Associating Requirements with Files

This chapter describes how to associate requirements with files so that the JIRA Traceability Report can show static analysis violations and build reviews within the context of the analyzed source files.

> **Implement This Workflow for Each Build ID**
>
> The following workflow describes how to enable requirements traceability for a specific build ID. You must repeat the steps for each additional build ID you want to view requirements traceability information for.

1. Developers edit and commit source code files to source control.
2. A special CSV file that maps files to requirements is prepared, e.g.:

File	Associated Requirement ID
Project-A/src/foo/goo.java	reqA
Project-A/src/foo2/goo.c	reqA, reqB, reqC

> **Creating the File-to-Requirement Map**
>
> The file may be prepared manually be team members or it can be automated with a script. For example, the script could scan the source control repository for requirements markers added by developers and generates the CSV file accordingly.

3. Parasoft code analysis and test execution tools analyze the source code with static analysis or metrics analysis. The analysis run must be run with the specific build ID and filter ID (see the documentation for the tool for instructions on how to specify these IDs). The purpose of this run is to feed the DTP database with information about files committed to the source control system for the specific build ID (see step #1).
4. Run the CSV-scanning script shipped in the DTP_HOME\grs\extras\traceability directory with the following arguments:

```
groovy fileReqAssoc.groovy -csv <CSV_FILE_NAME> -build <BUILD_ID> -dtp <DTP_HOST_INC_PROTOCOL> -user
<DTP_USERNAME> -password <DTP_PASSWORD>
```

   This is an example script intended to demonstrate a sequence of API calls that should be performed to associate files with artifacts in DTP. You can use this example script as a starting point for implementing a more advanced solution.

   The script is written in Groovy, but the directory also contains a JAR file that you can run with the same arguments.

   The script scans the CSV file prepared in step #2 and feeds the DTP database with the information about requirements to files mapping. See the README.txt file for details how to run the script.

After the script finishes, the traceability data for the specific filter ID and build ID will be stored in the database. The data can now be viewed using one of the following methods:

- Calling the artifactTraceability REST API endpoint
- In the JIRA Traceability Report DTP Enterprise Pack extension