is still necessary to identify the test entry data needed in order to execute the test scripts when the scripts require the tester to create data from scratch. This type of iterative testing ensures that the data being entered is valid and allows the tester to complete an entire Test Cycle using the same data entity.

## 2.5 Test Environment

A separate test instance will be set up for the test team. All successful component-tested and assembly-tested code modules will be migrated to the test environment prior to test execution.

The first test instance will be the current testing instance used to execute each System Test Pass.

### 2.5.1 Test Execution

The System Test team will execute the scripts according to the step-by-step directions on each script. Scripts that pass will be signed off and stored in a binder. Scripts that fail should be dealt with as described in the following section.

### 2.5.2 Test Passes

A test pass is defined as one full test execution cycle. Development remediation activities are conducted between passes within the same test phase. Additional passes will be run within a test phase until the test meets the defined exit criteria.

Test passes will be performed in steps. A pass contains the following steps:

- 1) The tester executes the test scripts.
- 2) The tester enters a SIR when results from the test deviate from what is expected.
- 3) The Test Team Lead notifies the development when the test pass is complete.
- 4) The Development Team Lead runs the SIR report at the end of the pass.
- 5) The Development Team Lead performs an impact analysis on the SIR list and reports his/her findings to the Test Team Lead.



- 6) SIR fixes are completed and moved into the testing environment according to Configuration Management guidelines.
- 7) The Development team updates the SIR database with the SIR status information.
- 8) The Development Team Lead notifies the Test Team Lead that testing may resume.
- 9) The test team begins a new pass and continues this process until the exit criteria have been achieved for the phase.

If a critical error occurs within a test pass that prevents the entire set of scripts from being executed and the fix requires lengthy remediation, the pass will be considered complete and the SIR resolution phase will begin. Once the SIR fixes are migrated into the test environment, a new pass will begin. The beginning of a new pass will be at the discretion of the System Test Team Lead, and a new pass may begin with unresolved SIRs outstanding. Test passes will continue until the scripts for that pass have been executed and the exit criteria have been satisfied. The amount of time necessary to complete a test pass will vary depending on the particular area of testing being performed.

### 2.5.3 Identification and Resolution of Issues

The following process will be followed for identifying and resolving problems encountered during testing.

- 1) The Tester discovers a discrepancy with a script and works with another tester to determine if the problem is with the software, the test script, or some other source. This will serve as a form of triage to identify true issues, and to not bother the programmers with issues not related to the code.
- 2) If the test team determines that the problem is with the software, the tester will log the issue in the SIR database with supporting detail including, screen prints or sections of the requirements document. The tester will document the problem by entering "Fail" in the Pass/Fail column of the script, at the step where the script failed. The tester should also capture the date and time the failure was noted.
- 3) The SIR database should be updated to reflect the script failure. In addition to tracking software failures, the team will also track failures related to undefined requirements, faulty scripts, database errors, and "other."
- 4) The test Team Lead will notify the Implementation Team Lead of any failed test scripts in order to assign the problem to a developer to correct it.
- 5) The developer should work with the tester in the development environment to ensure that the software change fixed the problem.
- 6) The Implementation Team Lead should notify the Test Team Lead when the changed software has been migrated to the test environment. This should be done following the FSA-defined process.
- 7) The tester will confirm, in the test environment, that the software change fixed the problem.
- 8) The tester will note, in the Pass/Fail column of the script, the date and time that the step of the script was successfully executed, and continue with the script.

During this process, if a problem is identified in testing, the System Test and Implementation Team Leads will make a determination on the need for and scope of regression testing.



The key to quickly resolving software defects is direct communication between the tester and the developer who is fixing the problem. The Implementation Team Lead and the System Test Team Lead should be kept in the loop, but must not become a bottleneck.

## 2.6 Development, Test, and Production Environments

The following diagram depicts the environments that will be required for the XML Registry and Repository project.

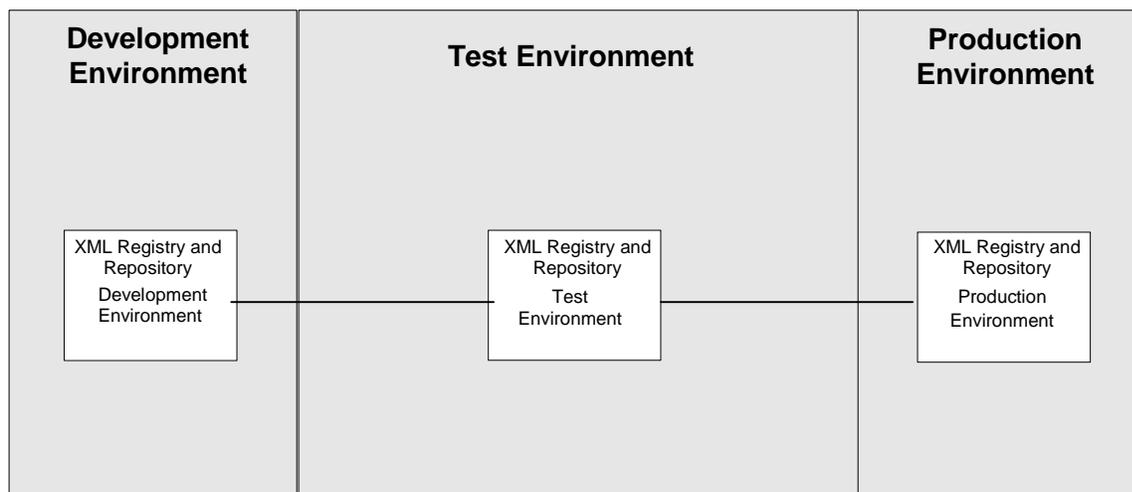


Figure 2 - Development, Test, and Production Environments

The following environments will serve different purposes:

- *Development Environment* - The Development environment will be created using the architecture for XML Registry and Repository and will be used for unit and assembly testing.
- *Test Environment* - The System Test environment will be created using the architecture required for the XML Registry and Repository and will be used for system testing. This environment will mirror the production environment.
- *Production Environment* - The Production environment will be created using the architecture for the XML Registry and Repository and will be used as the XML Registry and Repository production environment for the Deployment and Support project phases.