Practical Guidleines for Automated Disaster Recovery and Change Management Solution with Factory Talk AssetCentre 4.0.

Author: Peter Tiagunov

Creating integration model prototype for Factory Talk Asset Centre 4.0 with disaster recovery option.

Objective: 1. Construction and test of an abstract automation environment with controlled change(s).
All environment's changes fell into two categories; expected and unexpected.
Expected – all changes related to; new development; existing process change or update, applied trough adopted change management mechanism that requires approval and schedule steps. Further, the prove that described method requires less effort on maintaining and implementing changes, will be provided.
Unexpected – all other changes that may occur as a result of emergency maintenance, temporary change(s)/workaround(s), unauthorized modification(s).

2. Enablement of a cohesive environment for development and maintenance work performed by more than 1 resource in parallel mode. In other words; environment for coordinated effort(s) with limited opportunity for conflict(s) and conflict(s) resolution mechanism.

3. Reduce number of file copies related to assets configuration and/or reduce number of de-synchronized copies of the same files.

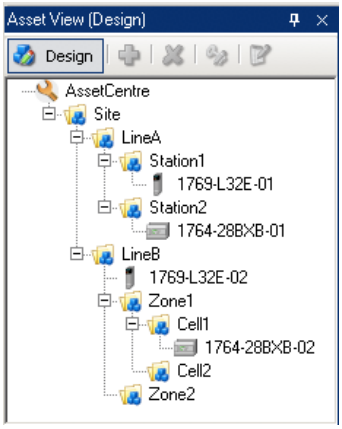
Reason: Persistent demand of knowing the state of automation code at any given time. Minimize opportunity for lost work or de-synchronized work effort(s) cases. Effective isolation of unexpected(temporary) changes to maintain convergence strategy.
Compose reliable and dynamic information of volume, frequency and type of changes occurring in automation environment.

Implementation: Using RS FTAC 4.0

1. Naming convention (not included in this document)
2. Automation Assets and infrastructure assessment.
3. Actors (Key participants)
4. Use case scenarios.

Requirements:
Each configurable and supported by (FTAC) asset (further the asset) has to have mechanisms implemented to enable the following:
a. Detect the change in asset configuration event.
b. As a result of the latter (a), create/add new version to asset's configuration history.
c. Notify responsible resource(s) about the event.
d. Identify released(controlled) version

Step 1. Creating Device's Logical Hierarchy. Logical hierarchy describes physical devices with their primary function(s) are organized with respect to logical entities, example(Abstract):

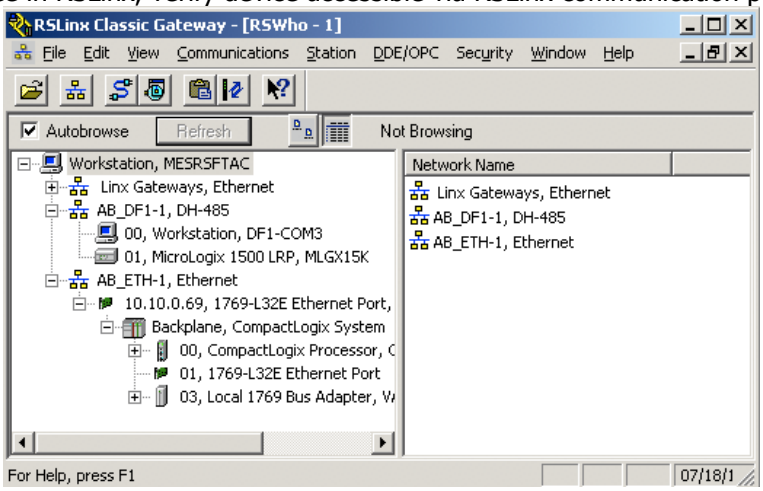


Note! While in LineA, Station1 and Station2 both have designated controllers,
LineB has controller associated with the line itself, along with another controller designated to Cell1.
Provide detailed information for each controller, include location, IP and MAC (if possible) addresses, and Serial Number.
This information is one of the critical artifacts that guarantees reliable management of the system in future.

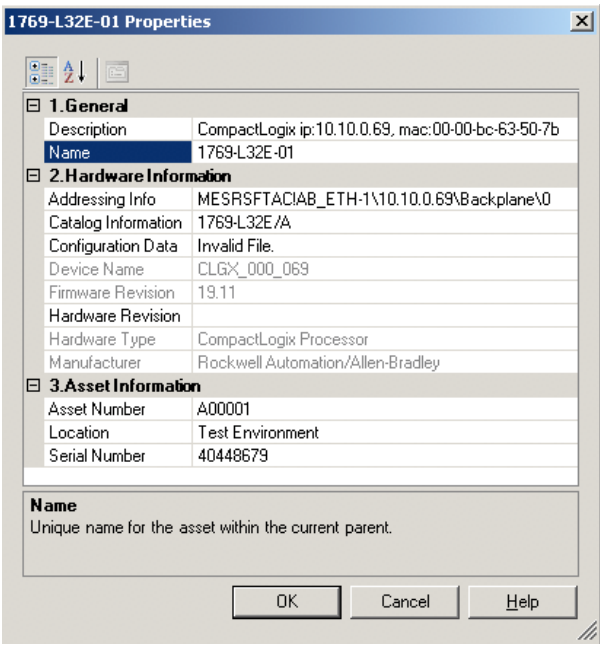
It is suggested to consider schedule factor in organization as well. In other words, in addition to functional domain, consider group assets that can be handled around the same time, using more generic level, for instance line or site.
Note! Schedule factor must be of the lower priority compare to functional factor.

Step 2. Acquire current configuration (program) files running in assets, these files are to be used for initial load to the asset centre database. Upload, verify and store programs for instance in C:\Initial Files\ folder.

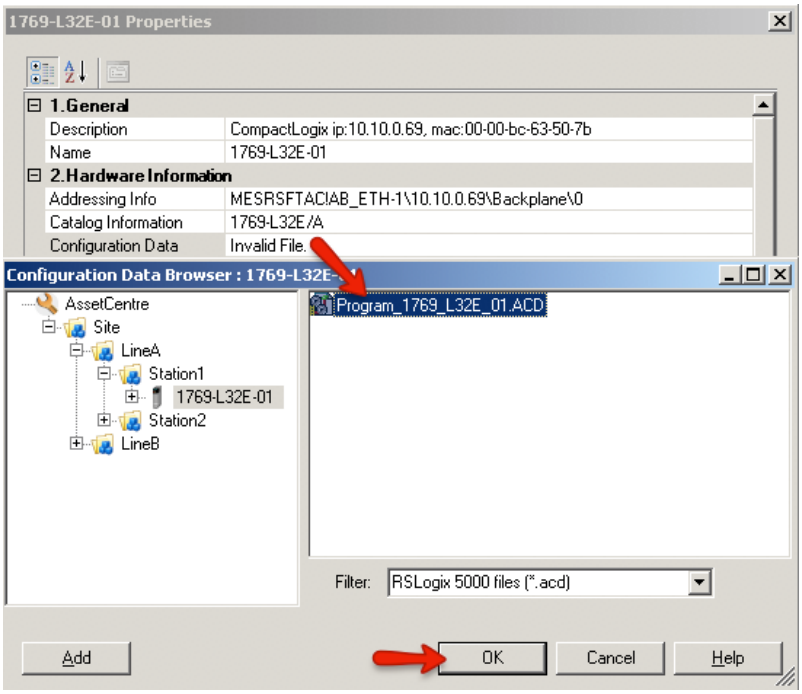
Step 3. Configure each device in RSLinx, verify device accessible via RSLinx communication path prior configuration.



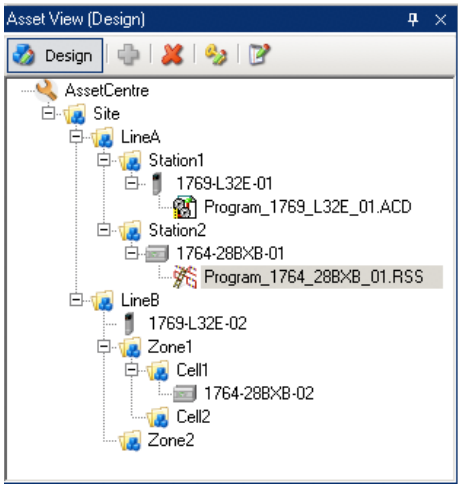
Step 4. For each controller in hierarchy, assign current corresponding configuration file. This creates the first revision of configuration (program) for each controller in central repository. Ones added the files are no longer needed, and can be deleted or archived.



Important! Verify all automation devices participated in a particular solution are part of the same broadcast domain.
Read the following article for details: http://en.wikipedia.org/wiki/Broadcast_domain



Configuration files assigned to their corresponding controllers:

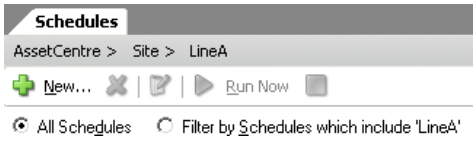


Step 5. Configure SMTP. Use FTAC Configure Server Settings Utility to setup SMTP server. Specify FQDN or ip address and port. Note! There is no authentication involved, thus SMTP server has to be on the same trusted network with the FTAC server.

Step 6. Creating schedules.

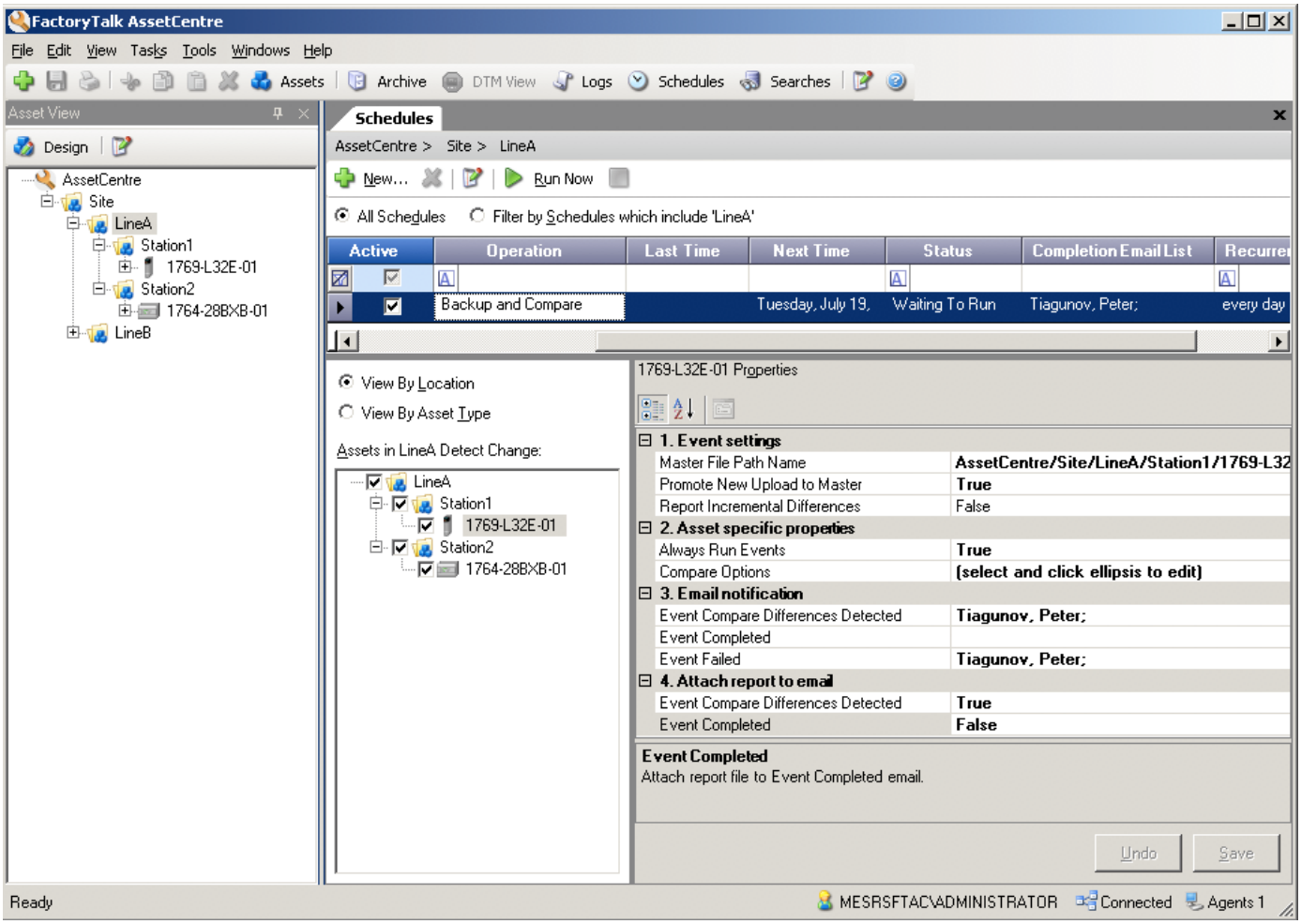
6.a. Creating a collective schedule:

Right click the LineA node and select Schedules.



Click the "+" New...

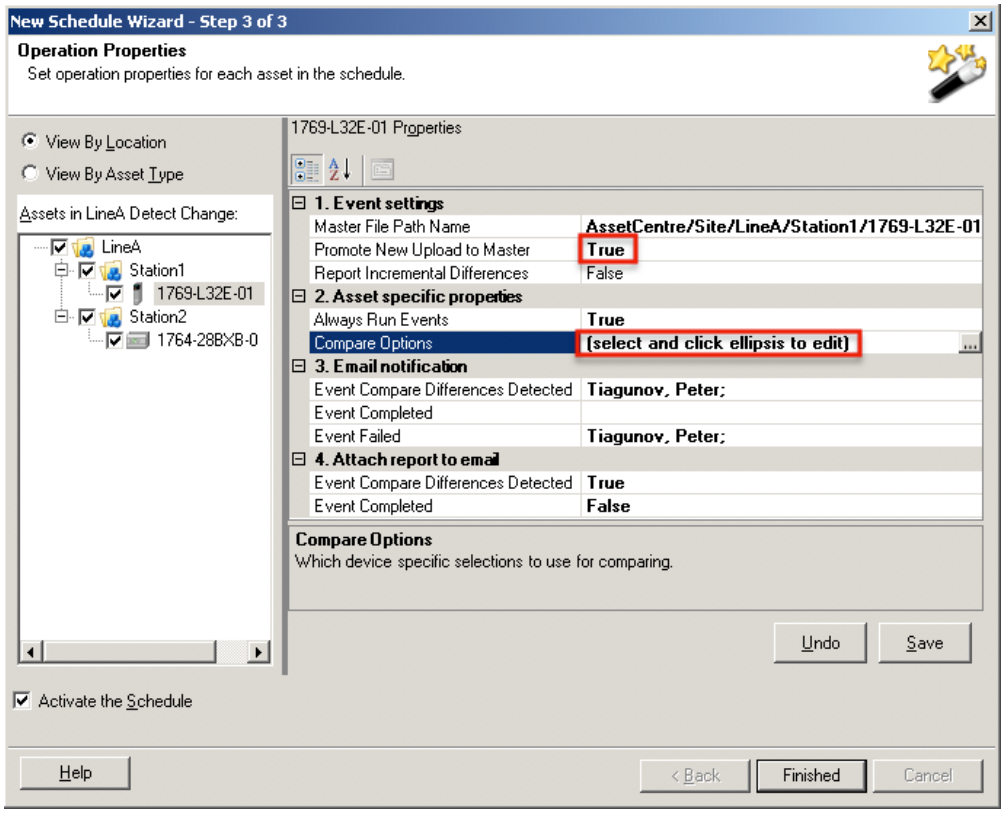
Example of a collective schedule for LineA.



6.b. Creating individual schedule: Each asset can have an individual schedule as well. Consideration. Avoid overlaps in schedules, e.g. same device in 2 schedules with potential time span overlap. Use of limited runtime is primary tool to avoid overlaps.

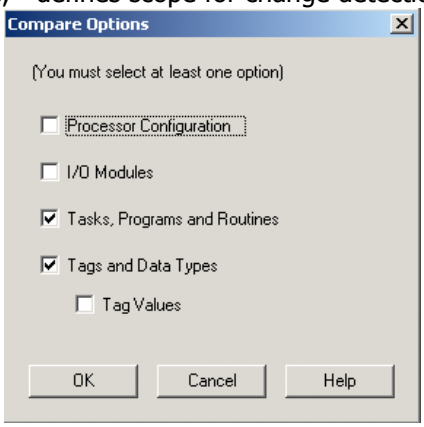
Defining schedules strategy. Each asset has to have at least 2 associated mutually exclusive schedules of 2 kinds.

Schedule1 (more frequent, example MO,TU,WE,TH,FR,SA) responsible for detection of a new configuration using current version to previous version compare mechanism. In other words, current configuration or program is uploaded and compared with the latest version in repository. If difference(s) found new, then uploaded version added(promoted) to repository. Typical period of re-occurrence for this schedule is 24-48 hours.
Primary responsibility of this kind of schedule is to detect occurred change(s) using pre-configured scope for each asset in the schedule. Once detected, change is record it in repository db in for of a new version. Change detected and recorded regardless of it's nature e.g. expected or unexpected.



Settings: Promote New Uploaded to Master =True, - on difference, will create a new version in repository using the uploaded configuration(program) as source.

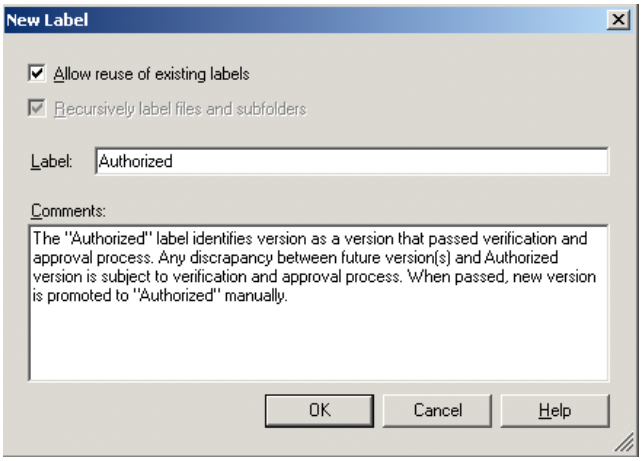
Settings: Compare Options, - defines scope for change detection, e.g.

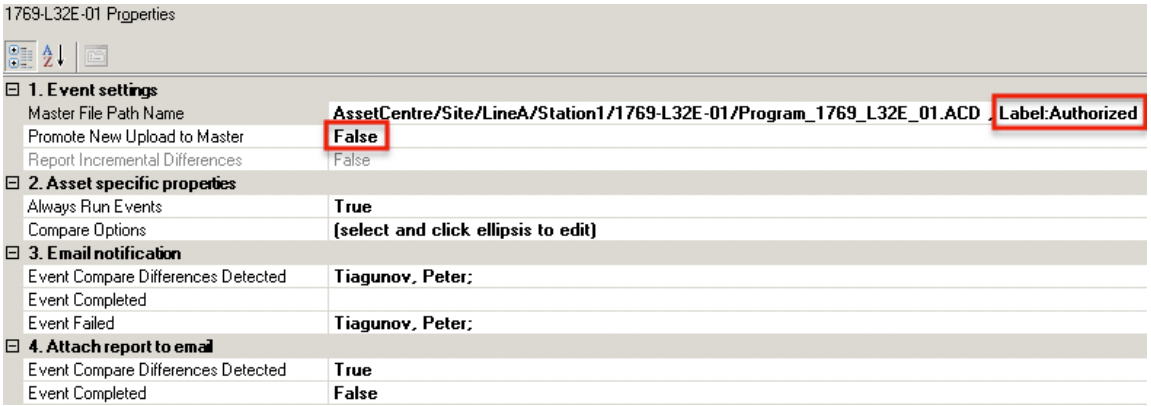


Schedule2 (less frequent example SU), responsible for comparison of uploaded(current) configuration(program) to a specific version in repository. A specific version is identified as a released or authorized version. The authorized version is the functional version without any unexpected change(s). Ideally, at any given time, all asset's has to have an authorized versions of configuration(program) l oaded. In reality it's never the case, unless no changes happening.
Primary objective of upload and comparison to authorized revision is to notify management of accumulated changes that has not been confirmed. Once acknowledgment and approval of the current version took place, it can be manually promoted to Authorized version. Otherwise, current version has to be modified to meet the requirements for authorization. Described scenario is the primary tool to manage temporary changes and changes as a result of emergency maintenance. Implementation of authorized versioning prevents control system from contamination by bypasses and workarounds.

The "Authorized" label identifies version as a version that passed verification and approval process. Any discrepancy between future version(s) and Authorized version is subject to verification and approval process. When passed, new version is promoted to "Authorized" manually.

Initial configuration: In “Archive” tab select a version identified as the “Authorized” version. Click the “New Label” menu.



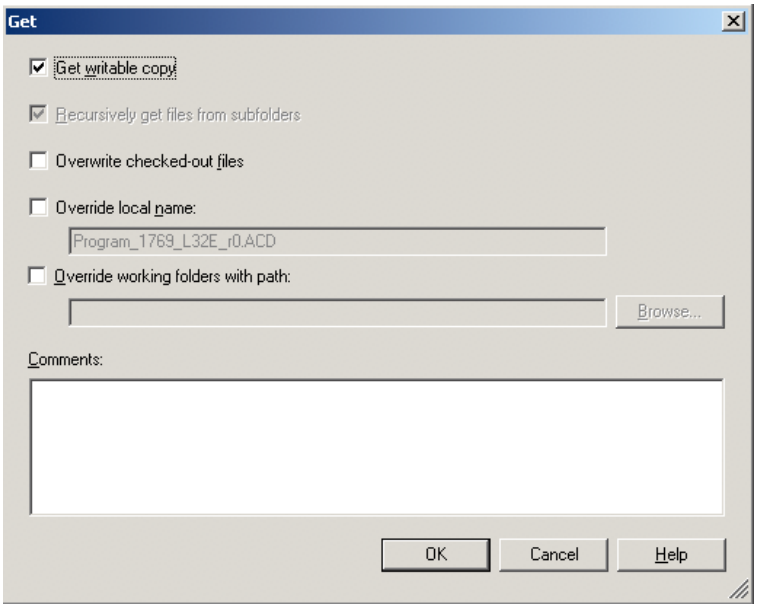


Settings: Masterfile Path Name set to “Authorized” label.
Promote New Upload to Master =false , uploaded version shall not be added as a new version to repository.

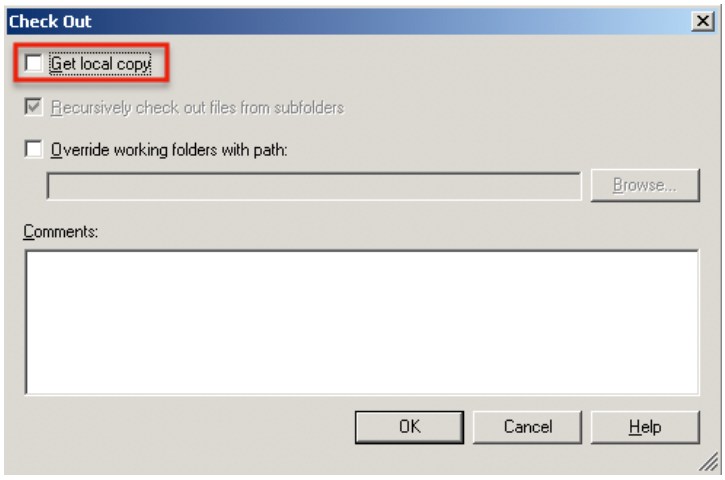
Ways of working with configuration files (programs):
So far 2 major strategy emerging:

a. Using Check-out - upload and compare will not run on checked-out file.

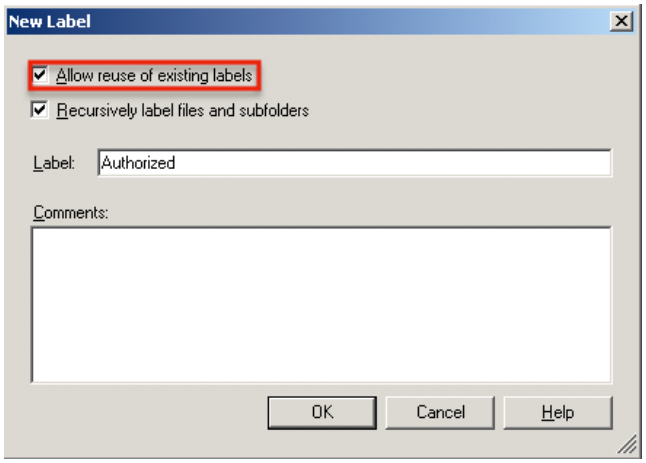
b. Using Get – suggested approach. (no pinned version allowed).
Click the asset file in Archive tab, click the “Get” button. Check “Get writable copy” option.
Click the “OK” button.



Writable copy of asset file(program) is retrieved from repository and placed in working folder.
Corresponding configuration software will open e.g. RSLogix5000.
From this point asset enters development phase.
Changes are being performed and downloaded to controller.
Each download of a changed configuration will create a discrepancy(s) that will be detected by the next Detect Change schedule. New version is created to indicated changes. This process may continue for several days.
Detect Change will continue to capture intermediate/incremental changes.
If modified asset cross the Compare Authorize schedule the discrepancies are reported but new version is not added to the repository. Once change is complete, completed version has to be checked-in to repository.
Since get was performed initially, blank Checked-out is required. In AccetCentre client application in Archive tab click or right-click the file. Uncheck the "Get local copy" box. Click the OK button.



Perform check-in , optionally use the Check in and delete the local copy feature.
New version is going to be created in repository. From this point acknowledgment and approval phase is on.
Once approval process complete new version is ready to be assigned the "Authorized" label. To assign label , in Active tab, select the latest version (the one just checked-in) in the menu click the "New Label' function.
Check the "Allow reuse of existing labels", click the OK button.



Latest version is now been approved and labeled Authorized. All future Compare Authorize schedule runs will use this version to compare with uploaded(current) version.

Notifications. E-mail notifications typically configured using the "on exception event" approach. Typical configuration enables notification e-mail for: a. on Compare Differences Detected event.
b. on Schedule fail event.

Similar approach recommended for report attachment strategy, e.g. attach report on Compare Differences Detected event.

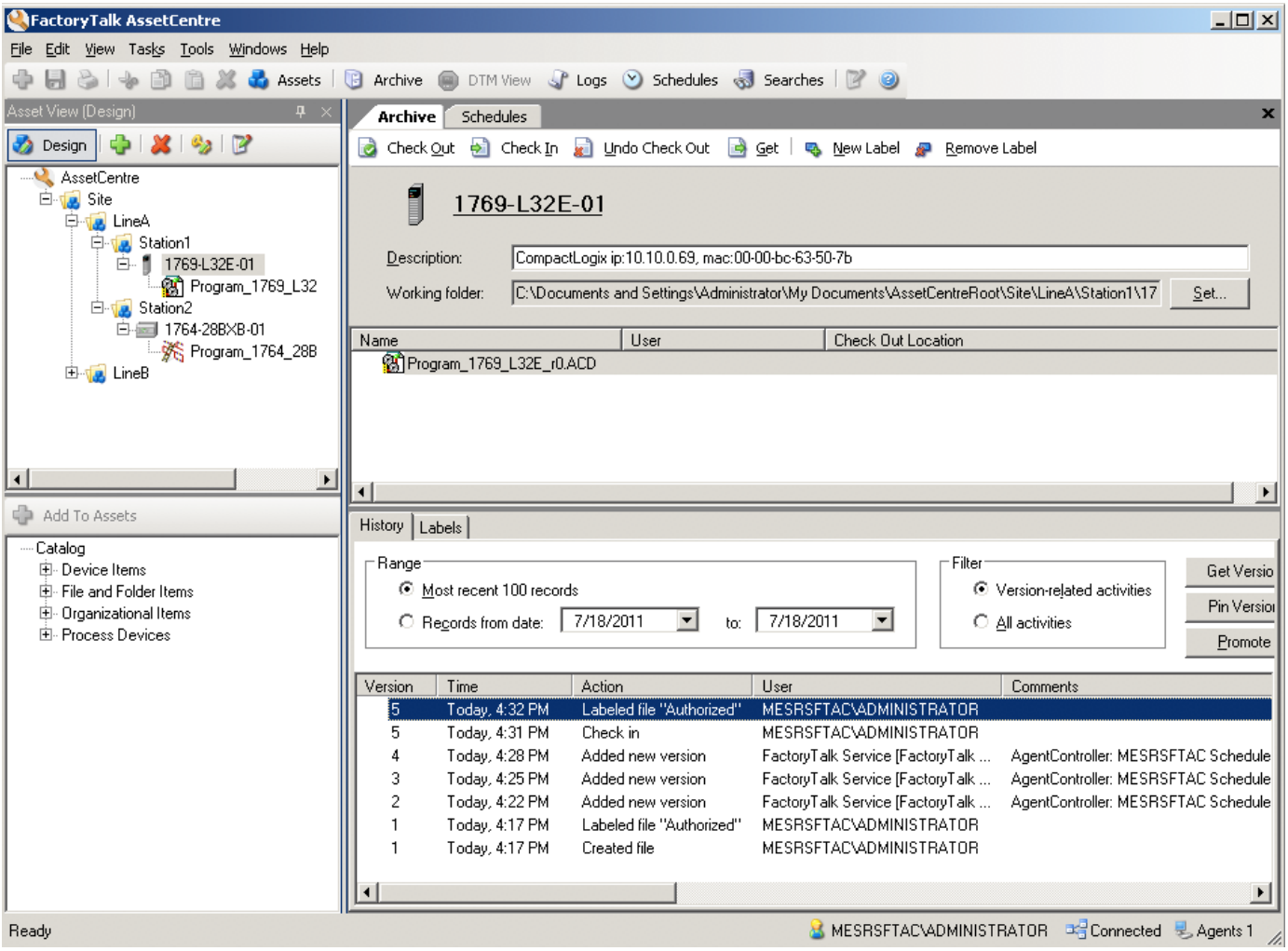
Examples of e-mail notifications and attachments provided in Appendix B.
Appendix A contains example of changes history. Note! All schedules runs are triggered manually to simulate actual environment.

Major Consideration! New description(s) are lost if new version created through upload and compare.
How to solve? Use checked-out/in for performing changes and updating adding descriptions.

Appendix A. Example of 1769-L32E-01 Asset archive history.

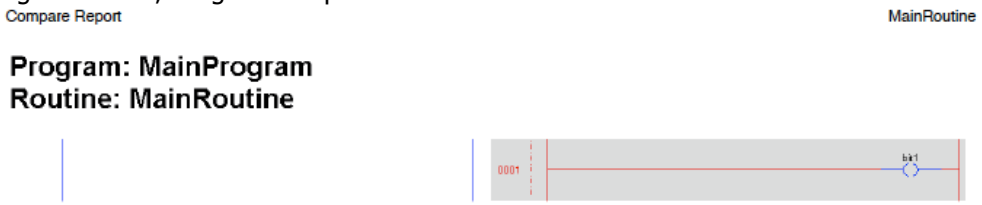
History of activity and events:

1. Asset(controller) assigned a corresponding configuration file. Ver.1.
2. Configuration file Ver.1 assigned label "Authorized"
3. Configuration file Ver.1 retrieved from repository using Get function (not shown).
4. First change created and downloaded to controller using RSLogix5000 (not shown).
5. LineA Detect Change schedule run (manually), difference(s) found, Ver.2 added to repository.
6. E-mail notification sent (See Appendix B).
7. Second change created and downloaded to controller using RSLogix500 (not shown).
8. LineA Detect Change schedule run (manually), difference(s) found, Ver.3 added to repository.
9. E-mail notification sent (See Appendix B).
10. Third change created and downloaded to controller using RSLogix500 (not shown).
11. LineA Detect Change schedule run (manually), difference(s) found, Ver.4 added to repository.
12. E-mail notification sent (See Appendix B).
13. Working copy checked-in using blank Check-out method. Ver.5 added to repository.
14. LineA Compare Authorize schedule run (manually), all 3 changes found (not shown).
15. E-mail notification sent (See Appendix B).
16. Ver.5 assigned label "Authorized".



Appendix B. Example of e-mail notifications reports.

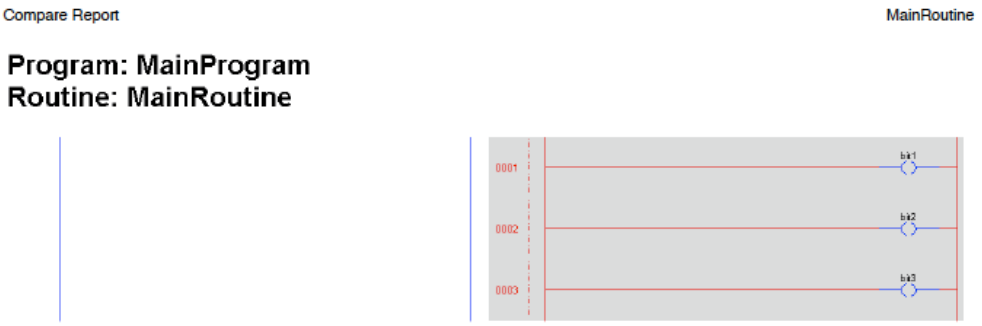
a. First change detected, rung 001 output bit1 added.



b. Second change detected, rung 002 output bit2 added.

c. Third change detected, rung 003 output bit3 added.

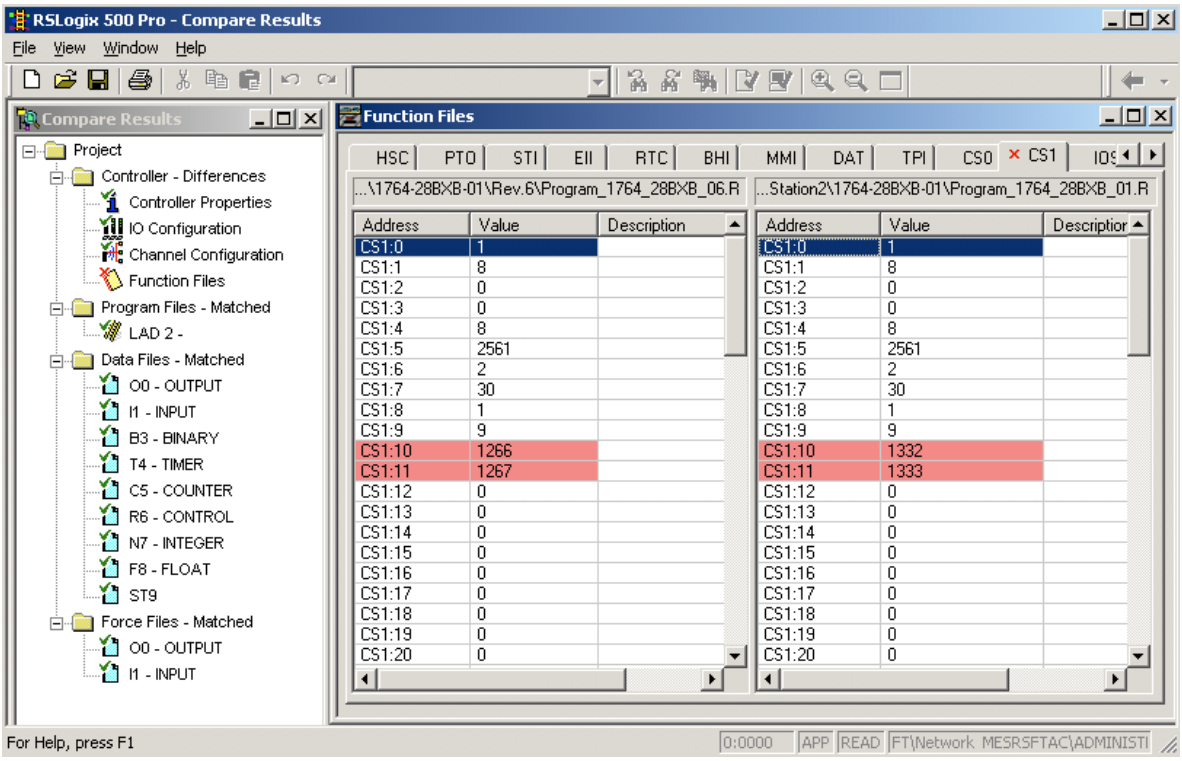
d. All 3 changes reported by the Compare Authorized schedule.



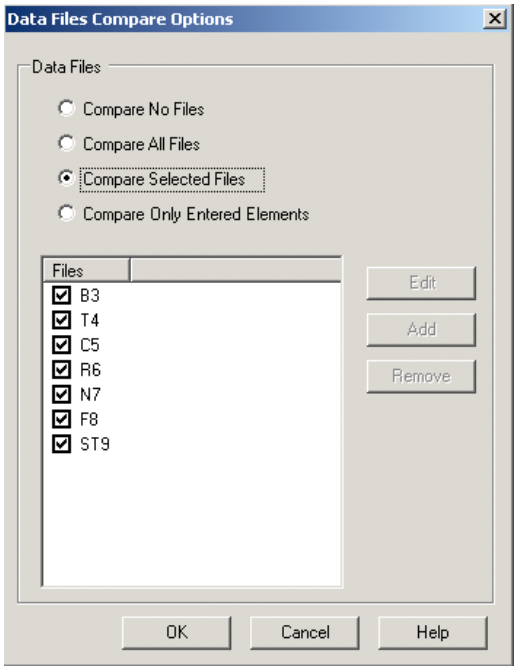
Appendix C. Known Issues:

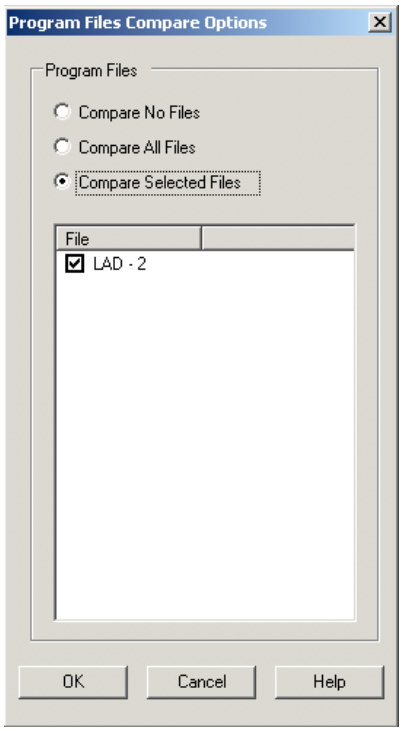
1. The same schedule allowing upload and compare for checked-out file for Logix platform, while it's not permitted on checked-out files for Micrologix platform (Waiting For Retry, The master file is checked out by user...). Inconsistency. Different outcome results for 2 assets being in the same state.

2. Upload and compare for Micrologix1500 always creates a new version due to a difference.



Compare settings in FTAC 4.0





3. Labeling mechanism has glitches.
On label change the history indicates version 4 as "Authorized".
Label "Authorized" is removed from version 1, but
Label view still showing version 1 labeled as Authorized. Fig.2
When Client reloaded it updates the label view with correct version Fig.3.

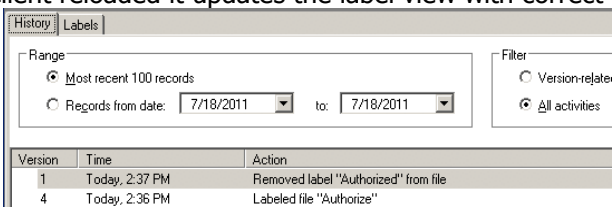


fig.1

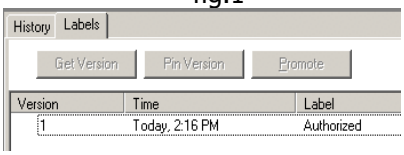


fig.2

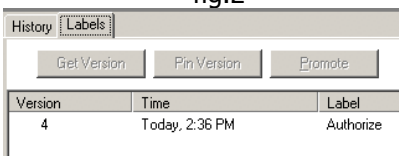


fig.3

4. Unable to explicitly assign label to a version, always assigns to the latest version.
5. Not confirmed, deleted version interfering with new versions when reinitialized