Asset Tracking and Revision Control for Automated Water/Wastewater Control Systems

Blair Sooley, M.B.A., P. Eng.

Trihedral Engineering Limited
1160 Bedford Hwy
Bedford, NS B4A1C1
(*correspondence: bsooley@trihedral.com)

KEYWORDS

version control, configuration management, SCADA, redundancy, configuration, risk mitigation, asset tracking

ABSTRACT

Utilities realized years ago the value of maintaining good records. Systems run for decades, yet many changes inevitably occur during their natural lifespan. Historically, these changes were managed with paper systems or in the minds of key employees. Information was stored across multiple systems again linked by the living-knowledge of staff. As the amount of information expanded and key personnel moved on, gaps appeared which resulted in costly "discoveries" as older systems were modified.

Today the common solution for dealing with physical assets is to deploy a GIS system. This tracks assets and key properties, but what about changes to the SCADA software system? These systems also undergo change to reflect the physical systems they monitor. Over time, many people will work on the application both internal and external. Yet the reality is that this is not typically tracked in an auditable and recoverable fashion. There may be the shoebox of past versions after an upgrade, but they do not detail what changes were made; only that new and old versions exist. This may be inconsistent with how the utility manages changes to its other assets, but is the accepted norm until a new technology offers substantial improvements.

It is now incumbent on SCADA suppliers to accept the challenge and develop integrated tools to record all incremental system changes. Ideally, this audit trail will also facilitate a rollback to a previous version of the application. This will provide the benefits of stability and the ability to manage accidental or deliberate internal sabotage of an existing system. As more systems become networked, there is a challenge in managing complex applications while guaranteeing system integrity. System maintenance and maximizing uptime will be the next challenge facing the SCADA world.

Record Keeping and Configuration Issues Increase the Potential for Loss

Water and wastewater utilities understand the value of maintaining good records. Systems of all types may run for decades, yet many changes will inevitably occur during their natural lifespan. Historically, these changes were managed with paper systems or in the minds of key employees. Today the common solution for tracking physical assets, such as piping and process equipment, is to deploy dedicated systems. For example Geographic Information Systems (GIS) track location and manufacturer data while Computerized Maintenance Management Systems (CMMS) track maintenance activity. But what of the supervisory control and data acquisition (SCADA) systems that monitor and control this equipment? These too are assets, and they change frequently to reflect not only changes in the physical assets they monitor, but also to address management initiatives, such as process transparency, efficiency analysis, and improved reporting. Over time, many people both external and internal may change the SCADA system. There may be the shoebox of past versions and a bookcase of new documentation, but a traceable version history of the SCADA assets rarely exists.

Indeed, good record keeping would be unnecessary if the SCADA system's validity could be always assured. Validity is the extent to which the assets, in this case the SCADA system and its components, provide the configured operational functionality as intended. Improper system configuration can result in downtime, reporting inaccuracies, product loss, equipment damage, or even injuries. These effects are not confined within the organization, where an incorrect recipe can ruin a batch of product; rather, they can reach the consumer, where incorrect measurement can result in product quality issues and adverse public health effects. Additionally, with the advent of cyber-based criminal activity, the potential for malicious, undocumented SCADA configuration has greatly increased. To mitigate the potential for these problems, some industries such as the pharmaceutical and food manufacturing enforce strict control on SCADA configuration management procedures, while others rely on the industry participants to define their own procedures. The latter method's weakness is that industries with no such regulation often have little guidance on configuration management and, as such, view it as unnecessary overhead.

There are many ways that SCADA configurations can occur. Regardless of the reasons, organizations need to be comfortable that interruption in SCADA operations and incorrect system configurations can be quickly diagnosed and addressed. This paper focuses on SCADA software assets, methods to protect them from incorrect configuration, and the use of comprehensive change tracking and version control for use in problem resolution.

Unintentional versus Intentional Configuration Problems

As noted previously, improper configuration falls into two broad categories; malicious and unintentional. Malicious activity is of greatest concern as it is the intention of the offender to cause damage by affecting process measurement, interfering with process operations, or introducing viruses for data gathering purposes. There is a growing body of documented cases of cyber attacks on SCADA systems. The mandate of the British Columbia Institute for Technology's Group for Advanced Information Technology (GAIT, 2012) states that "industrial production and human safety [are] at risk from cyber attacks" on SCADA systems. Indeed, a paper by Hildick-Smith (Hildick-Smith, 2005) of the SANS Institute, a global cooperative research and educational institute for security professionals, suggests such changes can come in a variety of forms; from malware, hackers, terrorists, and even insiders with intimate knowledge of the process. Such activity is often hard to detect in the short term and can be extremely difficult to trace. Hildick- Smith's research surmises that as of 2004 there were approximately 25 recorded industrial cyber incidents per year with the actual number of incidents potentially closer to 100 per year. Of the documented incidents, approximately 70% were perpetrated by outsiders.

These attacks affect all types of critical infrastructure. For example, the Stuxnet virus, discovered in June 2010, was reasonably categorized as an act of terrorism. This virus attacked at the hardware level, by leveraging weak security in Siemens' controller programming software. Many regarded this as an attempt to target the Iranian Nuclear industry. Of equal notoriety is the incident perpetrated by Vitek Boden, a highly knowledgeable inside contractor on the Maroochy Shire Council sewer system in Australia. This attack was purely malicious and resulted in approximately 260,000 gallons of raw sewage being released into the environment.

The second category, unintentional improper configurations, are often the result of an intentional system improvement that has unintended negative consequences. Careful procedures and testing help to reduce the occurrence of such issues but even so, not all possible consequences can be tested.

During the Christmas holidays of 2009, a Florida water utility [name withheld] was running with minimal staff. A member of the operations team with relatively modest SCADA training attempted to teach himself to configure the utility's SCADA software on the production system. Though well intentioned, the SCADA system configuration was altered to such an extent that communications with an entire network of remote water management infrastructure was lost. The operator, with the assistance of his SCADA supplier's support team, was able to rebuild the required functionality by combing engineering design documentation and hardware manuals. This process took most of a day, resulting in a loss of important monitoring and control functionality as well as a significant gap in historical data collection.

Additional Problems

The potential for such issues grows in significance as the SCADA system ages. Most SCADA systems are initially configured to meet a specific set of goals. During the initial deployment stage, configuration is often provided by either an outside contractor or a dedicated internal team. At this point, the system functionality is generally well defined and documented.

Few systems stay stagnant, however, and most evolve as a result of external pressures and changing organizational goals. New contractors make configuration changes during various expansions and internal teams change. Minor changes can feel like a burden when manual documentation and strict procedures are required. As a result, these minor changes often go undocumented.

Process uptime also plays a significant role. Changes are made quickly to reduce process interruption and SCADA updates are often seen as ancillary to the addition or reduction of new processes and equipment. Even when strict testing and commissioning procedures are applied to find problems before introduction to the production environment, it is often not possible or feasible to capture all use cases, nor is it reasonable to expect even the most rigorous analysis will uncover all potential effects.

The Solution - Security, Backup, and Version Control

When things go right, few people care. When things go wrong, a comprehensive change management history is invaluable in problem diagnosis and resolution. The solution seems simple. Secure the SCADA software so that no unauthorized configuration changes can be made, keep an emergency backup of the system configuration on hand, and ensure authorized changes are well documented. Yet often these steps are minimized or overlooked altogether as they require strict adherence to procedure and can be time consuming and thus expensive to manage.

A properly implemented security scenario can go a long way to protecting the system. However, no application is immune to improper configuration; even one that is physically separated from external influence, not to mention the negative impact such architecture can have on business system interconnectivity and remote access. Malicious activity by internal users is always a risk and even the most authenticated users can make honest mistakes.

Emergency backups provide assurance that the SCADA software can be restored to a known good state. It is a time-tested solution for rebuilding damaged systems after catastrophic failure. However, creating a useful backup and restore system requires strict adherence to procedure and often IT expertise to configure and manage. Additionally, many backup systems require applications to be taken offline during the backup process. This limits the backup process to a periodic occurrence; run daily, weekly, or monthly based on storage and system availability. As a result, minor changes may not be backed up for days or possibly even weeks. Restoring a backup version, then, may result in a loss of all changes made since the backup utility was last run.

Finally, comprehensive change management documentation is very effective for diagnosing issues; providing details of who did what, when it was done, and often providing context as to why it was done. The complexity and large numbers of files in SCADA systems makes comprehensive change documentation very involved, requiring strict dedication to procedure to capture exactly what parts of what files were changed. This is expensive and slow, especially if documenting every minor change. This process is eased somewhat by the use of electronic version control systems. Many such systems maintain a repository of application files, automatically manage versions, and provide a means to define differences between versions. Unfortunately, in some such systems, the version repository exists in only one place unless manually copied, which may lead to loss of the entire version control repository in the event of disasters such as fire, flood, or other weather events. Further, because the version control system does not form an integrated part of the SCADA system, procedures require significant manual action and oversight.

Reduce Complexity and Procedural Demands with Integrated SCADA Configuration Management and Version Control

Many utilities address these requirements through third party software solutions from several manufacturers, introducing complexity and administrative overhead. However, developments in SCADA software technology now allow change management and version control to be integrated components of the development environment. Such comprehensive packages eliminate compatibility issues and leverage the tight integration of components to reduce the administrative burden of strict procedural adherence.

Given the benefits, it is suggested that utilities look for the following integrated components in evaluating SCADA platforms:

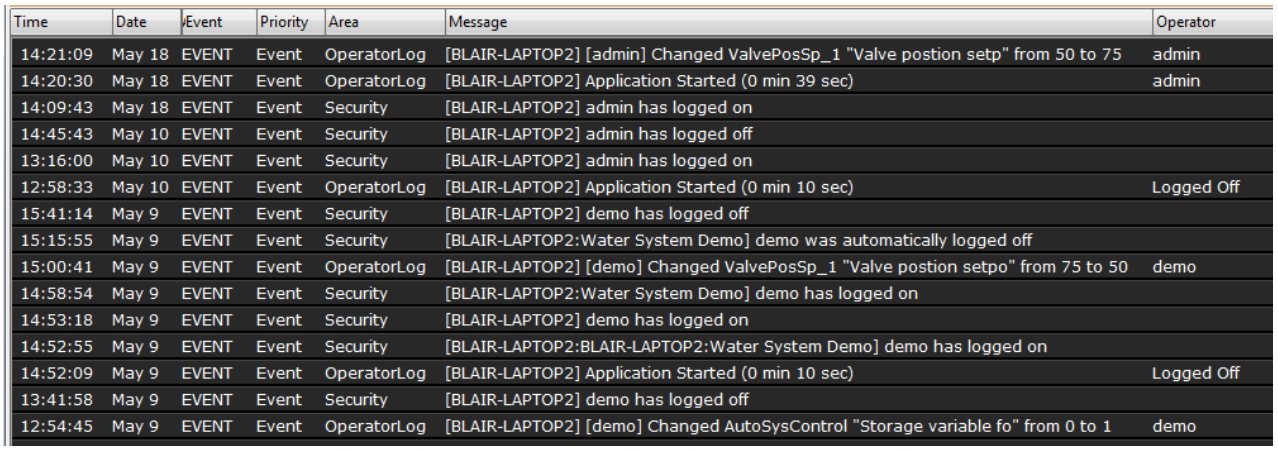
- Rules- and roles-based security management
- Real-time recording of security account activity and operational events
- Multi-developer environment
- Automatic recording of configuration changes with a complete, traceable version history
- Online configuration change testing and deployment
- Automatic version control with a distributed repository

The features provide a secure, comprehensive, high-speed development environment, ensuring one or more authorized users can affect operational and configuration changes with automatic recording, strict procedural enforcement, and disaster protection. Moreover, this functionality is implemented with minimal manual intervention.

The security system forms the cornerstone of these tools, allowing operational and configuration change privileges to only specific users and further restricting the subset of users who can alter accounts. The use of roles and individual privileges allows unlimited flexibility on a per-account basis rather than more restrictive level-based security paradigm.

Account log-in/ log-out activity should be automatically recorded along with any operator actions. Operator actions are events that affect the process, including changing set-points and equipment control. All such events should be logged with a time/date stamp, the authorized user who initiated the action, and the workstation from which the action was initiated.

Similarly, changes to any application configuration files should be automatically recorded at the time they are implemented. This process ensures all changes are captured while simultaneously enforcing strict change recording without human intervention.



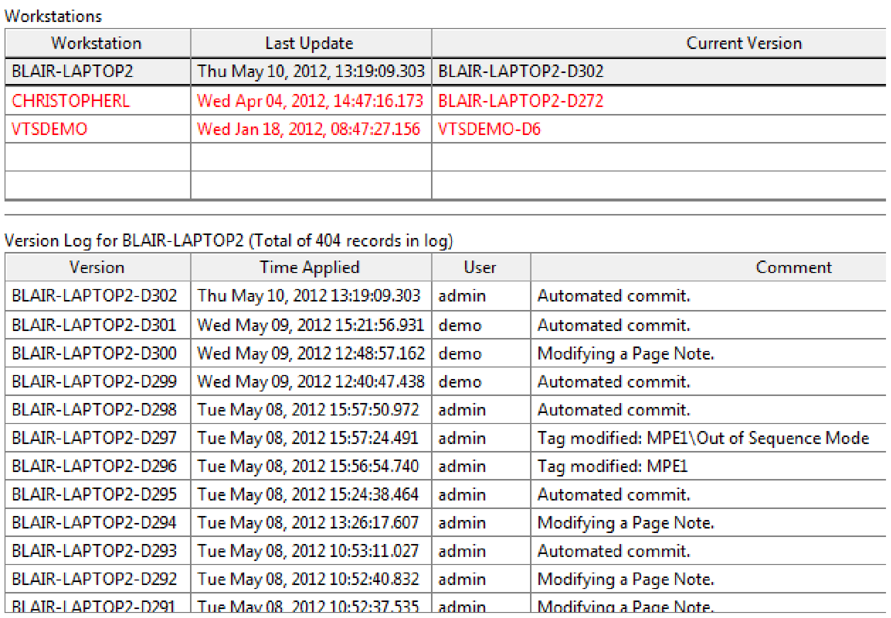
Time	Date	Event	Priority	Area	Message	Operator
14:21:09	May 18	EVENT	Event	OperatorLog	[BLAIR-LAPTOP2] [admin] Changed ValvePosSp_1 "Valve postion setp" from 50 to 75	admin
14:20:30	May 18	EVENT	Event	OperatorLog	[BLAIR-LAPTOP2] Application Started (0 min 39 sec)	admin
14:09:43	May 18	EVENT	Event	Security	[BLAIR-LAPTOP2] admin has logged on	
14:45:43	May 10	EVENT	Event	Security	[BLAIR-LAPTOP2] admin has logged off	
13:16:00	May 10	EVENT	Event	Security	[BLAIR-LAPTOP2] admin has logged on	
12:58:33	May 10	EVENT	Event	OperatorLog	[BLAIR-LAPTOP2] Application Started (0 min 10 sec)	Logged Off
15:41:14	May 9	EVENT	Event	Security	[BLAIR-LAPTOP2] demo has logged off	
15:15:55	May 9	EVENT	Event	Security	[BLAIR-LAPTOP2:Water System Demo] demo was automatically logged off	
15:00:41	May 9	EVENT	Event	OperatorLog	[BLAIR-LAPTOP2] [demo] Changed ValvePosSp_1 "Valve postion setpo" from 75 to 50	demo
14:58:54	May 9	EVENT	Event	Security	[BLAIR-LAPTOP2:Water System Demo] demo has logged on	
14:53:18	May 9	EVENT	Event	Security	[BLAIR-LAPTOP2] demo has logged on	
14:52:55	May 9	EVENT	Event	Security	[BLAIR-LAPTOP2:BLAIR-LAPTOP2:Water System Demo] demo has logged on	
14:52:09	May 9	EVENT	Event	OperatorLog	[BLAIR-LAPTOP2] Application Started (0 min 10 sec)	Logged Off
13:41:58	May 9	EVENT	Event	Security	[BLAIR-LAPTOP2] demo has logged off	
12:54:45	May 9	EVENT	Event	OperatorLog	[BLAIR-LAPTOP2] [demo] Changed AutoSysControl "Storage variable fo" from 0 to 1	demo

Figure1: Security and operator actions recorded in an encrypted log file

In addition, the SCADA package may identify the individual files that have changed as well as the specific sections of those files. This method of incremental change recording is preferred as it reduces the overall size of the version control repository by recording only the changed portions of the application.

The version control repository should be both secure and encrypted. These features ensure the version control system will only accept properly applied configuration changes from authenticated users. Conversely, configuration changes introduced using alternative methods should be ignored.

These features mitigate malicious activity but provide no protection against improper functionality introduced unintentionally by authenticated users. Because such changes are deployed using the proper techniques, they may enter the system legitimately. To address such situations, the version control system should support single-step and two-step deployment methodologies. In a single-step process, one user can both make configuration changes and deploy them. In a two-step process, one user may be authorized to change configuration but may not be authorized to deploy these changes, while another user may be authorized to deploy changes but not to configure them. This two-step approach ensures two users review all changes. Since the change management process is fully integrated, unintentional mistakes must pass two sets of eyes and are more likely to be found before affecting the production process.



Workstations

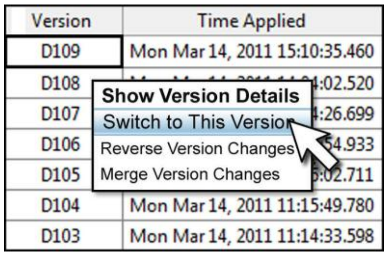
Workstation	Last Update	Current Version
BLAIR-LAPTOP2	Thu May 10, 2012, 13:19:09.303	BLAIR-LAPTOP2-D302
CHRISTOPHERL	Wed Apr 04, 2012, 14:47:16.173	BLAIR-LAPTOP2-D272
VTSDEMO	Wed Jan 18, 2012, 08:47:27.156	VTSDEMO-D6

Version Log for BLAIR-LAPTOP2 (Total of 404 records in log)

Version	Time Applied	User	Comment
BLAIR-LAPTOP2-D302	Thu May 10, 2012 13:19:09.303	admin	Automated commit.
BLAIR-LAPTOP2-D301	Wed May 09, 2012 15:21:56.931	demo	Automated commit.
BLAIR-LAPTOP2-D300	Wed May 09, 2012 12:48:57.162	demo	Modifying a Page Note.
BLAIR-LAPTOP2-D299	Wed May 09, 2012 12:40:47.438	demo	Automated commit.
BLAIR-LAPTOP2-D298	Tue May 08, 2012 15:57:50.972	admin	Automated commit.
BLAIR-LAPTOP2-D297	Tue May 08, 2012 15:57:24.491	admin	Tag modified: MPE1\Out of Sequence Mode
BLAIR-LAPTOP2-D296	Tue May 08, 2012 15:56:54.740	admin	Tag modified: MPE1
BLAIR-LAPTOP2-D295	Tue May 08, 2012 15:24:38.464	admin	Automated commit.
BLAIR-LAPTOP2-D294	Tue May 08, 2012 13:26:17.607	admin	Modifying a Page Note.
BLAIR-LAPTOP2-D293	Tue May 08, 2012 10:53:11.027	admin	Automated commit.
BLAIR-LAPTOP2-D292	Tue May 08, 2012 10:52:40.832	admin	Modifying a Page Note.
BLAIR-LAPTOP2-D291	Tue May 08, 2012 10:52:37.535	admin	Modifying a Page Note.

Figure2: VTS Application Version Control repository summary

Regardless of whether a single-step or two-step process is used, improper configuration still may happen and there must be a way to recover from these situations. Version control systems should offer two methods; reversing specific changes and switching to a previous application version. Reversing changes allows an authorized user to select one or more changes and selectively choose which of the changes to reverse. This is best applied when specific configuration changes have been identified as having caused an issue, while other subsequent changes have been verified as correct. Alternatively, switching to a previous version, also known as "rolling back to a known good version," switches the entire application to the exact configuration of a selected version. This is most useful when the application must quickly be restored to a functional state. Neither of the methods above should result in the deletion of any configuration history.



Version	Time Applied
D109	Mon Mar 14, 2011 15:10:35.460
D108	:02.520
D107	:26.699
D106	4.933
D105	:02.711
D104	Mon Mar 14, 2011 11:15:49.780
D103	Mon Mar 14, 2011 11:14:33.598

Figure2: VTS Application Version Control options

Version Control in the Multi-Developer Collaborative Environment

Multi-developer environments present a more challenging problem. Some version control systems require users to lock files when checking them out of the repository, then check them back in again, releasing the lock; a process that inherently disallows multiple-developer collaboration. Since large SCADA systems may require many users to make configuration changes simultaneously, SCADA version control systems should utilize version merging. This process allows many users to change the same set of files and merge their changes. Given the greater likelihood of conflicts in this type of an environment, the version control system should be able to automatically identify and resolve such conflict based on priority rules.

Combine this functionality with the two-step deployment methodology defined earlier and you have strict change management with two-step validation in a multi-developer environment. In such a situation, many developers can make changes to an application while one or more selected users validate and deploy these changes.

Version Control in a Redundant Client/Server Architecture

Redundancy has become ubiquitous in SCADA systems driven by requirements for continuous service and fault tolerance. Redundancy is generally accomplished through the use of distributed architecture, whereby each system function can be carried out by at least two separate computers in the network. Similarly, a SCADA version control system should use a distributed paradigm, such that a complete copy of the version control repository exists in more than one location. As changes are deployed, they are automatically synchronized to each of the repositories. This provides protection from catastrophic failure and prevents loss of any configuration history.

Summary

SCADA software systems are assets similar to other physical assets in the system. Unlike other assets, however, its reliance on common technology (e.g. operating systems, 3rd party software interfaces, computer components) and various constantly evolving organizational and external drivers makes it subject to continuous change. Simultaneously, the need for high-speed problem resolution and fault tolerance has made tracking the SCADA software asset challenging but essential. The three primary elements in tracking this asset are properly configured security, emergency backup and comprehensive change documentation. Industries that do not have a regulated procedural methodology for record keeping should select a system with these three elements available in a tightly integrated package, providing a solution that can greatly reduce procedural burdens while delivering a high-level of reliability.

List of Acronyms:

SCADA	Supervisory Control and Data Acquisition
GIS	Geographic Information System
CMMS............	Computerized Maintenance Management System
GAIT...............	Group for Advanced Information Technology
IT....................	Information Technology

Main Author Name*: Blair Sooley* *is an Account Manager/Pre-Sales Engineer with Trihedral. He holds an Bachelor of Electrical Engineering from Dalhousie University in Halifax, an MBA from St. Mary's University and has sat on the board of the Consulting Engineers of Nova Scotia. Blair has worked in the automation and controls industry for 17 years and has led projects in the United States, Canada, and Southeast Asia. He holds SCADA seminars and webinars annually across North America.*

Contact: bsooley@trihedral.com