CSi.Testing

CSi.Testing

Exported on 06/24/2018

Table of Contents

Welcome!

This is the home page for your knowledge base space within Confluence. You can use your knowledge base to capture best practices and address common problems.

Next, you might want to:

- □ **Create your first knowledge base article** - Click "Create" and select a "How-to" or "Troubleshooting" article.
- □ **Customise the home page** - Click "Edit" to get started. Include useful information like what users can expect to find in this knowledge base, frequently asked questions and links to other resources.

For all projects related to testing CSI products.

1 Frequently asked questions

- Add links to popular how-to and troubleshooting articles.
- Highlight important documentation.

2 Need more help?

- Link to resources such as your service desk, questions & answers or a forum.
- List contacts for getting additional help.

3 Other resources

- Include links to relevant external sites and services.
- Add other helpful information.

4 Browse by topic

Label list

As you and your team label content this area will fill up and display the latest updates.

5 Recently updated articles

- ExamplesReleaser.Config.xml (see page 12)
  a minute ago • updated by Mark P Thomas[1] • view change[2]
- Examples Releaser (see page 11)
  a minute ago • updated by Mark P Thomas[3] • view change[4]
- ResultsArchiver (see page 14)
  12 minutes ago • updated by Mark P Thomas[5] • view change[6]
- ResultsArchiver.Config.xml (see page 15)
  49 minutes ago • updated by Mark P Thomas[7] • view change[8]
- Testing Quickstart (see page 21)
  about an hour ago • updated by Mark P Thomas[9] • view change[10]
- Source Control Testing Directory Structure (see page 17)
  about 2 hours ago • updated by Mark P Thomas[11] • view change[12]
- Running Tests (see page 17)
  about 2 hours ago • updated by Mark P Thomas[13] • view change[14]
- CSiTester (see page 10)
  about 2 hours ago • updated by Mark P Thomas[15] • view change[16]
- CSi.Testing (see page 0)
  Jun 22, 2018 • created by Mark P Thomas[17]

[1] http://localhost:8090/display/~MarkPThomas
[2] http://localhost:8090/pages/diffpagesbyversion.action?pageId=2424870&selectedPageVersions=3&selectedPageVersions=4
[3] http://localhost:8090/display/~MarkPThomas
[4] http://localhost:8090/pages/diffpagesbyversion.action?pageId=2424863&selectedPageVersions=5&selectedPageVersions=6
[5] http://localhost:8090/display/~MarkPThomas
[6] http://localhost:8090/pages/diffpagesbyversion.action?pageId=2424861&selectedPageVersions=4&selectedPageVersions=5
[7] http://localhost:8090/display/~MarkPThomas
[8] http://localhost:8090/pages/diffpagesbyversion.action?pageId=2424868&selectedPageVersions=3&selectedPageVersions=4
[9] http://localhost:8090/display/~MarkPThomas
[10] http://localhost:8090/pages/diffpagesbyversion.action?pageId=2424892&selectedPageVersions=11&selectedPageVersions=12
[11] http://localhost:8090/display/~MarkPThomas
[12] http://localhost:8090/pages/diffpagesbyversion.action?pageId=2424876&selectedPageVersions=7&selectedPageVersions=8
[13] http://localhost:8090/display/~MarkPThomas
[14] http://localhost:8090/pages/diffpagesbyversion.action?pageId=2424865&selectedPageVersions=2&selectedPageVersions=3
[15] http://localhost:8090/display/~MarkPThomas
[16] http://localhost:8090/pages/diffpagesbyversion.action?pageId=2424859&selectedPageVersions=1&selectedPageVersions=2
[17] http://localhost:8090/display/~MarkPThomas

6 CSiTester

On this page:

7 Examples Releaser

On this page:

7.1 Purpose

The purpose of the ExamplesReleaser is to automate the process of creating a collection of model files and any supporting files needed that are to be shipped with a given program. This is done by checking all model control *.xml files of the appropriate program and copying it if the file indicates if it is appropriate for release.

7.1.1 Release Criteria

The criteria for this are:

1. Model is of the specified program
2. Model is complete: //model/@status = done
3. Model is public: //model/@is_public = yes
4. Model is not turned off: //model/keywords does not contain /keyword = "exclude from regression testing"

7.2 Distribution

The ExamplesReleaser can be distributed both as a *.dll library for use in other programs, as well as in a console *.exe program for command line use.

7.3 Command Line Operation

The console program includes a default *.xml configuration file, described at ExamplesReleaser.Config.xml (see page 12). Values can be changed, added, and removed in order to control the program.

Config files of other names or at other locations may be used as alternative means of controlling the program. These can be specified through the command line.

For the console program to work correctly, it must be located at the following path:

- ~~.\TFS\Models Database_Tests~~

7.3.1 Overview

ExamplesReleaser.exe can be run with a number of command line parameters using the following format: **(TODO: FINISH THIS! BELOW IS NOT UPDATED)**

```
ExamplesReleaser.exe "file name or path"
```

The command line parameters are listed below.

7.3.1.1 File Name

Syntax:

```
ExamplesReleaser.exe "file name"
```

Causes the program to read the specified configuration file, assumed to be located at the location of the program.

7.3.1.2 File Path

Syntax:

```
ExamplesReleaser.exe "path\file name"
```

Causes the program to read the specified configuration file at the absolute path given.

7.4 ExamplesReleaser.Config.xml

This page is used to document the schema for the XML config file used for ExamplesReleaser settings.

On this page:

7.4.1 Description of individual XML elements

XML Element	Multiple?	Optional?	Description
/**examples_release**			Root element.
/examples_release/ **application**			Application name.
/examples_release/ **path_source**		Y	The path to a root directory to check for examples to release. Choosing a more specific root will reduce processing time. Default is: **TODO**

XML Element	Multiple?	Optional?	Description
/examples_release/ **path_release**		Y	Path to copy the released examples to. Default is: **TODO**

8 ResultsArchiver

On this page:

8.1 Purpose

The purpose of the ResultsArchiver is to automate the process of archiving the following files from a given test run associated with a given program of a given version and and build:

- Test results HTML file.
- All model log files

The ResultsArchiver will either override existing archived results, or only add to the archive for new models, program builds, etc.

8.2 Distribution

The ResultsArchiver can be distributed both as a *.dll library for use in other programs, as well as in a console *.exe program for command line use.

8.3 Command Line Operation

The console program includes a default *.xml configuration file, described at ResultsArchiver.Config.xml (see page 15). Values can be changed, added, and removed in order to control the program.

Config files of other names or at other locations may be used as alternative means of controlling the program. These can be specified through the command line.

For the console program to work correctly, it must be located at the following path:

- .\TFS\Models Database_Tests

8.3.1 Overview

ResultsArchiver.exe can be run with a number of command line parameters using the following format:

```
ResultsArchiver.exe "file name or path"
```

The command line parameters are listed below.

8.3.1.1 File Name

Syntax:

```
ResultsArchiver.exe "file name"
```

Causes the program to read the specified configuration file, assumed to be located at the location of the program.

8.3.1.2 File Path

Syntax:

```
ResultsArchiver.exe "path\file name"
```

Causes the program to read the specified configuration file at the absolute path given.

8.4 ResultsArchiver.Config.xml

This page is used to document the schema for the XML config file used for ResultsArchiver settings.

On this page:

- Description of individual XML elements (see page 15)

8.4.1 Description of individual XML elements

XML Element	Multiple?	Optional?	Description
/**test_results**			Root element.
/test_results/**@runType**			Type of run corresponding with the models testing directories. These are: • single (all models run once, under a single analysis setting) • psb (all models are run under each of the 9 analysis setting combinations) • development (model files under development that are not checked in to source control)
/test_results/**@overWriteExisting**			True: Existing results will be overwritten by the current ones generated for archiving. False: Only results not yet archived will be added.

XML Element	Multiple?	Optional?	Description
/test_results/**application**			Application name.
/test_results/**version**			Tested version, in the format of vv.v.v.bbbb where bbbb is the build number. This must match what is specified in the regTest.xml used for the desired test run results.
/test_results/**suite_names**			List of suite names to archive. If none is specified, then all suites will be archived.
/test_results/suite_names/**suite_name**	Y	Y	Name of the suite to archive.
/test_results/**test_descriptions**			Test descriptions to archive. If none is specified, then all default test descriptions will be checked. This must match what is specified in the regTest.xml used for the desired test run results.
/test_results/test_descriptions/**test_description**	Y	Y	Test description according to the regTest system. In most cases this is 'run as is'. For PSB tests this will be the PSB setting (e.g. P1 S1 B1). If archiving results from all PSB settings of a PSB run, make sure that no test description is specified.

9 Running Tests

The child pages describe various details related to the running, reviewing, and archiving of test results.

On this page:

9.1 Source Control Testing Directory Structure

The following is the structure of the testing directories stored through source control. Directories that are not version controlled are created by running the deployment batch files after getting the latest files from source control.

- Names within curly braces { } are generic and there may be more than once occurrence.
- Names within brackets [] are optional.
- Names starting with _ tend to be auxiliary directories or contain supporting files that are deployed after pulling from source control.

On this page:

9.1.1 Models Database Directories

The following is the structure of all directories stored under 'Models Database' in source control. These files all relate to the testing of CSi programs.

Path	Files Under Version Control?	Description
.\Models Database	Yes	Root node in source control.
.\Models Database_Tests	Yes	**Contains all files and directories relevant to running and reviewing tests in an organized way.**
.\Models Database_Tests\Archive	No	Contains log files and html summary of test runs.
.\Models Database_Tests\Results	No	Contains complete results for any regTest runs.
.\Models Database_Tests\Testing	Yes	Contains batch files for running all suites of a given program (or several), in parallel or series.

.\Models Database\Examples Development	**No**	**Example models under development, which are not yet checked in to source control.**
.\Models Database\Examples Development**{Program Name}**	No	Contains example models under development, organized by program.
.\Models Database\{Program Name}	**Yes**	**All model files checked in to source control that are used for running tests.**
.\Models Database\{Program Name}**{Suite Name}**	**Yes**	Contains all model files used for testing, organized by program and suite.

9.1.2 _Tests Directories

The following is the structure of all directories stored under 'Models Database_Tests', which contains all files and directories relevant to running, reviewing, and archiving tests in an organized way.

Path	**Files Under Version Control?**	**Description**
.\Models Database_Tests	**Yes**	**Contains all files and directories relevant to running, reviewing, and archiving tests in an organized way.**
.\Models Database_Tests\Archive	**No**	**Contains log files and html summary of test runs.**
.\Models Database_Tests\Archive\**{Program Name} v{version} Build {build}**	No	Contains results organized by program name, version, and build.
.\Models Database_Tests\Archive\{Program Name} v{version} Build {build}**{bit}-bit**	No	Contains results organized by 32-bit or 64-bit runs.
.\Models Database_Tests\Archive\{Program Name} v{version} Build {build}\{bit}-bit**{Suite Name}**	No	Contains all log files of the run, and the html run summary results for 32-bit runs of the suite using the program of a particular version and build.
.\Models Database_Tests\Results	**No**	**Contains complete results for any regTest runs.**
.\Models Database_Tests\Results\Examples Development	No	

.\Models Database_Tests\Results\Examples Development**{Program Name}**	No	Single run suite results of examples under development, organized by program name.
.\Models Database_Tests\Results\ **{Program Name}**	No	
.\Models Database_Tests\Results\ {Program Name}**{Suite Name}**	No	
.\Models Database_Tests\Results\ {Program Name}\{Suite Name}**{Single}**	No	Single run results of the complete suite.
.\Models Database_Tests\Results\ {Program Name}\{Suite Name}**{PSB}**	No	PSB run results of the suite.
.\Models Database_Tests\Testing	**Yes**	**Contains builds, XML files, and batch files for running all suites of a given program (or several), in parallel or series.**

9.1.3 _Tests\Testing Directories

The following is the structure of all directories stored under 'Models Database_Tests\Testing', which contains all program and regTest builds, all instruction XML files, and all batch files for running all suites of a given (or several) program(s), in parallel or series.

Path	**Files Under Version Control?**	**Description**
.\Models Database_Tests	**Yes**	**Contains all files and directories relevant to running and reviewing tests in an organized way.**
.\Models Database_Tests\Testing	**Yes**	**Contains builds, XML files, and batch files for running all suites of a given program (or several), in parallel or series.**
.\Models Database_Tests\Testing_builds	**No**	**Contains all builds to be used for testing.**
.\Models Database_Tests\Testing_builds\ **{Program Name}**	No	Contains all builds to be used for testing for a given program. Also contains auxiliary files for testing builds, such as level, licence, and native imaging files for a given program.

.\Models Database_Tests\Testing_builds\{Program Name}**\{Program Name} Build**	No	Generic build to be use for testing. Clear this directory and copy all files from the desired build into this directory.
.\Models Database_Tests\Testing_builds\{Program Name}**\{Program Name} v{version} [Build {build}] {bit}-bit**	No	Specific build to be used for testing. These are copied down from where the builds are stored on the network.
.\Models Database_Tests\Testing_regTest	**Yes**	**regTest files to use for testing.**
.\Models Database_Tests\Testing_regTest**\previous_test_results**	**Yes**	
.\Models Database_Tests\Testing_regTest\previous_test_results**\{Program Name}**	**Yes**	regTest previous test results, organized by program, to use for testing. All builds of regTest deployed will be updated with these results.
.\Models Database_Tests\Testing_regTest**\regTest**	**Yes**	regTest program to use for testing. Clear this directory and copy all files from the desired build into this directory.
.\Models Database_Tests\Testing_src	**Yes**	**Contains source files for deployment.**
.\Models Database_Tests\Testing_src**_localBatchFiles**	**Yes**	Batch files used to run regTest that are copied in deployment to the local setup. These are re-used among the different setups.
.\Models Database_Tests\Testing_src**\Examples Development**	**Yes**	Source files unique to 'Example Development' runs.
.\Models Database_Tests\Testing_src\Examples Development**\regTest-Development**	**Yes**	Destination directory for local regTest copy and all XML files used for development model test runs.
.\Models Database_Tests\Testing_src**\PSB**	**Yes**	Source files unique to 'PSB' runs.
.\Models Database_Tests\Testing_src\PSB**_src**	**Yes**	Batch files used to run regTest for each PSB setting that are copied in deployment to the local setup. These are re-used among the different PSB setups for each suite.
.\Models Database_Tests\Testing_src\PSB**\{Suite Name}**	**Yes**	

.\Models Database_Tests\Testing_src\PSB\{Suite Name}**_regTestXml**	**Yes**	All XML files used for PSB runs of the corresponding suite.
.\Models Database_Tests\Testing_src**Single**	**Yes**	Source files unique to 'Single' runs.
.\Models Database_Tests\Testing_src\Single\regTest-{Suite Name}	**Yes**	Destination directory for local regTest copy and all XML files used for suite model test runs.
.\Models Database_Tests\Testing\Examples Development	**No**	**Directory for examples under development. This is created by deployment scripts.**
.\Models Database_Tests\Testing\Examples Development**regTest-Development**	No	Contains run batch file, regTest & local instructions for examples under development.
.\Models Database_Tests\Testing\PSB	**No**	**Directory for examples run under the 9 PSB analysis cases. This is created by deployment scripts.**
.\Models Database_Tests\Testing\PSB\**{Suite Name}**	No	
.\Models Database_Tests\Testing\PSB\{Suite Name}**regTest-P{n}S{n}B{n}[- Standard]**	No	Contains run batch file, regTest & local instructions for running all 9 analysis settings variations in parallel or series. The only irregular directory naming is regTest-P2S2B1 - Standard. All others are named as regTest-P{n}S{n}B{n}.
.\Models Database_Tests\Testing\Single	**No**	**Directory for examples run under the single default analysis case. This is created by deployment scripts.**
.\Models Database_Tests\Testing\Single\regTest-{Suite Name}	No	Contains run batch file, regTest & local instructions for running all suites in parallel or series.

9.2 Testing Quickstart

This page describes the testing setup organization and procedures in a brief and simple way to quickly run tests of the most common and generic type.

On this page:

For testing, everything you will work with or view is within:

```
.\TFS\Models Database\_Tests
```

You can see in general what the directory structures contain within Source Control Testing Directory Structure (see page 17).

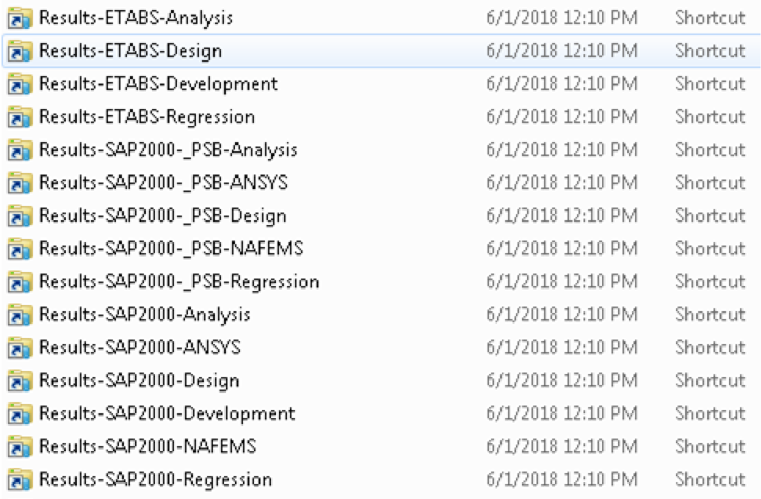
9.2.1 Testing

This directory is where you will place updated builds and initialize test runs.

9.2.2 Results

This directory contains all results of any test.

Within 'Testing' there are handy shortcuts that jump you to the correct sub-directory within this directory for viewing results of a test you have just run.



Results-ETABS-Analysis	6/1/2018 12:10 PM	Shortcut
Results-ETABS-Design	6/1/2018 12:10 PM	Shortcut
Results-ETABS-Development	6/1/2018 12:10 PM	Shortcut
Results-ETABS-Regression	6/1/2018 12:10 PM	Shortcut
Results-SAP2000-_PSB-Analysis	6/1/2018 12:10 PM	Shortcut
Results-SAP2000-_PSB-ANSYS	6/1/2018 12:10 PM	Shortcut
Results-SAP2000-_PSB-Design	6/1/2018 12:10 PM	Shortcut
Results-SAP2000-_PSB-NAFEMS	6/1/2018 12:10 PM	Shortcut
Results-SAP2000-_PSB-Regression	6/1/2018 12:10 PM	Shortcut
Results-SAP2000-Analysis	6/1/2018 12:10 PM	Shortcut
Results-SAP2000-ANSYS	6/1/2018 12:10 PM	Shortcut
Results-SAP2000-Design	6/1/2018 12:10 PM	Shortcut
Results-SAP2000-Development	6/1/2018 12:10 PM	Shortcut
Results-SAP2000-NAFEMS	6/1/2018 12:10 PM	Shortcut
Results-SAP2000-Regression	6/1/2018 12:10 PM	Shortcut

9.2.3 Archive

This likely won't be of concern in most cases, but this is where archives of test results are being added. The process is being automated to make it faster, more consistent, and more organized. In order to contribute, please see ResultsArchiver (see page 14).

9.2.4 Deploying Testing Directories

For greater organization, automation, and ease of transferring to other testing machines, there is a testing directory under source control that contains the basic files, structure, and deployment scripts. In this way, on any machine where testing is to be done, all that is necessary is the following:

1. 'Get latest' from source control for 'Models Database\' of the desired program to pull the necessary model testing files down to the local machine.
2. 'Get latest' from source control for 'Models Database_Tests' to pull the necessary testing directory files down to the local machine.
3. Run the _Deploy- script under '.\TFS\Models Database_Tests'. There are several to choose from, which are described below.

9.2.4.1 Deployment Scripts

The following are the main scripts used to deploy the testing directories. For finer control to create a more limited setup for specific testing, see the scripts contained deeper withing the directories. The following are global scripts that call these more local scripts.

1. **_Deploy-PSB-All.bat**
   a. This deploys all files and folders related to testing all programs where PSB tests are applicable. This may take some time and require a large amount of available space.
2. **_Deploy-Single-All.bat**
   a. This deploys all files and folders related to single test runs of all programs, including examples development.
3. **_Deploy-Testing-All.bat**
   a. This script runs all deployment scripts. Only run this one if you want to be able to run both single, development, and PSB tests as it may take a while and require a large amount of available space.

9.2.5 Running Tests

9.2.5.1 Generic Builds

The simplest way to test a build is to just do the following steps:

1. Copy the files of the desired build to:
   a. **.\TFS\Models Database_Tests\Testing_builds\{Program Name}\{Program Name} Build**
2. Run the following batch file to run examples:
   a. **.\TFS\Models Database_Tests\Testing\Run-{Program Name}-{Parallel | Series}-GenericBuild.bat**

Note that testing in this way is not compatible with storing results in the archive using ResultsArchiver (see page 14).

9.2.5.2 Specific Builds

With a bit more work you can test with specific builds such that results are easier to track and can be archived using ResultsArchiver (see page 14). The procedure is as follows:

1. Copy the desired build folder from the network to:
   a. **.\TFS\Models Database_Tests\Testing_builds\{Program Name}**

2. For all tests to be run, update the XML elements in the regTest-{Program Name}.xml files for the latest build location, version, and build numbers. A build find/replace action in an editor such as Notepad++ makes this more reasonable.
    a. Go to: **.\TFS\Models Database_Tests\Testing\{Single | PSB | Examples Development}\[{PSB Suite Name}\]regTest-{Suite Name}\regTest**
    b. Open **regTest-{Program Name}.xml**
    c. Update **regtest/testing/program/version** (~line 137)
    d. Update **regtest/testing/program/build** (~line 138)
    e. Update **regtest/testing/program/path** (~line 139)
3. Run the following batch file to run the examples:

1. **.\TFS\Models Database_Tests\Testing\Run-{Program Name}-{Parallel | Series}.bat**

9.2.5.3 Example with Generic Builds

Example with Generic Builds

The simplest way to test a build is to just do the following steps (for programs other than ETABS, just change the name accordingly):

1. Copy the files of the desired build to:
    a. **.\TFS\Models Database_Tests\Testing_builds\ETABS\ETABS Build**
2. Run the following batch file to run both the analysis and design verification examples concurrently:
    a. **.\TFS\Models Database_Tests\Testing\Run-ETABS-Parallel-GenericBuild.bat**

9.2.6 Viewing Test Results

To review the results is to just do the following steps:

1. Double click the following shortcuts, located at the same place as where you initiated the test run:
    a. **.\TFS\Models Database_Tests\Testing\Results-{Program Name}-{Suite Name}**
2. Or go to here:
    a. **.\TFS\Models Database_Tests\Results\{Program Name}\{Suite Name}\{Single | PSB | Examples Development}**
3. Enter the most recently created directory (date/time are in the name), and once the test run is far enough along, you can find and open the following HTML file to view the run status and results:
    a. **test_results_{YYYY-MM-DD(seconds) Program Name & Build} run_as_is.html**

9.2.6.1 Example with ETABS Analysis and Design Suites

Example with ETABS Analysis and Design Suites

The simplest way to review the results is to just do the following steps after a test run has been started:

1. Double click the following shortcuts, located at the same place as where you initiated the test run:
    a. **.\TFS\Models Database_Tests\Testing\Results-ETABS-Analysis**
    b. **.\TFS\Models Database_Tests\Testing\Results-ETABS-Design**

2. Or go to here:
    a. **.\TFS\Models Database_Tests\Results\ETABS\Analysis\Single**
    b. **.\TFS\Models Database_Tests\Results\ETABS\Design\Single**
3. Enter the most recently created directory (date/time are in the name), and once the test run is far enough along, you can find and open the following HTML file to view the run status and results:
    a. **test_results_{YYYY-MM-DD(seconds) Program Name & Build} run_as_is.html**

9.2.7 Archiving Test Results

Test results may be archived for future storage in source control or on the network. See Source Control Testing Directory Structure (see page 17) for the preferred naming convention and directory organization. This may be automated using ResultsArchiver (see page 14).