

---

---

# Testing PL/SQL with Ounit

UCRL-PRES-215316

---

---



**December 21, 2005**  
**Computer Scientist**  
**Lawrence Livermore National Laboratory**  
**Arnold Weinstein**

# Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

## **Auspices Statement**

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

# Definition

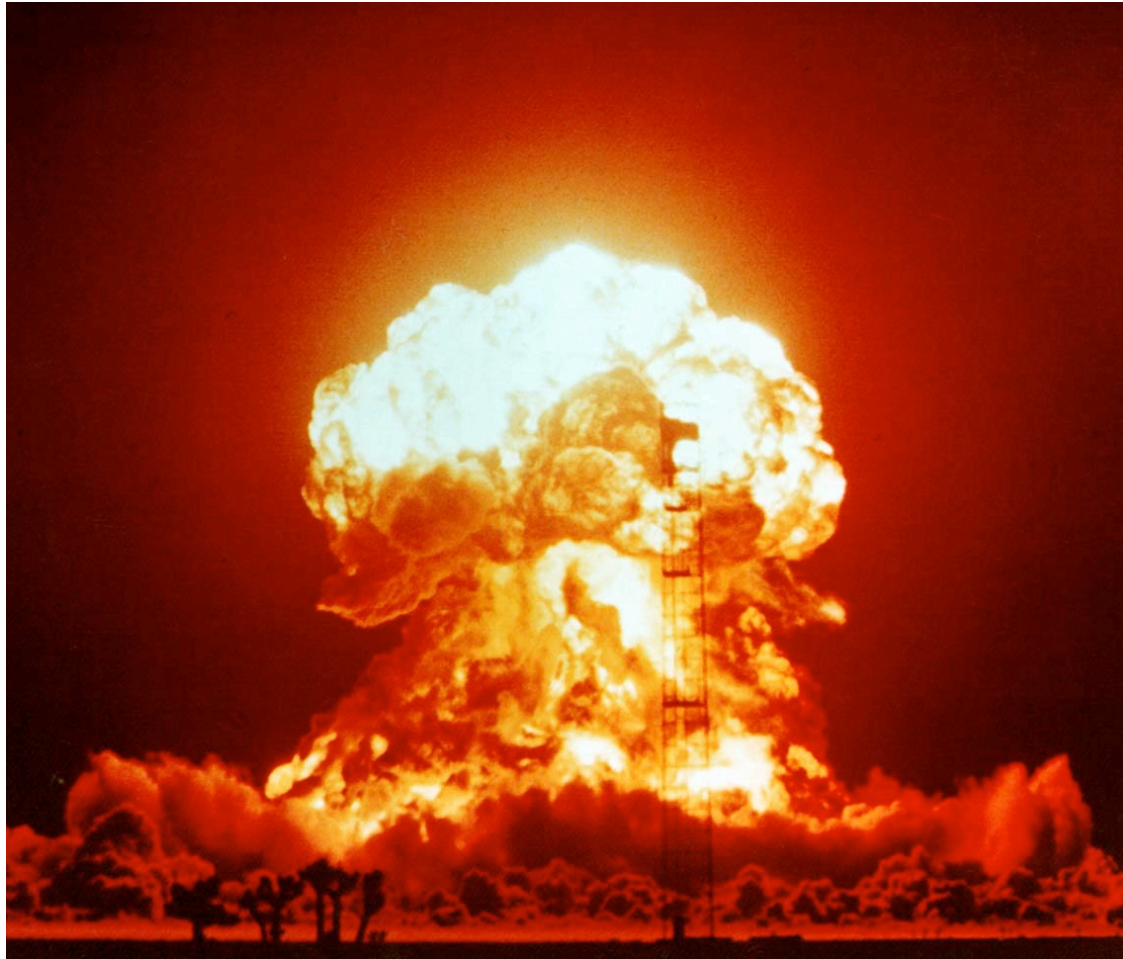
- **Software testing** is a process used to identify the correctness, completeness and quality of developed computer software.
- Actually, testing can **never** establish the correctness of computer software, as this can only be done by formal verification. It can only find defects, not prove that there are none.
- There are many approaches to software testing, but effective testing of complex products is essentially a process of **investigation**, not merely a matter of creating and following rote procedure.

# Why do we test ?



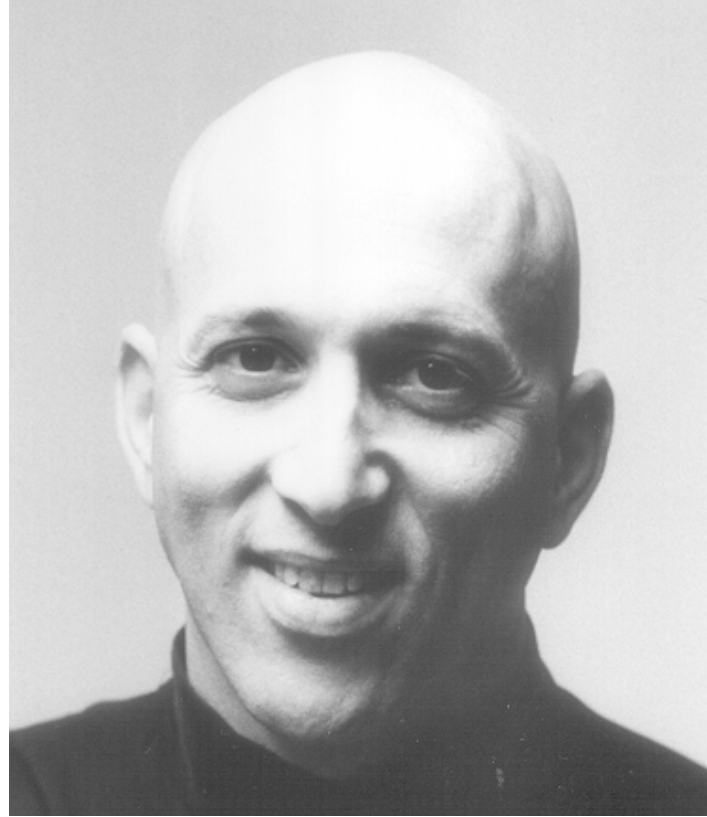
Its something we just do.

# Why do we really test !



Because failure is not an option.

# Where did Ounit come from



Steven Feuerstein

# What is Ounit ?

- Ounit is a utility that helps Oracle developers unit test their code faster, easier and more comprehensively than ever before.
- Ounit offers a powerful graphical interface to utPLSQL, the open source unit testing framework for the Oracle PL/SQL language.
  - Ounit gui only available on windows
  - utPLSQL available anywhere SQL/PLUS works
- With Ounit, you can simply point and click your way through testing sessions, and instantly see the outcomes. Because testing is easier and faster you will test more frequently and more thoroughly.
- How much does it cost? It's FREE.

# What isn't Ounit

- Ounit does not help you build your test cases and unit test procedures.
- Ounit is not intended to replace powerful interactive development environments. Instead, they will complement those tools with powerful, GUI-driven testing.



# What is utPLSQL

- utPLSQL is a unit **testing framework** for programmers using Oracle's PL/SQL language. It allows the automated testing of PL/SQL packages, functions and procedures.
- You must develop the test code to exercise your application code and return results that the utAssert command can analyze.
- How much does it cost? It's FREE.

# Testing with Ounit and utlPLSQL

- Build a test package
  - Generate a test package shell with **utGen** package procedure
  - Modify test package

SQL/PLUS script

```
SET serveroutput on size 1000000
SPOOL c:\temp\cca_to_room.sql

EXEC utgen.testpkg('cca_to_room',null,null,'UT_');

SPOOL off
```

# Generated Test Package

```
CREATE OR REPLACE PACKAGE BODY ut_cca_to_room IS
  PROCEDURE ut_setup IS
  BEGIN
    NULL;
  END;
  PROCEDURE ut_teardown IS
  BEGIN
    NULL;
  END;
  PROCEDURE ut_cca_to_room IS -- Verify and complete data types.
    against_this VARCHAR2(2000);
    check_this   VARCHAR2(2000);
  BEGIN
-- Define "control" operation
    against_this := NULL;
-- Execute test code
    check_this := cca_to_room(string_in => '');
-- Compare the two values.
    utassert.eq('Test of CCA_TO_ROOM', check_this, against_this);
  END ut_cca_to_room;
END ut_cca_to_room;
```

# Modify test package

- Modify test package
  - Add Setup and Teardown code
  - Add specific test case
    - Start with null case
    - Add case for every possible combination of inputs or as many as needed

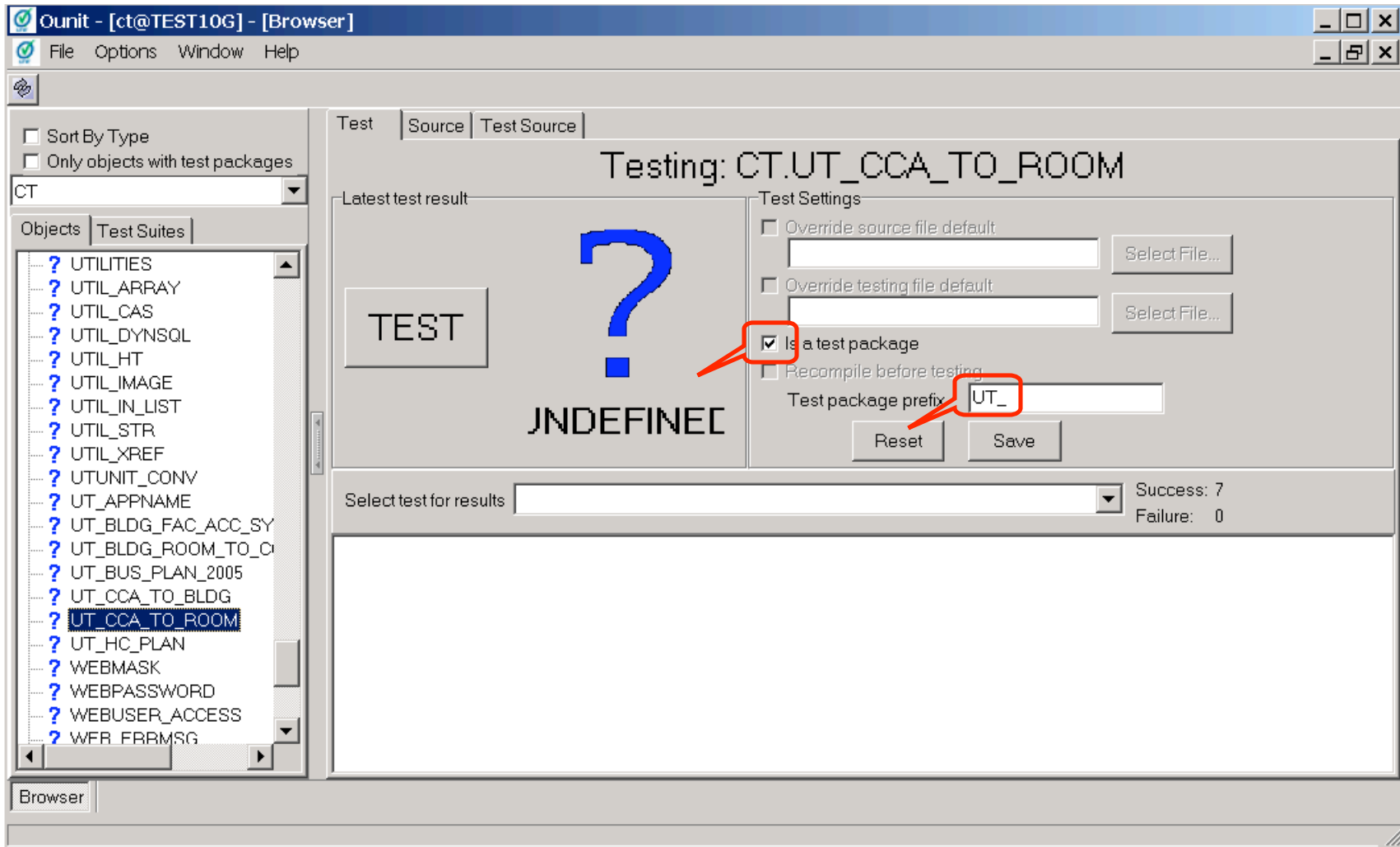
# Modified Test Package

```
PROCEDURE ut_cca_to_room IS    -- Verify and complete data types.
    against_this VARCHAR2(2000); check_this VARCHAR2(2000);
BEGIN
    -- Null test case 1.0
    against_this := NULL;
    check_this := cca_to_room(string_in => NULL);
    utassert.isnull('Test 1.0 of cca_to_room null', check_this);
    -- Normal test case 1.1
    against_this := '100';
    check_this := cca_to_room(string_in => 'B111 R100');
    utassert.eq('Test 1.1 of cca_to_room B111 R100',check_this,against_this);
    -- Leading Blank test case 1.2
    against_this := 'B100';
    check_this := cca_to_room(string_in => ' B111 RB100');
    utassert.eq('Test 1.2 of cca_to_room B111 RB100',check_this against_this);
    ...
END ut_cca_to_room;
```

# Testing with Ounit and utlPLSQL

- Build a test package
  - Generate a test package shell with **utGen** package procedure
  - Modify test package
    - Add Setup and Teardown code
    - Add specific test case
      - Start with null case
      - Add case for every possible combination of inputs or as many as needed
- Run test case with Ounit
  - Set the test package to use with program being tested
  - Run test package
  - Make adjustments to test package or program based on results of test.
  - When you make changes to program rerun test to make sure everything is still working properly.

# Ounit Test Package Coordination



# Ounit Test Package Source

The screenshot shows the Ounit software interface. On the left, there is a tree view of test packages under the 'CT' category. The 'CCA\_TO\_ROOM' package is selected and highlighted. The main window displays the source code for this package, which is a SQL function definition. The code includes a header comment with system and procedure information, a description, and a history table.

```
1 CREATE OR REPLACE
2 FUNCTION cca_to_room(string_in VARCHAR2)
3 /*
4  *****
5  -- System Name      : ChemTrack
6  --
7  -- Procedure Name:  CCA_TO_ROOM_FNC.sql
8  --
9  -- Author:         : Arnold Weinstein
10 --
11 -- Classification:  Unclassified
12 --
13 -- Description     : This function converts cca_code to room.
14 --
15 -- Processing      : Returns room value only.
16 --
17 --      Called By - na
18 --      Input     - na
19 --      OutPut    - na
20 --
21 -- History        :
22 --
23 -- Date Ver Name      Description
24 --      No.
25 -----
26 -- 30-JAN-2002 01 A. Weinstein Created
27 -- 17-FEB-2004 02 A. Weinstein Correct leading and trailing character
28 --                               problem
29 -- 08-MAR-2004 03 A. Weinstein Correct leading and trailing character
30 --                               problem
```



# Ounit Test Package Test Source

The screenshot shows the Ounit software interface. The title bar reads "Ounit - [ct@TEST10G] - [Browser]". The menu bar includes "File", "Options", "Window", and "Help". On the left, there is a tree view under "CT" with "Objects" and "Test Suites" tabs. The "Objects" tab is active, showing a list of test packages. "CCA\_TO\_ROOM" is selected and highlighted in blue. The main editor area shows the source code for the selected package. The code is as follows:

```
1 CREATE OR REPLACE
2 PACKAGE ut_cca_to_room IS
3 /* *****
4 -- System Name : ChemTrack
5 -- Procedure Name: ut_cca_to_room.pks
6 -- Author: : Arnold Weinstein
7 --
8 -- Classification: Unclassified
9 --
10 -- Description : This procedure test the cca_to_room function
11 --
12 -- Called By - na
13 -- Input - na
14 -- OutPut - na
15 --
16 -- History :
17 -- Date Ver Name Description
18 -- No.
19 -----
20 -- 14-JUN-2005 01 A. Weinstein Created version for CT 2.0 schema
21 -----
22 */
23 PROCEDURE ut_setup;
24
25 PROCEDURE ut_teardown;
26
27 -- For each program to test...
```

# Ounit Test Package Results

The screenshot displays the Ounit testing application window. The title bar reads "Ounit - [ct@TEST10G] - [Browser]". The menu bar includes "File", "Options", "Window", and "Help".

On the left side, there is a tree view under "CT" with "Objects" and "Test Suites" tabs. The "Objects" tab is active, showing a list of objects. "CCA\_TO\_ROOM" is highlighted in blue. Other objects include CCA\_TO\_BLDG, CG\$ERRORS, CHEMEXPORT, CHEM\_ORD\_OPS\_T, CHEM\_ORD\_REVIEW, CLEAN\_UP\_PROJ, CNT\_PC\_CCA, COLLECTOR, CONS\_ADJ, CONVERT\_DIR, CORRECT\_DATE, CORRECT\_MATERIAL\_L, CORRECT\_TH\_DATE, COUNTSQLHITS, CPN\_REVIEW\_SCRIPTS, CRE\_REJECTED, CRE\_RJ\_SYN, CRE\_RJ\_SYN\_OLD, CRE\_TH, and CT\_INSRT\_UPDT.

The main area is titled "Testing: CT.CCA\_TO\_ROOM". It features a "Latest test result" section with a large green checkmark and the word "SUCCESS" in bold. To the right of this is the "Test Settings" section, which includes checkboxes for "Override source file default", "Override testing file default", "Is a test package", and "Recompile before testing". There are also input fields for "Test package prefix" (containing "UT\_") and buttons for "Reset" and "Save".

Below the test settings, there is a "Select test for results" dropdown menu showing "6/28/2005 8:47:07 AM" with a green checkmark. To the right of this menu, it displays "Success: 7" and "Failure: 0".

The bottom section of the main area contains a list of test results, each preceded by a green checkmark:

- CCA\_TO\_ROOM.UT\_CCA\_TO\_ROOM: EQ "Test 1.6 of cca\_to\_room T123456789 R100" Expected "100456789" and got "1004567"
- CCA\_TO\_ROOM.UT\_CCA\_TO\_ROOM: EQ "Test 1.5 of cca\_to\_room T1111 R100" Expected "100" and got "100"
- CCA\_TO\_ROOM.UT\_CCA\_TO\_ROOM: EQ "Test 1.4 of cca\_to\_room ALMOND AVE PODF" Expected "AVE PODF" and got "AVE"
- CCA\_TO\_ROOM.UT\_CCA\_TO\_ROOM: EQ "Test 1.3 of cca\_to\_room 11R.0176" Expected "0176" and got "0176"
- CCA\_TO\_ROOM.UT\_CCA\_TO\_ROOM: EQ "Test 1.2 of cca\_to\_room ^B111 RB100" Expected "B100" and got "B100"
- CCA\_TO\_ROOM.UT\_CCA\_TO\_ROOM: EQ "Test 1.1 of cca\_to\_room B111 R100" Expected "100" and got "100"
- CCA\_TO\_ROOM.UT\_CCA\_TO\_ROOM: ISNULL "Test 1.0 of cca\_to\_room null" Expected "" and got ""

The bottom of the window has a "Browser" tab.

# Test Package Results from SQL/PLUS

```
execute utPLSQL.test ('CCA_TO_ROOM', recompile_in => FALSE);
```

```
SUCCESS: "CCA_TO_ROOM"
```

```
> Individual Test Case Results:
```

```
SUCCESS - CCA_TO_ROOM.UT_CCA_TO_ROOM: ISNULL "Test 1.0 of cca_to_room null"
```

```
Expected "" and got ""
```

```
SUCCESS - CCA_TO_ROOM.UT_CCA_TO_ROOM: EQ "Test 1.1 of cca_to_room B111 R100"
```

```
Expected "100" and got "100"
```

```
SUCCESS - CCA_TO_ROOM.UT_CCA_TO_ROOM: EQ "Test 1.2 of cca_to_room ^B111 RB100"
```

```
Expected "B100" and got "B100"
```

```
...
```

```
>
```

```
> Errors recorded in utPLSQL Error Log:
```

```
> NONE FOUND
```

# Using setup and teardown

```
PROCEDURE utsetup IS
BEGIN
  -- Remove test cases
  DELETE   bldgs
    WHERE bldg IN('001TEST', '011TEST', '111TEST');
--
  DELETE   chemcontrolarea
    WHERE bldg IN('001TEST', '011TEST', '111TEST');
END;
PROCEDURE utteardown IS
BEGIN
  -- Remove test cases
  DELETE   bldgs
    WHERE bldg IN('001TEST', '011TEST', '111TEST');
--
  DELETE   chemcontrolarea
    WHERE bldg IN('001TEST', '011TEST', '111TEST');
END;
```

12/21/05

# Setup Data

```
-- Test for proper set-up of no records matching test records
--
  bldg_v := '011TEST';
OPEN bldgs_curvar FOR SELECT * FROM bldgs WHERE bldg = bldg_v;
FETCH bldgs_curvar INTO bldgs_rec;
--
OPEN bldg_facs_curvar FOR SELECT * FROM bldg_facs WHERE bldg = bldg_v;
FETCH bldg_facs_curvar INTO bldg_facs_rec;
--
...
--
utassert.isnull('Test-1.1 of bldgs field bldgs', bldgs_rec.bldg);
utassert.isnull('Test-1.2 of bldg_facs field bldgs', bldg_facs_rec.bldg);
...
--
CLOSE bldgs_curvar;
CLOSE bldg_facs_curvar;
CLOSE cca_curvar;
```

# Create some Data

```
-- Build first set of records for CCA="B11TEST YARD"  
--  
INSERT INTO chemcontrolarea (cca_code, admin_org_id, TIMESTAMP, quad)  
VALUES ('B011test YARD', 1000005, 465722822, '4');  
--  
OPEN bldgs_curvar FOR SELECT *FROM bldgs WHERE bldg = bldg_v;  
FETCH bldgs_curvar INTO bldgs_rec;  
OPEN bldg_facs_curvar FOR SELECT * FROM bldg_facs WHERE bldg = bldg_v;  
FETCH bldg_facs_curvar INTO bldg_facs_rec;  
...  
--  
utassert.this('Test-2 of BLDG_FACS_SYNC', if_true);  
utassert.eq('Test-3.1 of bldgs field bldgs', bldgs_rec.bldg, bldg_v);  
utassert.eq('Test-3.2 of bldg_facs field bldgs', bldg_facs_rec.bldg, bldg_v);  
...  
--  
CLOSE bldgs_curvar;  
CLOSE bldg_facs_curvar;  
...
```

# Ounit Results

The screenshot displays the Ounit testing interface. The main window title is "Ounit - [ct@teST10G] - [Browser]". The interface is divided into several sections:

- Left Panel:** A tree view showing a list of test suites and objects. The "Test Suites" tab is active, showing a list of suites including "CT". Under "CT", several test suites are listed with green checkmarks, indicating they passed, such as "BLDG\_FACS\_SYNC", "BLDG\_FAC\_ACC\_SYNC", "BLDG\_ROOM\_TO\_CCA", and "CCA\_TO\_BLDG".
- Top Panel:** Contains tabs for "Test", "Source", and "Test Source". The "Test" tab is selected, showing the title "Testing: CT.BLDG\_FACS\_SYNC".
- Center Panel:** Displays the "Latest test result" as "TEST SUCCESS" with a large green checkmark.
- Right Panel:** Shows "Test Settings" with options for overriding source and testing file defaults, a checkbox for "Is a test package", and a checkbox for "Recompile before testing". The "Test package prefix" is set to "UT". Buttons for "Reset" and "Save" are visible.
- Bottom Panel:** A summary bar shows "Select test for results" with a dropdown menu displaying "6/14/2005 1:35:08 PM". To the right, it reports "Success: 33" and "Failure: 0".
- Bottom Section:** A list of test results, each preceded by a green checkmark. The results include:
  - BLDG\_FACS\_SYNC.UTBLDG\_FACS\_SYNC: EQ "Test-3.7 of chemcontrolarea field quad" Expected "4" and got "4"
  - BLDG\_FACS\_SYNC.UTBLDG\_FACS\_SYNC: EQ "Test-3.6 of chemcontrolarea field dir" Expected "DIRECTORS OFFICE" and got "DIRE"
  - BLDG\_FACS\_SYNC.UTBLDG\_FACS\_SYNC: EQ "Test-3.5 of chemcontrolarea field org\_id" Expected "1000005" and got "1000005"
  - BLDG\_FACS\_SYNC.UTBLDG\_FACS\_SYNC: EQ "Test-3.4 of chemcontrolarea field room" Expected "YARD" and got "YARD"
  - BLDG\_FACS\_SYNC.UTBLDG\_FACS\_SYNC: EQ "Test-3.3 of chemcontrolarea field bldgs" Expected "011TEST" and got "011TEST"
  - BLDG\_FACS\_SYNC.UTBLDG\_FACS\_SYNC: EQ "Test-3.2 of bldg\_facs field bldgs" Expected "011TEST" and got "011TEST"
  - BLDG\_FACS\_SYNC.UTBLDG\_FACS\_SYNC: EQ "Test-3.1 of bldgs field bldgs" Expected "011TEST" and got "011TEST"
  - BLDG\_FACS\_SYNC.UTBLDG\_FACS\_SYNC: Test-2 of BLDG\_FACS\_SYNC
  - BLDG\_FACS\_SYNC.UTBLDG\_FACS\_SYNC: ISNULL "Test-1.3 of chemcontrolarea field bldgs" Expected "" and got ""
  - BLDG\_FACS\_SYNC.UTBLDG\_FACS\_SYNC: ISNULL "Test-1.2 of bldg\_facs field bldgs" Expected "" and got ""
  - BLDG\_FACS\_SYNC.UTBLDG\_FACS\_SYNC: ISNULL "Test-1.1 of bldgs field bldgs" Expected "" and got ""

# TOAD and Procedure

```
5:55 PM Last Update: 7/18/2005 1:30:27 PM
UT_HC_PLAN | UT_HC_PLAN | HC_PLAN | HC_PLAN
--
PROCEDURE Set_de_minimis(bldg_in IN VARCHAR2) IS
-- STEP 8 Set de_minimis_code after container_max_qty_inkg is set
--
    p_bldg    VARCHAR2(15) := UPPER(bldg_in);
BEGIN
--
-- Reset de-minimis code if max container size less than 1 kg
--
    UPDATE hc_fac_list
        SET de_minimis_cd = NULL
        WHERE (    letter_code NOT LIKE '%ET%'
                OR letter_code IS NULL)
        AND priority IN('1', '2')
        AND container_max_qty_inkg < 1
        AND bldg = p_bldg;
--
-- Set de-minimis code if max container size greater than 1 kg
--
    UPDATE hc_fac_list
        SET de_minimis_cd = '1kg'
        WHERE (    letter_code NOT LIKE '%ET%'
                OR letter_code IS NULL)
        AND priority IN('1', '2')
        AND container_max_qty_inkg > 1
        AND bldg = p_bldg;
--
-- Reset de-minimis code if max container size less than 10 kg
--
```



# TOAD and utPLSQL test


```
15:15 AM Last Update: 7/18/2005 2:56:24 PM
UT_HC_PLAN | UT_HC_PLAN | HC_PLAN | HC_PLAN

PROCEDURE |ut_set_de_minimis IS
BEGIN
  -- Execute test code
  hc_plan.set_de_minimis(bldg_in => cg$bldg132n);
  COMMIT;
--
cg$sql :=
  'select count(*) from ct.hc_fac_list WHERE '
  || ' bldg = '
  || ' '
  || cg$bldg132n
  || ' '
  || ' and ( letter_code NOT LIKE 'ET%' or letter_code is null) '
  || ' and priority IN( '1', '
  || ' '2' ) '
  || ' AND container_max_qty_inkg > 1 '
  || ' AND ( de_minimis_cd != '1kg' or de_minimis_cd is null) '
  || ' AND chemical_id NOT IN (select chemical_id from DE_MIN_LIST) '
  || ' ';
--
-- Assert success
utassert.this('Test 4.0 of SET_DE_MINIMIS', if_true);
utassert.eqqueryvalue('Test 4.1 of SET_DE_MINIMIS in 1kg for bldg => '
  || cg$bldg132n,
  cg$sql,
  0);
--
cg$sql :=
  'select count(*) from ct.hc fac list WHERE '
```

# Ounit Home Page

Back Forward Reload Stop <http://www.ouunit.com/aboutouunit.html>

Home Bookmarks



**unit**  
the Oracle Unit Tester

download  
**Ounit**

more about  
**utPLSQL and  
Unit testing**

request  
**Enhancements**

report  
**Bugs**

[home](#) | [contact us](#)

Ounit is the Oracle Unit Tester, a utility that helps Oracle developers unit test their code faster, easier, and more comprehensively than ever before (actually, for the first time ever).

Ounit consists of a powerful graphical interface to [utPLSQL](#), the open source unit testing framework for the [Oracle PL/SQL language](#). This framework offers a "cooperative paradigm" for testing: you follow utPLSQL processes and use the utPLSQL testing routines, and utPLSQL in turn runs your tests and records the results.

With Ounit 1.0, you can simply point and click your way through testing sessions, and instantly see the outcomes of your tests on the screen. The result is that you will test more frequently and more thoroughly.

Ounit is currently available at no charge to all PL/SQL developers and DBAs. *Note: Ounit 1.0 does not help you build your test cases and unit test procedures. Please see [utPLSQL documentation](#) for instructions for this step.*

Ounit is and will remain free to all PL/SQL developers and Oracle DBAs. Future and more sophisticated unit testing functionality (including test package generation) will be integrated into Swyg, a product that is being developed by the author of the utPLSQL framework, Steven Feuerstein. For more information, please visit [www.gnxxo.com](http://www.gnxxo.com).

copyright © 2003 PL/Solutions. All Rights Reserved.

# utPLSQL Home Page

Back Forward Reload Stop <http://utplsqli.sourceforge.net/Doc/index.html> Search Print

Home Bookmarks

## utPLSQL

[ [Home](#) | [Getting Started](#) | [Build Test Packages](#) | [Examples](#) | [User Guide](#) | [Release Notes](#) | [Document Map](#) ]

[Next Section: Getting Started >](#)

## Table of Contents

### Welcome to utPLSQL - a unit testing framework for the Oracle PL/SQL Language

#### Getting Started

This document tells you the minimum you need to know in order to get started with utPLSQL: how to install the software, build simple test packages, and run your tests.

#### Build Test Packages

utPLSQL provides with you a framework in which to run your tests. You still have to write your test code, and that code must follow some rules if utPLSQL is going to know how to run those tests.

#### Examples

There is no better way to learn how to build and run utPLSQL test packages than to work from the many examples found here.

#### User Guide

Once you are familiar with utPLSQL basics, have run some tests, and are ready to learn and use more of the many utPLSQL features, the User Guide will tell you all you need to know about the different features and programs of utPLSQL.

#### Release Notes

Well, you know what these are: a description of fixes and enhancements in the latest release!

#### Document Map

The full list of the pages in the documentation

The utPLSQL project is hosted at Sourceforge - the home page is to be found at <http://utplsqli.sourceforge.net>. From here, there are links to the various resources available and details on how to get involved in the project. Discussion of utPLSQL takes place at the utPLSQL Forum, so to ask further questions, or for help with problems visit <http://utplsqli.oracledeveloper.nl>.

# Summary

- Testing is good, but its time consuming and difficult.
- Ounit and utPLSQL are very useful tools for testing PL/SQL.
  - They make testing PL/SQL easier and faster
  - They are easy to install and configure
  - They formalize and store test procedures so they can be rerun as changes are made to the code.
- The testing however is only as good as you make it. But with this frame work your **testing will improve.**