

TITAN 7.2.1 CHANGE LOG (RELATIVE TO 7.2.PLO)

2021-03-25

HIGHLIGHTS



› General

- Titan version numbering changed to X.Y.Z format

› C++ compiler

- Object Oriented features bugfixes ([568694](#), [568714](#), [568716](#), [568742](#), [568743](#), [568745](#))
- Config file parsing enhancement ([570921](#))
- XML decoding bugfixes ([570707](#), [569238](#))

› Eclipse Plug-ins

- Support for Eclipse IDE Dark theme in TTCN-3 editor code colorization
- Java code generation and execution simplified

TITAN VERSION NUMBERING CHANGED TO X.Y.Z FORMAT



- › From now on Titan version numbers will consist of 3 numerals separated by 2 dots: X.Y.Z, where
 - X: Major version (Major version will step when backward-incompatible change is delivered)
 - Y: Sub-release version. We release new software regularly, this number indicates the steps inside the major releases.
 - Z: Minor version: When it is business justifiable we release on demand versions, the last number indicates an intermittent release.

OBJECT ORIENTED FEATURES BUGFIXES



- › Bugs [568694](#), [568714](#), [568716](#), [568742](#), [568743](#), [568745](#) are part of the effort to fully implement the TTCN-3 Object Oriented Language Extension v1.1.1 ([Link to Standard](#)) in the C++ compiler
- › Implementation is not yet complete (e.g. exception handling has not yet been covered) and will continue in upcoming releases. Also the implementation will be aligned to the release 1.2.1 of the standard.

CONFIG FILE PARSING ENHANCEMENT



- › Bug [570921](#)
- › Macro multiplication in config file implemented.
- › Limitation: whitespaces cannot be used between the macro references and the asterisk (*).
- ›
- › For example:
 - `RESULT := ${DEF_1}*${DEF_2}*${DEF_3} # works`
 - `RESULT := ${DEF_1} * ${DEF_2} * ${DEF_3} # doesn't work`

XML DECODING BUGFIXES



- › The following bugs are fixed:
 - [Bug 570707](#) - XML decoding error
 - [Bug 569238](#) - xsd2ttcn creates cyclical definitions

SUPPORT FOR ECLIPSE IDE DARK THEME IN TTCN-3 EDITOR CODE COLORIZATION



- › Code colorization in dark mode of Eclipse IDE modified to be much more ergonomic visually

A screenshot of the Eclipse IDE interface in dark theme. The main editor window displays TTCN-3 code with syntax highlighting. The code includes function definitions, a class definition, and various control structures. The IDE's interface elements, such as the Project Explorer on the left and the Console/Problems/Error Log at the bottom, are also visible in the dark theme.

```
File Edit Navigate Search Project Run Window Help
Toggle Comment
Project Explorer
Regression_Test_java
Test2[Shared] [f-w]
TTCN3HashMap.ttcn
TTCN3LinkedList.ttcn
TTCN3Queue.ttcn
TTCN3Stack.ttcn
TTCN3TreeSet.ttcn

process function hashFunction (key: string) {
  if (other of hashFunction) {
    var hashValue: integer := other -> hashFunction;
    return key.hashCode() + hashValue;
  } else {
    return false;
  }
}

public function hashCode() return integer {
  return key.hashCode();
}

public function toString() return universal charstring {
  var universal charstring value := "HashFunction (key: " & key.toString() & ", val: " & val.toString() & ")";
  return value;
}

type class LinkedListMap extends HashMap {
  const integer MAXIMUM_SIZE := 97; //largest prime number under 100
  //var table: record table
  var LinkedList table;

  create() {
    table := LinkedList.create();
    for (var integer i := 0; i < MAXIMUM_SIZE; i := i + 1) {
      var LinkedList l := LinkedList.create();
      table.add(i);
    }
  }

  private function getIndex(Storage key) return integer {
    var integer index := key.hashCode();
    return index mod MAXIMUM_SIZE;
  }

  public function put(object key: object, object value: object) /"exception Exception"/ {
    if (key instanceof Storage) {
      var Storage storage := key;
      var hashValue := hashCode(key);
      var hashValue := hashCode(key);
      insert(hashValue);
    }
  }

  private function insert(Storage data) {
    var integer index := getIndex(data);
    /* XXX: array should store the hash table instead of linked list
    if (not isEmpty(table[index])) {
      table[index] := LinkedList.create();
    }
    var LinkedList tree := table[index];
    tree.add(data);
    */
    var object hashValue := getIndex(data);
    var LinkedList hashValue := hashValue;
    if (hashValue instanceof Storage) {
      insert(hashValue);
    }
  }

  public function get(object key: object) /"exception Exception"/ return object {
    if (key instanceof Storage) {
      var integer index := getIndex(key);
      /* XXX: array should store the hash table instead of linked list
      if (not isEmpty(table[index])) {
        table[index] := LinkedList.create();
      }
    }
  }
}
```

TITAN JAVA PROJECT EXECUTION SIMPLIFIED



- › Launch shortcuts are available for Titan Java projects
- › Using the launch shortcut, a default launch configuration is generated for the project, and then the test execution is automatically started
- › The default settings are available and customizable in the *Create, manage, and run configuration* dialog window
- › Further details are available in Section 7 of the [TITAN Executor user guide](#)



ERICSSON