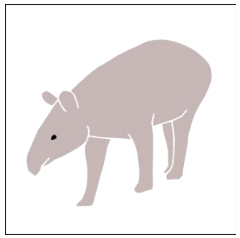

db2unit

v2_beta - Cheat Sheet

https://github.com/angoca/db2unit/wiki/API

Assertions

General

Signature	Description
`REGISTER_MESSAGE (MSG VARCHAR(256))`	Registers a message for the test that will be used if an exception is raised.
`FAIL ()` `FAIL (MSG VARCHAR(256))`	Fails the test.

Booleans

Signature	Description
`ASSERT_BOOLEAN_EQUALS (EXPD VARCHAR(32672), ACTL VARCHAR(32672))` `ASSERT_BOOLEAN_EQUALS (MSG VARCHAR(256), EXPD VARCHAR(32672), ACTL VARCHAR(32672))`	Compares two booleans.
`ASSERT_BOOLEAN_TRUE (CONDITION BOOLEAN)` `ASSERT_BOOLEAN_TRUE (MSG VARCHAR(256), CONDITION BOOLEAN)`	Checks if the given BOOLEAN has a value of TRUE.
`ASSERT_BOOLEAN_FALSE (CONDITION BOOLEAN)` `ASSERT_BOOLEAN_FALSE (MSG VARCHAR(256), CONDITION BOOLEAN)`	Checks if the given BOOLEAN has a value of FALSE.
`ASSERT_BOOLEAN_NULL (CONDITION BOOLEAN)` `ASSERT_BOOLEAN_NULL (MSG VARCHAR(256), CONDITION BOOLEAN)`	Checks if the given boolean is null.
`ASSERT_BOOLEAN_NOT_NULL (CONDITION BOOLEAN)` `ASSERT_BOOLEAN_NOT_NULL (MSG VARCHAR(256), CONDITION BOOLEAN)`	Checks if the given boolean is not null.

Datetime

Signature	Description
`ASSERT_TIMESTAMP_EQUALS (EXPD TIMESTAMP, ACTL TIMESTAMP)` `ASSERT_TIMESTAMP_EQUALS (MSG VARCHAR(256), EXPD TIMESTAMP, ACTL TIMESTAMP)`	Compares two timestamps.
`ASSERT_TIMESTAMP_NULL (VALUE TIMESTAMP)` `ASSERT_TIMESTAMP_NULL (MSG VARCHAR(256), VALUE TIMESTAMP)`	Checks if the given timestamp is null.
`ASSERT_TIMESTAMP_NOT_NULL (VALUE TIMESTAMP)` `ASSERT_TIMESTAMP_NOT_NULL (MSG VARCHAR(256), VALUE TIMESTAMP)`	Checks if the given timestamp is not null.
`ASSERT_DATE_EQUALS (EXPD DATE, ACTL DATE)` `ASSERT_DATE_EQUALS (MSG VARCHAR(256), EXPD DATE, ACTL DATE)`	Compares two dates.
`ASSERT_DATE_NULL (VALUE DATE)` `ASSERT_DATE_NULL (MSG VARCHAR(256), VALUE DATE)`	Checks if the given date is null.
`ASSERT_DATE_NOT_NULL (VALUE DATE)` `ASSERT_DATE_NOT_NULL (MSG VARCHAR(256), VALUE DATE)`	Checks if the given date is not null.
`ASSERT_TIME_EQUALS (EXPD TIME, ACTL TIME)` `ASSERT_TIME_EQUALS (MSG VARCHAR(256), EXPD TIME, ACTL TIME)`	Compares two times.
`ASSERT_TIME_NULL (VALUE TIME)` `ASSERT_TIME_NULL (MSG VARCHAR(256), VALUE TIME)`	Checks if the given time is null.
`ASSERT_TIME_NOT_NULL (VALUE TIME)` `ASSERT_TIME_NOT_NULL (MSG VARCHAR(256), VALUE TIME)`	Checks if the given time is not null.

Decimals

Signature	Description
`ASSERT_DEC_EQUALS (EXPD DOUBLE, ACTL DOUBLE)` `ASSERT_DEC_EQUALS (MSG VARCHAR(256), EXPD DOUBLE, ACTL DOUBLE)`	Compares two decimals.
`ASSERT_DEC_NULL (VALUE DOUBLE)` `ASSERT_DEC_NULL (MSG VARCHAR(256), VALUE DOUBLE)`	Checks if the given decimal is null.
`ASSERT_DEC_NOT_NULL (VALUE DOUBLE)` `ASSERT_DEC_NOT_NULL (MSG VARCHAR(256), OBJECT DOUBLE)`	Checks if the given decimal is not null.

Integers

Name	Description
`ASSERT_INT_EQUALS (EXPD BIGINT, ACTL BIGINT)` `ASSERT_INT_EQUALS (MSG VARCHAR(256), EXPD BIGINT, ACTL BIGINT)`	Compares two integers.
`ASSERT_INT_NULL (VALUE BIGINT)` `ASSERT_INT_NULL (MSG VARCHAR(256), VALUE BIGINT)`	Checks if the given integer is null.
`ASSERT_INT_NOT_NULL (VALUE BIGINT)` `ASSERT_INT_NOT_NULL (MSG VARCHAR(256), VALUE BIGINT)`	Checks if the given integer is not null.

Strings

Signature	Description
`ASSERT_STRING_EQUALS (EXPD VARCHAR(32672), ACTL VARCHAR(32672))` `ASSERT_STRING_EQUALS (MSG VARCHAR(256), EXPD VARCHAR(32672), ACTL VARCHAR(32672))`	Compares two strings.
`ASSERT_STRING_NULL (STRING VARCHAR(32672))` `ASSERT_STRING_NULL (MSG VARCHAR(256), STRING VARCHAR(32672))`	Checks if the given string is null.
`ASSERT_STRING_NOT_NULL (STRING VARCHAR(32672))` `ASSERT_STRING_NOT_NULL (MSG VARCHAR(256), STRING VARCHAR(32672))`	Checks if the given string is not null.

Tables

Signature	Description
`ASSERT_TABLE_EQUALS (EXPD_SCHEMA VARCHAR(128), EXPD_TABLE_NAME VARCHAR(128), ACTL_SCHEMA VARCHAR(128), ACTL_TABLE_NAME VARCHAR(128))` `ASSERT_TABLE_EQUALS (MSG VARCHAR(256), EXPD_SCHEMA VARCHAR(128), EXPD_TABLE_NAME VARCHAR(128), ACTL_SCHEMA VARCHAR(128), ACTL_TABLE_NAME VARCHAR(128))`	Compares two tables in structure and content.
`ASSERT_TABLE_EMPTY (SCHEMA VARCHAR(128), TABLE_NAME VARCHAR(128))` `ASSERT_TABLE_EMPTY (MSG VARCHAR(256), SCHEMA VARCHAR(128), TABLE_NAME VARCHAR(128))`	Checks if the given table is empty.
`ASSERT_TABLE_NON_EMPTY (SCHEMA VARCHAR(128), TABLE_NAME VARCHAR(128))` `ASSERT_TABLE_NON_EMPTY (MSG VARCHAR(256), SCHEMA VARCHAR(128), TABLE_NAME VARCHAR(128))`	Checks if the given table is not empty.

Execution

Signature	Type	Description
CLEAN()	Proc	Cleans the environment. This procedure sets to null of sessions variables used by the framework.
CLEAN_LAST_EXEC	Proc	Cleans the values of the last execution. This is mainly used to test the framework itself (self-testing).
CLEAN_TEST_RESULT()	Proc	Clean some environment variables. This procedure is used for self-testing of db2unit.
CREATE_TAP_REPORT	Proc	Creates a TAP report. The output is generated in TAP v13.
EXPORT_TESTS_LIST()	Proc	For UNIX/Linux/Mac OS X: Generates a file that contains the commands to execute all test suites registered in the database. This is useful to integrate the framework with an external tool.
GET_CURRENT_EXEC_ID	Func	Retrieves the ID of the current test suite execution. This is mainly used to test the framework itself (self-testing).
GET_CURRENT_TEST_SUITE_NAME	Func	Retrieves the test suite name of the current execution. This is mainly used to test the framework itself (self-testing).
GET_LAST_EXEC_ID()	Func	Retrieves the ID of the last test suite execution. If there has not been executed any test suite in the current session, this will return null. This is mainly used to test the framework itself (self-testing).
GET_LAST_TEST_SUITE_NAME	Func	Retrieves the test suite name of the most recent execution. This is mainly used to test the framework itself (self-testing).
GET_LAST_EXECUTION_ORDER()	Proc	Returns a result set with the order of the tests of the most recent execution in the current session.
LICENSE()	Proc	Returns a result set with the license of this framework.
RANDOM_SORT(RANDOM BOOLEAN)	Proc	Changes the configuration of the current session to execute the tests of the test suites in random order or not.
REPORT_RECENT_EXECUTIONS()	Proc	Returns a result set of the most recent execution for all test suites registered in the database. This shows the quantity of test that passed, failed or hit errors.
REGISTER_SUITE(SCHEMA_NAME VARCHAR(128))	Proc	Registers the name of a test suite in the table of test suites. This is useful when configuring an environment, and all test suites will be executed. This procedure does not execute the test suite.
RELEASE_LOCK(SCHEMA_NAME VARCHAR(128))	Proc	Release the lock associated with the test suite name given if exist. When a test suite is executed, a lock is created during the execution to prevent concurrent executions. If the execution did not finish correctly, the lock is kept in the database. This procedure release this lock.
RELEASE_LOCKS()	Proc	Release all locks from the database. This is useful when the database was stopped, or several connections were forced.
REPORT_RECENT_EXECUTIONS	Proc	Shows a report of the most recent execution of all test suites.
RESET_TABLES()	Proc	Delete the values from all tables of the framework. This brings the installation as when it was installed. All executions, all test suites are deleted.
RUN_SUITE(SCHEMA_NAME VARCHAR(128), PREV_EXEC_ID INT, TEST_NAME VARCHAR(128))	Proc	Main procedure of this project. The schema_name is the test suite. The prev_exec_id is the ID of a previous execution that execute the suite in the same random order; if this is null, the order will be randomized again. Test_name is the name of the test that is going to be executed; if null, all tests of the given suite are executed. This procedure registers the test suites in the test suites table.
RUN_SUITE(SCHEMA_NAME VARCHAR(128), PREV_EXEC_ID INT)	Proc	A wrapper of the previous procedure.
SET_AUTONOMOUS(AUTONOMOUS BOOLEAN)	Proc	Changes the configuration of the current session to execute the test suites in autonomous mode or not.

ER diagram