Using Quest Code Tester 1.8.1 with a centralized code schema

by Steven Feuerstein, Quest Software

February 2008

Overview

There are many ways to configure Quest Code Tester, all depending for the most part on how you have configured your own application environment.

This document describes the steps you must take with Quest Code Tester 1.8.1 to support the following application configuration:

- A single schema owns all the application packages. Each package is defined with AUTHID CURRENT_USER. Let's call this schema CODE_SCHEMA.
- Data is stored in other schemas, a separate schema for each different *something* (different country, different department, you name it). For the purposes of this document and example below we will work with countries.
- A user of the application connects to a schema for a given country and then executes the code owned by CODE_SCHEMA. Since the CODE_SCHEMA packages are AUTHID CURRENT_USER, all references to tables are resolved according to the privileges of the current (country) schema.

The development team wants to set up their Quest Code Tester test definitions and packages so that they are defined in the CODE_SCHEMA, but they are executed from any of the country schemas.

The rest of this document shows how to achieve this. I will provide all the code in this document to demonstrate the technique; I will also reference other files. You can use this code and the files to reproduce my steps, or you can simply apply them to your own environment.

Note: the current production version of Quest Code Tester is 1.8. Unfortunately, a change in 1.8 (specifically, to leave AUTHID CURRENT_USER off of the test package specification), means that the above scenario cannot be supported. Instead, a change has been made for the patch release 1.8.1 to correct this problem. So you will need to obtain the 1.8.1 patch release to implement this configuration.

Relevant files

central_code_schema.sql – contains all the code shown in this document and provides instructions for the sequence of steps needed to achieve the desired effect.

q##app_code.qut – the export file for the test definition of the APP_CODE procedure

Create schemas and objects

The code below performs the following steps:

- Create three schemas: CODE_SCHEMA (owns the application code) and ERITREA and LATVIA (two country schemas that contain the application data).
- Creates in the CODE_SCHEMA the APP_CODE procedure, a "stand-in" for the real application code. This procedure deletes from the APP_CODE_TABLE. This procedure is defined as AUTHID CURRENT_USER.
- Grants execute on APP_CODE to the country schemas.
- All three schemas contain their own "version" of the APP_CODE_TABLE, each with one row of data. CODE_SCHEMA's table contains the value 0, LATVIA the value 1, and ERITREA the value 2.

```
CONNECT Sys/<your password> AS SYSDBA

DECLARE
   PROCEDURE create_user (NAME_IN IN VARCHAR2)
   IS
      user_does_not_exist   EXCEPTION;
      PRAGMA EXCEPTION_INIT (user_does_not_exist, -01918);
   BEGIN
      /* Drop user if already exists */
      BEGIN
         EXECUTE IMMEDIATE 'drop user ' || NAME_IN || ' cascade';
      EXCEPTION
         WHEN user_does_not_exist
         THEN
            NULL;
      END;

      /* Grant required privileges...*/
      EXECUTE IMMEDIATE    '
    grant Create Session, Resource to '
                        || NAME_IN
                        || ' identified by '
                        || NAME_IN;
   END;
BEGIN
   create_user ('code_schema');
   create_user ('latvia');
   create_user ('eritrea');
END;
/

CONNECT code_schema/code_schema

CREATE TABLE app_code_table (n NUMBER)
/
INSERT INTO app_code_table
     VALUES (0)
/
```

```
COMMIT
/

CREATE OR REPLACE PROCEDURE app_code (n_in IN NUMBER)
AUTHID CURRENT_USER
IS
BEGIN
   DELETE FROM app_code_table
         WHERE n = n_in;
END;
/

GRANT EXECUTE ON app_code TO latvia
/
GRANT EXECUTE ON app_code TO eritrea
/
CONNECT latvia/latvia

CREATE TABLE app_code_table (n NUMBER)
/
INSERT INTO app_code_table
     VALUES (1)
/
COMMIT
/
CONNECT eritrea/eritrea

CREATE TABLE app_code_table (n NUMBER)
/
INSERT INTO app_code_table
     VALUES (2)
/
COMMIT
/
```

Import the test definition

Next, import the test definition I have already created into Code Tester, while connected to the CODE_SCHEMA. In other words, the CODE_SCHEMA owns the test definition.

The file q##app_code.qut contains the export of the test definition.

You can perform the import through the Code Test UI, or simply execute the q##app_code.qut file from Toad or another editor.

Make sure country schemas can run tests

You will now need to explicitly grant EXECUTE authority on the test package to each of your country schemas.

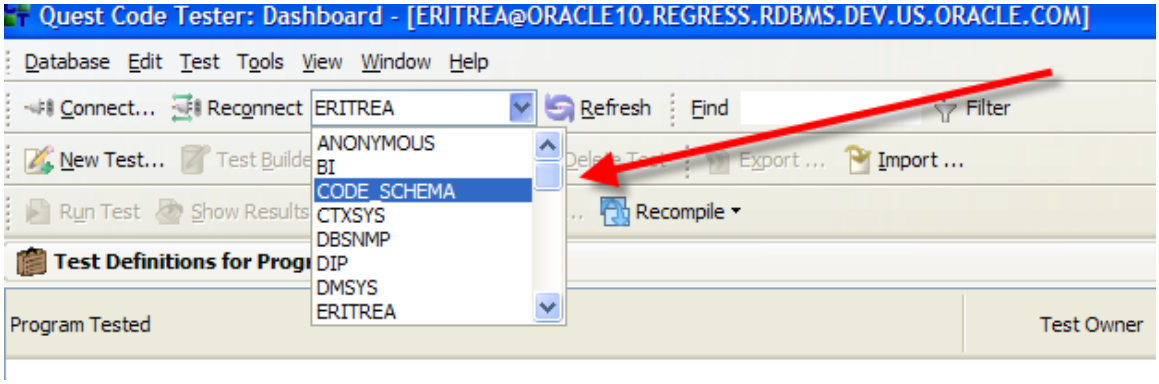
This step is *not* performed automatically by Quest Code Tester, and it will be quite impossible to run the test package from the country schemas without it. That's simply the way that Oracle works.

```
CONNECT code_schema/code_schema

GRANT EXECUTE ON q##app_code TO eritrea
/
GRANT EXECUTE ON q##app_code TO latvia
/
```

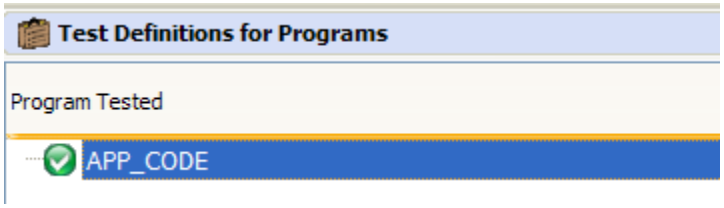
Demonstrate problem with 1.8

Before showing you how to get things to work very nicely in 1.8.1, I will demonstrate the problem in 1.8.

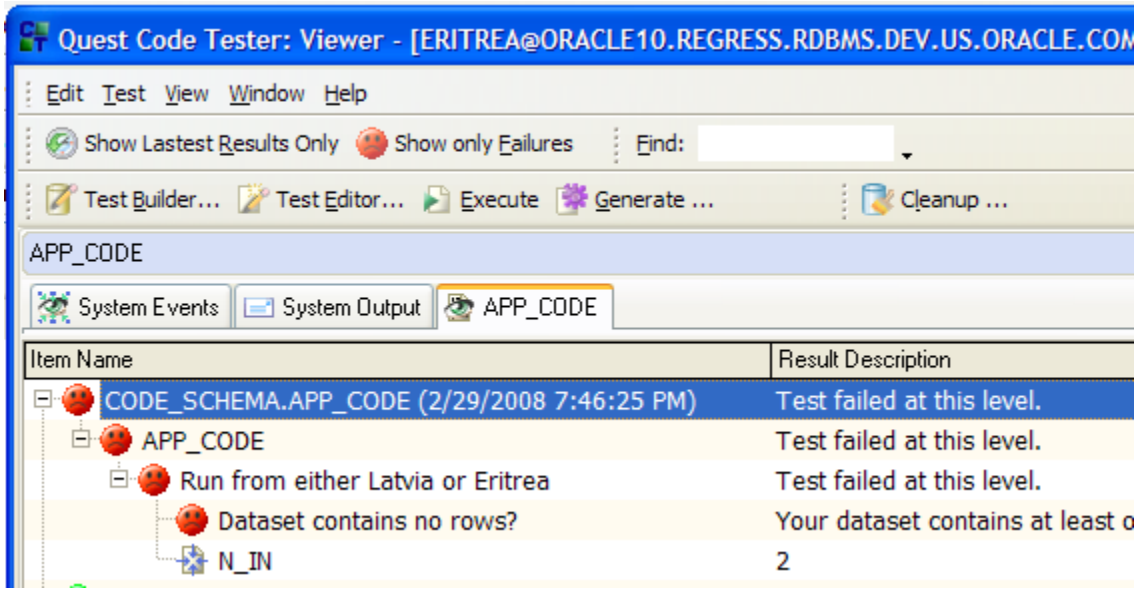
Start Code Tester and connect to the ERITREA schema. Change the current schema to CODE_SCHEMA:



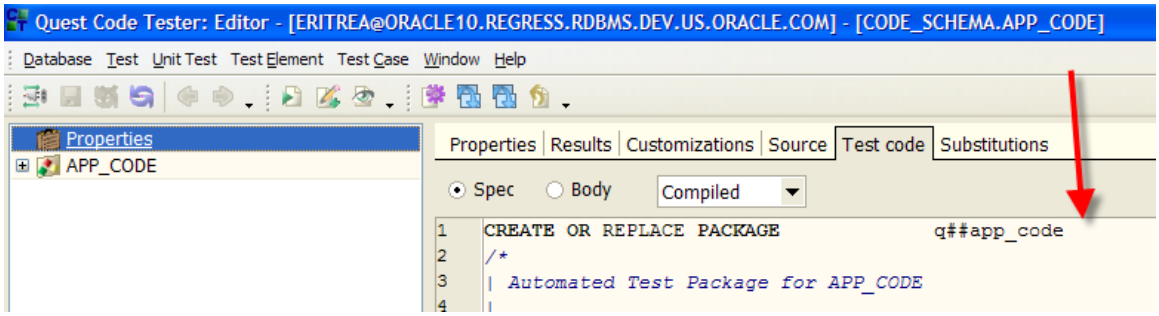
The APP_CODE program name should then appear in the list:



Now run the test and you should see failure:



That's because by default, there is no AUTHID CURRENT_USER on the test package specification, as you can see in the test editor:



Without CURRENT_USER, when you run the test, the program is executed under CODE_SCHEMA's authority and the wrong action is taken on the underlying table.

And in 1.8, there is no way to add this clause, unless you edit the test package after generation, which is certainly something you should avoid doing.

In 1.8.1, we have added the ability for you to specify code that should be run just before the test package is generated. Specifically, at this pre-generation event, you can tell Code Tester to include the AUTHID CURRENT_USER clause to your package.

Fixing the problem with 1.8.1

Here's how it works:

At certain points in its execution, Code Tester checks for the existence of a package named *my_codetester*. If this package exists and if it contains the correct event trigger procedure, that procedure is called. In 1.8.1, it will now check for the existence of my_codetester.gen_pre_generation and if present, Code Tester will execute it.

So...inside Toad, connect to the CODE_SCHEMA schema and compile this package:

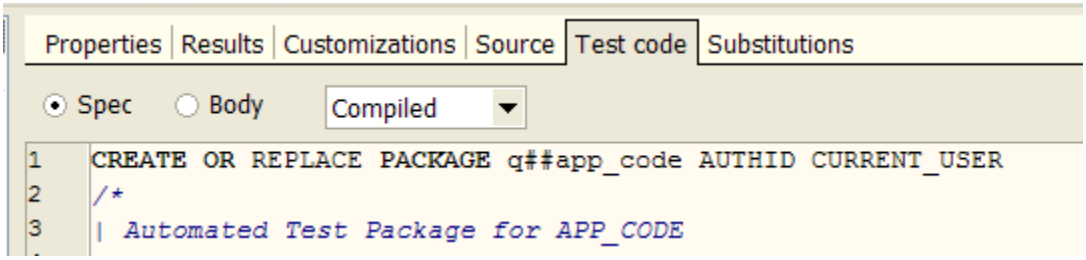
```
CREATE OR REPLACE PACKAGE my_codetester
IS
    PROCEDURE gen_pre_generation (
       schema_in IN VARCHAR2, NAME_IN IN VARCHAR2);
END my_codetester;
/

CREATE OR REPLACE PACKAGE BODY my_codetester
IS
    PROCEDURE gen_pre_generation (
        schema_in IN VARCHAR2, NAME_IN IN VARCHAR2)
    IS
    BEGIN
       qu_generate.authid_current_user (add_in => TRUE);
    END gen_pre_generation;
END my_codetester;
/
```

With that package in place, whenever you ask to generate a test package, Code Tester will first run the gen_pre_generation program, turn on inclusion of the CURRENT_USER clause, and *then* generate the code. The approach shown above enables AUTHID CURRENT_USER for all test definitions. You can be more selective by using the schema and/or program name parameters, as in:

```
CREATE OR REPLACE PACKAGE BODY my_codetester
IS
    PROCEDURE gen_pre_generation (
        schema_in IN VARCHAR2, NAME_IN IN VARCHAR2)
```

```
    IS
    BEGIN
       IF schema_in = 'CODE_SCHEMA'
       THEN
          qu_generate.authid_current_user (add_in => FALSE);
       ELSIF name_in = 'SPECIAL_PROG'
       THEN
          qu_generate.authid_current_user (add_in => FALSE);
       ELSE
          qu_generate.authid_current_user (add_in => TRUE);
       END IF;
    END gen_pre_generation;
END my_codetester;
/
```

And now, when I run my test, I see that my test package includes the AUTHID clause:



And I receive a green result: