



pathvIEW

Code Coverage Testing for CA Gen

pathvIEW is a Code Coverage Testing tool designed to ensure that a CA Gen application or changes to the application have been properly tested.

pathvIEW has been designed to provide an easy method of capturing code coverage data for CA Gen applications. It has been optimized to provide a minimal runtime overhead, and integration with IET's GuardEn product allows developers, testers and managers to rapidly view code coverage results at a variety of levels of detail, for example, by Change Request, model, application or load module.

Code Coverage Testing describes the degree to which an application or module has been tested based on the number of statements that have been executed. Code Coverage coupled with an intelligent analysis of the results can greatly improve the quality of the application. Whilst the ideal scenario is that 100% of the statements are tested, often this is not practical, for example where there is exception logic that should never occur, and so a lower threshold is usually acceptable, for example 85 or 90% of statements should have been executed.

pathvIEW enables code coverage testing at the action block statement level. It logs the statements that are executed when running a Gen-generated application.

By analysing the code coverage results, developers and testers can improve their test cases for functions or statements that have not been adequately tested. When a statement or block of statements are not executed by the test cases, it is important that the reason is understood rather than simply trying to alter the test cases to ensure that the statements are executed. It might be that the statements are redundant, can never be executed or that the test cases fail to adequately test all branches in the logic.

Architecture

The diagram below illustrates the various pathvIEW components and the processes involved.

- (1) Standard CA Gen source code is generated (from the toolset or encyclopaedia).

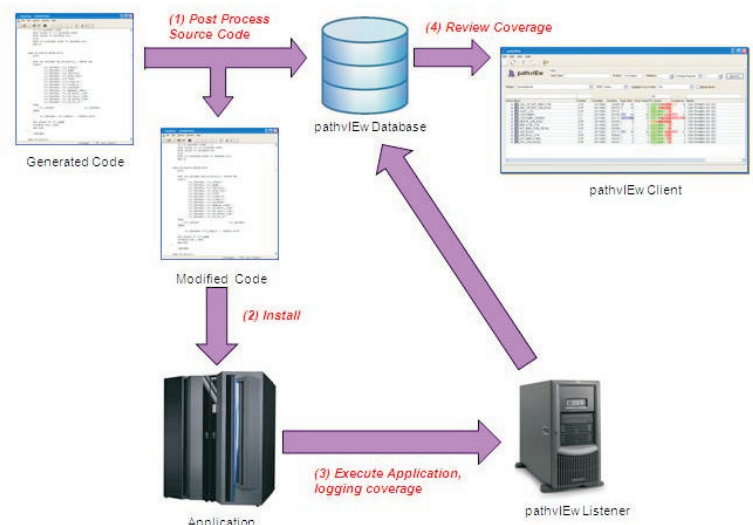
Prior to compilation, it is modified by the pathvIEW Source Code Post-Processor, which adds in calls to the pathvIEW runtime to log statement executions.

- (2) The modified source code is then installed and made available for execution in the testing region.

- (3) On execution of the application, statement coverage data is gathered by the pathvIEW runtime, and sent (using TCP/IP) to the pathvIEW Listener at the end of the transaction.

The Listener stores the execution results in the pathvIEW database. The Listener is a multi-threaded application that supports the logging from multiple application executables in parallel.

- (4) The code coverage data can be viewed at any time using the pathvIEW client. Summary information provides a quick overview of the degree of testing. Filters are available to select which action blocks are displayed, for example by Change Request or Test Case.





Reviewing Results

The pathIEW client provides an overview of the Gen procedure steps and action blocks and the percentage of statements tested. Various filters are available to allow developers and testers to focus on the desired objects, for example by GuardIEn Change Request, Release Pack, Test Case or using name or model name wild cards.

Action Block	Variant	Gendate	Gentime	Num Stmt	Num Tested	% Tested	Complexity	Model
PATHVIEW	CLT1	20110830	145705	701	535	76%	164	GUARDIEN CS 8.1
PATHVIEW_COMPARE	CLT1	20110825	121542	384	318	82%	117	GUARDIEN CS 8.1
PATHVIEW_CONVERT_BAR	CLT1	20110825	110344	9	9	100%	6	GUARDIEN CS 8.1
PATHVIEW_COUNT_COMPLEXITY	SVR1	20110826	100114	84	66	78%	31	GUARDIEN CSE BATCH 8.1
PATHVIEW_DETAIL	CLT1	20110826	121252	564	473	83%	156	GUARDIEN CS 8.1
PATHVIEW_DETAIL_SVR	SVR1	20110822	163532	46	43	93%	15	GUARDIEN CS 8.1
PATHVIEW_EXPORT	CLT1	20110819	093135	83	81	97%	5	GUARDIEN CS 8.1
PATHVIEW_HOUSEKEEP	SVR1	20110817	174706	123	91	73%	19	GUARDIEN CSE BATCH 8.1
PATHVIEW_INSTANCE_SVR	SVR1	20110819	103239	59	48	81%	25	GUARDIEN CS 8.1
PATHVIEW_LIST_SVR	SVR1	20110825	132002	699	429	61%	70	GUARDIEN CS 8.1
PATHVIEW_SHOW_PATH	CLT1	20110831	094908	143	133	93%	49	GUARDIEN CS 8.1

Detailing the action block opens the Statement Coverage window.

This shows the statements that have been executed in green and those not executed in red with a useful overview of the entire action diagram.

The example below indicates that the 'SHOW TEST EXECUTIONS' logic has not been fully tested.

Select a test case to view the tested statements or enter a test case filter using % as a wild card

Test Case: <ALL> Start Expanded

Statements: 525 Complexity: 237 Tested: 450 Manually Flagged: 0

Path	Statement	Execution Status
+	EVENT: SELECT_TEST_CASE	Executed (Green)
+	SET out_filter test_case name TO SPACES	Executed (Green)
+	SET out_filter test_case id TO 0	Executed (Green)
+	GET ROW HIGHLIGHTED IN group_out_test_cases STARTING AT 1	Executed (Green)
+	GIVING SUBSCRIPT OF group_out_test_cases	Executed (Green)
+	++IF SUBSCRIPT OF group_out_test_cases > 0	Executed (Green)
+	MOVE out_g test_case TO out_filter test_case	Executed (Green)
+	++IF out_filter test_case name = "ALL"	Executed (Green)
+	SET out_filter test_case name TO SPACES	Executed (Green)
+	++	Executed (Green)
+	++	Executed (Green)
+	++IF out_filter test_case name = SPACES	Executed (Green)
+	ENABLE COMMAND "ADD"	Executed (Green)
+	ENABLE COMMAND "REMOVE"	Executed (Green)
+	ELSE	Executed (Green)
+	DISABLE COMMAND "ADD"	Executed (Green)
+	DISABLE COMMAND "REMOVE"	Executed (Green)
+	++	Executed (Green)
+	SET temp_tirevent ieif_supplied command TO "REFRESH"	Executed (Green)
+	USE tirevent	Executed (Green)
+	WHICH IMPORTS: Work View temp_tirevent ieif_supplied TO Work	Executed (Green)
+	View import ieif_supplied	Executed (Green)
+	++	Executed (Green)
+	EVENT: SHOW_TEST_EXECUTIONS	Not Fully Executed (Red)
+	++FOR SUBSCRIPT OF group_out FROM 1 TO LAST OF group_out BY 1	Not Fully Executed (Red)
+	++IF out_g ieif_supplied select_char = ""	Not Fully Executed (Red)
+	++IF out_g ab_statement number = 0	Not Fully Executed (Red)
+	MOVE out_g ab_statement TO out_selected ab_statement	Not Fully Executed (Red)
+	EXIT STATE IS link_to_statement_executions	Not Fully Executed (Red)
+	ESCAPE	Not Fully Executed (Red)
+	++	Executed (Green)
+	++	Executed (Green)
+	EVENT: EXIT	Executed (Green)
+	EXIT STATE IS return_from_link	Executed (Green)
+	++	Executed (Green)

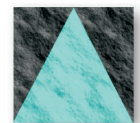
Expand All Contract All Show Path Hide Path Flag Tested Flag Untested All Executions Close

Green=Tested, Blue=Manual, Cyan=Merged, Pink=Merged Manual, Red=Untested

Benefits

- ✓ Ensures all statements are tested when 100% coverage required.
- ✓ Ensure changed modules are tested when performing a complete application test for a new release.
- ✓ Provides a quick overview by Change Request or Release of the proportion of changed modules that have been tested and the degree to which statements have been executed.
- ✓ Improves test cases by highlighting code that has not been tested.
- ✓ Identifies redundant code (statements are never executed).
- ✓ Identify duplicate test cases and hence simplify the testing process.
- ✓ 'Show Path' feature helps testers easily identify the path through the logic to the selected statement, thus enabling them to modify their testing to cover the selected statement.
- ✓ Minimal runtime overhead enables statement logging to be enabled by default for all test regions.
- ✓ Provides a capability that cannot easily be replicated by other means.
- ✓ Simple TCP/IP architecture enables rapid deployment.

**For more information, call
+44 1225 863060
e-mail information@iet.co.uk,
or visit us at www.iet.co.uk**



Information Engineering Technology

