AOC Source Control Tool - Requirements	<Enter Vendor Name(s) Here > In the **Response** column, indicate how your product complies with the AOC requirements using the following response key. **4**=Out of Box (in Comments/Notes indicate version). **3**=Future release (in Comments/Notes, indicate version, planned release date). **2**=Gap: Requires customization (in Comments/Notes, indicate estimated cost of customization). **1**=Gap: Other (in Comments/Notes, explain your proposed tool). **0**=Not available in this tool. Tip: Move cursor over red triangle in the upper right corner of any cell in the Response column to view response key.

From any AOC location, the Source Control Tool must:

Category		Requirement	Response	Comments/Notes
I. User Interface				
	1	Provide encrypted web-based user interface as well as desktop client interface.		
	2	Provide an intuitive user interface so the average user can use the system without having to get specific training.		
	3	Adhere to section 508 standards for disabilities. Standards shall apply to both test repository output and to the solution interface used by administrators and content developers (1194.21 and 1194.22).		
	4	Provide built in viewer that clearly identifies differences between versions.		
	5	Provide ability to view differences between more than two versions of the same file.		
	6	Provide ability to view differences between different versions of single or multiple files (e.g. branched, folders, & sub-folders).		
	7	Support viewing highlighted differences for multiple file formats including but not limited to Java, C, C++, VB, SQL, XML, HTML, simple text, etc., and binary files (compiled objects).		
	8	Provide version history of files including branching and labeling.		
	9	Provide ability to set selective notification on files.		
	10	Provide ability to plug into an existing user interface, such as Windows File Explorer.		

	11	Solution must be WYSIWYG - what you see is what you get.		
	12	Supports multiple browsers, including IE 6, IE 7, and Firefox 2.		
II.	***Functionality***			
	1	Support version control for multiple projects.		
	2	Support multiple versions of code.		
	3	Support different testing cycles as defined in test categories (see Test Category Definitions worksheet in this workbook).		
	4	Provide ability to automatically merge differences from different versions (single & multiple files).		
	5	Provide interface for resolving merge conflicts.		
	6	Provide role-based permissions for managing user rights.		
	7	Provide permission-based ability to override system constraints (breaking code locks).		
	8	Provide email notification mechanism to watch for changes to particular files or branches.		
	9	Provides ability to compare: file/set of files with previous version of file/set of files in the branch file/set of files with any file/set of files in another branch file/set of files with unchecked-in file/set of files.		
	10	Import files from existing version control systems, including their complete modification history.		
	11	Provide ability to relate multiple files to an activity or task for checkout.		
	12	Provide ability to support atomic operations (Check-in or Check-out) on file sets and partial file sets (set of files associated to a task or an activity).		
	13	Provide ability to summarize as well as provide detailed differences of a directory tree.		
	14	Provide users with visual feedback indicating a file or source code's status (who has it checked out, how long it has been checked out, etc.).		
	15	Provide ability to query modifications made by a user or group of users.		
	16	Provide ability to undo committed modifications.		
	17	Support industry standard file formats including binary files.		
	18	Provide ability to refresh partial trees of the source to developer’s machine.		

	19	Support multiple branches of a tree.		
	20	Support merging of branches.		
	21	Provide ability to create a directory tree along with its entries using directory structure on user's machine.		
	22	Provide ability to selectively lock branches, directories, individual files, and etc.		
	23	Support a well defined branch promotion process.		
	24	Provide scripting capabilities.		
	25	Provide ability to export files with complete modification history to leading vendor formats.		
	26	Support the ability to work off-line, download files or source code and re-synch with the repository once online.		
III.	***Reporting***			
	1	Provide out-of-the-box reporting.		
	2	Provide access to view data and statistics across multiple projects, vendors, implementation sites, applications etc., i.e. "slice-n-dice" data as needed (ad hoc reporting).		
	3	Report with number of files by file type.		
	4	Provide ability to schedule and distribute reports automatically via email.		
	5	Provide ability to create reports based on any point in time (current or past).		
	6	Export reports into Microsoft Office, PDF and HTML formats.		
	7	Provide ability to produce a release note from developer comments on submitted files.		
	8	Provide ability to view the current state of the repository by project (Who has the possession of the files, for how long, and so on).		
	9	Provide ability to view users and objects/projects for which they have access.		
	10	Describe the level of expertise required to create custom or ad hoc reports.		
	11	Provide a query building tool that supports formulation of complex queries.		
	12	Log all commitments that are undone (related to Functionality requirement #15).		
IV.	***System Requirement***			
	1	Provide comprehensive online help.		

	2	Adhere to section 508 standards for disabilities. Standards shall apply to both test repository output and to the solution interface used by administrators and content developers (1194.21 and 1194.22).		
	3	Run on Solaris, Linux, or Microsoft Windows operating systems.		
	4	Easily scale to accommodate hundreds of concurrent users.		
	5	Configure to support master/master - master/slave repositories where projects can create their own remote repository that periodically can be synchronized to the main enterprise repository.		
	6	Provide license(s) that allow for distributed use of all components for AOC employees, contractors, associated vendors and partners that are physically located in different locations.		
	7	Provide mechanisms (e.g. compression) to perform efficiently over the WAN and LAN.		
	8	Provide ability to work with multiple character sets.		
	9	Must support 3-tiered infrastructure (presentation, application/middleware, database).		
	10	Provides support for compressed storage of source code in repository.		
	11	Function in a cluster configuration (Active/Active and/or Active/Passive).		
V.	***Enterprise Infrastructure & Security***			
	1	Support all components to be maintained and stored on a centralized repository.		
	2	Provide role based access control, allowing different levels and types of access to user based on user's role.		
	3	Enable individual security to be defined at the folder, subfolder, file and process level. This may override role based security, so that a file or source code might have "default" security which can be restricted at the user or administrator's discretion.		
	4	Integrate with LDAP, such as MS-Active Directory and/or Computer Associate eTrust (Netegrity/SiteMinder) for single sign-on.		
	5	Be JSR-168 and JSR-170 compliant.		
	6	Provide mechanisms (e.g. compression) to perform efficiently over the WAN and LAN.		
	7	Be compatible with all firewall and proxy systems.		
	8	Provide ability to be monitored by standard application monitoring software, e.g. SNMP.		

	9	System shall support encryption of data between the system and the user's computer through standard web encryption (SSL). Encryption algorithm used shall be based on Advanced Encryption Standards (AES).		
	10	Provide ability to have multiple administrative or delegated admin users.		
	11	System shall allow administrators to control who may set, change or override document security settings.		
	12	Provide ability to notify authorized users of critical events.		
	13	Provide ability to archive as well as restore from archives.		
	14	Provide ability to archive partial history of files. (As modifications and labeling happen the archives keep growing. This will help in archiving infrequently used old modifications to storage)		
	15	Support backup and restore.		
VI.	***Integration***			
	1	Integrate workflow notification via email (for example, SMTP).		
	2	Integrate with popular Integrated Development Environments, and testing tools. Including but not limited to ColdFusion Studio 5, Visual Studio, Dreamweaver, Eclipse, TextPad, WebLogic Workshop.		
	3	Integrate with the popular, industry standard build tools.		
	4	Provide ability to be tightly coupled and integrated with other testing tools.		
	5	Cross-link testing suite, issue tracking and code control systems to provide holistic view of dependencies.		
	6	Integrate workflow notification via email (e.g. SMTP).		
VII.	***Vendor Technical Support***			
	1	Provide a toll free number for AOC production support.		
	2	Provide staffed phone support (as opposed to an automated system) to deliver production support.		
	3	Provide tiered product support levels to choose from during normal business hours.		
	4	Provide off-hour support with advanced AOC management notice (including evenings and weekends, in person or via phone, as needed for special activities, such as conversion, critical test times for projects).		
	5	Provide an advance copy of a fixed Release Calendar to AOC Management.		

	6	Provide written confirmation (paper or electronic) for all planned releases or emergency patches (two (2) weeks in advance for planned releases and three (3) days for emergency patches). Release Notes must include known defects.		
	7	Provide software upgrade guidelines (for example, what constitutes a major release, minor release or interim release) of the product versions.		
	8	Provide software licenses that may be moved easily from one physical location to another.		
	9	Provide online tech support (chat style).		
	10	Documentation – user operational manual and training.		

Item #	AOC Test Category	AOC Definitions & Typical Order of Execution	Typical Resource Skill set Required	Application Environment
1	Unit Testing	These tests ensure that individual components (or "units") function properly. Most often this is performed in some sort of test framework in the Development environment – which imitates the other components with which this component must interact.	Developer	Development
2	Integration Testing	As the components are integrated together, these tests ensure that they work together as expected. This is often a multi-staged integration; in which trenches of the components are rolled together to make larger components … and so on, until the whole application is integrated. The early stages of integration might happen within the Development environment. The major part of this testing is performed in the Testing environment; in which the larger "conglomerate" components are integrated.	QA Analyst / Developer	Development/Testing
3	Functional Testing	A level of testing that starts in the Development environment as unit testing then moves to the Testing environment. Testing ensures the required functionality of applications passes test plans with use cases based on business requirements. Use cases are tested, including its variants and exceptions.	QA Analyst	Development/Testing
4	Enterprise Integration Testing	Enterprise testing ensures that the underlying hardware, databases, and other third-party software or services are installed correctly and function properly together. This test phase will ensure the next stages of test can be executed and all the monitoring tools and scripts run properly.	QA Analyst / Developer	Testing
5	Regression Testing	This is a (sub-) set of Functional tests and Stress tests that can be repeated to verify that the application is still working after some change has taken place (e.g. code changes, firmware patches, etc.). Ideally, these tests are executed with automated testing tools for both Functional and Stress testing. The ease of re-application of these tests means that they can be used in whichever pre-live environment and whenever they are required. Usually the Functional tests are developed for the Testing environment and then they are captured in automated scripts to form Regression tests that are repeated in Staging or in Testing as required. Both Functional and Stress tests are combined in these tests.	QA Analyst	Testing/Staging
6	Stress Testing	As the name implies, tests are applied to stress an application, its environment and data systems, etc., under simulated user volumes that impact performance and recovery. The idea is to progressively put more stress on an application (increasing number of users, transactions, size of the data requests, etc.), until they break, no longer perform or are stressed to capacity. The tests are used to determine the stability of a given system or entity. It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results. Some testing may occur in the Testing environment, yet majority happens in the Staging environment. Exceptions are made to test in a Production environment before it 'goes live' when this environment is larger than Staging.	Developer	Testing/Staging

7	Load Testing	Testing of applications and infrastructure to measure the effects of a system's current and expected user capacity based on the current configuration, the AOC's performance criteria, Service Level Agreement or Vendor Contract, and projected usage. The results determine a 'load baseline' for current user capacity and projects what will be required to accommodate increased user capacity over the course of time. The majority of testing happens in the Staging environment. Exceptions are made to test in a Production environment before it 'goes live' when this environment is larger than Staging.	Developer	Staging/Production
8	Performance Testing	Tests determine if the system will meet the performance terms and conditions stated in the AOC's Service Level Agreement or Vendor Contract. Tests verify that the expected maximum use of the system falls within tolerance, that the system fails in a reasonably graceful manner, and that performance is acceptable. Also, performance testing is used to test response times for schema validation, overall latency/response time, availability. Majority happens in the Staging environment. Exceptions are made to test in a Production environment before it 'goes live' when this environment is larger than Staging.	Developer	Staging/Production
9	Fail-over Testing	To meet a 99.9% up-time, many hardware components are doubled up. These redundant components are useless however, unless they can take over in a predictable and acceptable way. These tests show how the redundancy works and help identify any additional steps necessary to recover. These tests should be included in the Regression test suite as appropriate (e.g. if redundant components are in Staging). Usually, this test requires that an active server is unplugged (or otherwise assaulted) to verify that its (hitherto) passive counterpart takes over as predicted. In a load-sharing scenario (active:active), the test is to verify that the remaining server takes over the whole load. This test should happen in Staging (or in Production if the Staging environment is scaled-down).	Developer	Staging/Production
10	Disaster Recovery	This test is typically performed periodically to ensure that business/operations can continue in the event that the primary data center/systems implementation is down due to a major disaster, e.g., earthquake, flood, fire, etc. The terms laid out in the disaster recovery (DR) specification are tested; this environment will not typically provide the same performance as the original environment. Tests used for the DR environment will need to be downsized to test the DR environment.	Developer, Business Subject-Matter-Expert, QA Analyst	N/A
11	User Acceptance Testing (UAT)	Acceptance criteria are typically defined in the design phase. As the previous test categories complete in order to 'accept' the progression of development and testing, this level of testing occurs at the end of the development phase, typically as a review gate or milestone test to pass prior to product launch. Various sub-sets of the above tests may be agreed to up-front in order to accept that various stages of the development are complete.	Business Subject-Matter-Expert, QA Analyst	Staging/Production
12	Cut-Over Testing	This testing is usually a subset of Regression testing. It provides for a sub-set of Regression tests in a new Production environment before giving the application approval to 'go live.' Care is taken not to change any 'live' production data, or to ensure any changes to live data are reversed. Cut-over testing occurs in Staging and also gives assures the network port from Staging to Production happened correctly.	QA Analyst; Business Subject-Matter-Expert	Production

13	End-to-End Testing	Testing that involves test monitors that perform end-to-end tests of each application of a system (typically from the CCTC to a court). This level of testing is to determine if and when any function or system integration point breaks prior to system implementation. Tests for the application and network through the Court's network to the CCTC network are executed. Functional and non-functional testing are included ensuring security requirements are met. Output facilitates ongoing monitoring to ensure all production systems are working as expected.	Developer	Staging/Production
14	Product Acceptance Testing (PAT)	This is the set of tests from all tests above that determines whether the product is viable based on the agreed requirements. While delivering a product to meet both functional and technical requirements is the goal, these requirements can only be validated through PAT. Acceptance criteria, therefore, tends to tag on more flexible statements as well as referencing rigorous testing. For an application provided by a third party, these tests allow the AOC to decide to go ahead or stop a new application. Typically a 0:0:10 criteria is used, which allows 0 grade 1 errors, 0 grade 2 error and up to 10 grade 3 errors. For an internal development, these tests determine whether we are ready to move onto the next stage of implementation.	Business Subject-Matter-Expert, QA Analyst	Staging/Production
15	Localization	This tests the various concurrent builds of an application individually. Some applications are customized (localized) for different groups of users. Common functionality can be tested in common – per the above tests – but localized functionality should be tested in each build as appropriate. Several versions of Regression tests are required: to test that a localized version maintains its core functionality and to test the "delta" of each localized version.	Developer, QA Analyst	Testing, Staging/Production
16	Turn-up Testing	When a court is first given access to an application housed at the CCTC, these tests are used to determine whether the court has suitable access to the application. Various tests can be used, from: • Testing the bandwidth available to/from the CCTC • Running the Regression Tests from the court to determine whether all functionality works • Running a Load Test from the court (which may be covered in a good Regression Test) to determine the performance.	Technical SME	Staging/Production
17	Data Conversion	When a new court is on-boarded, data from previous systems is often ETL'd into the new application environment. Data conversion testing determines whether that data has been uploaded correctly. Typically, this involves a manual process in which information is accessed via both the previous and the new applications to see whether it matches. Since the new environment may require different data architectures, direct database comparison is not always viable and the two versions need to be compared by accessing data in either version via its user interface.	Developer, QA Analyst, Business Subject-Matter-Expert	Staging/Production
18	Usability	A representative group of business users use the system to see how it functions relative to their expectations. Typically the testing occurs after training as the users try out features of the application intuitively and have the opportunity to validate that their requirements have been interpreted correctly. Usability testing gives the Developers valuable insight from the end users prior to UAT and occurs in the Testing environment.	Business Subject-Matter-Expert, QA Analyst	Testing

19	System	This is the final stage of (non-enterprise) integration testing, in which all the components have been integrated and the whole application (except links to other enterprise functionality) is available.	QA Analyst	Testing
20	Test Data	Testing an application invariably changes the environment in which the test occurs. Steps can be taken to script “reset” tests that return the environment to the original state. E.g., if a test changes a user’s password, the equivalent reset test must change it back (or cycle through the necessary iterations before it can return the original). Also, another approach is to install a fresh set of test data before each test commences. In this way, all the parameters used in the test cases can be re-used, since they relate to this clean version of the data. This process involves: 1. Backing up any live data 2. Loading the Test Data 3. Running the test 4. Analyzing any end-state data 5. Re-loading the original live data. One of the reasons for creating cut-over tests, rather than running a full Regression test, is to remove any tests that change data that can’t be reset by running another test (e.g. it may not be possible to un-approve a workflow that was approved in a test). This allows the cut-over test to be run in a live environment without compromising the data and without havin	Developer, QA Analyst	Staging/Production