

Release Notes for TITAN TTCN-3 Test Executor

Elemer Lelik, Lenard Nagy

Version 11.2.1, 2026-08-05

Table of Contents

1. Introduction . . . 1
2. What's new in this version? . . . 2
   Version 11.2.1 . . . 2
3. Version 11.2.0 . . . 3
   3.1. Version 11.1.0 . . . 3
   3.2. Version 11.0.0 . . . 3
   3.3. Version 10.1.2 . . . 3
   3.4. Version 10.1.1 . . . 3
   3.5. Version 10.1.0 . . . 4
   3.6. Version 10.0.0 . . . 4
   3.7. Version 9.0.0 . . . 4
   3.8. Version 8.3.0 . . . 5
   3.9. Version 8.2.0 . . . 5
   3.10. Version 8.1.0 . . . 6
   3.11. Version 8.0.0 . . . 6
   3.12. Version 7.2.2 . . . 6
   3.13. Version 7.2.1 . . . 7
   3.14. Version 7.2.0 (7/CAX 105 7730 R2A) . . . 7
   3.15. Version 7.1.0 (CRL 113 200/7 R1A) . . . 7
   3.16. Version 6.6.1 (CRL 113 200/6 R6B) . . . 8
   3.17. Version 6.6.0 (CRL 113 200/6 R6A) . . . 8
   3.18. Version 6.5 (CRL 113 200/6 R5A) . . . 9
   3.19. Version 6.4 (CRL 113 200/6 R4A) . . . 9
   3.20. Version 6.3 (CRL 113 200/6 R3A) . . . 10
   3.21. Version 6.2 (CRL 113 200/6 R2A) . . . 11
   3.22. Version 6.1 (CRL 113 200/6 R1A) . . . 12
   3.23. Version 5.5 (CRL 113 200/5 R5A) . . . 12
   3.24. Version 5.4 (CRL 113 200/5 R4A) . . . 13
   3.25. Version 5.3 (CRL 113 200/5 R3A) . . . 13
   3.26. Version 5.2 (CRL 113 200/5 R2A) . . . 14
   3.27. Version 5.1 (CRL 113 200/5 R1A) . . . 14
   3.28. Version 4.2 (CRL 113 200/4 R2A) . . . 15
   3.29. Version 4.1 (CRL 113 200/4 R1A) . . . 15
   3.30. Version 3.2 . . . 16
   3.31. Version 3.1 . . . 17
   3.32. Version 2.2 . . . 17
   3.33. Version 2.1 . . . 17
   3.34. Version 1.10 . . . 18

3.35. Version 1.9 . . . 18
3.36. Version 1.8 . . . 19
3.37. Version 1.7 . . . 20
3.38. Version 1.6 . . . 21
3.39. Version 1.5 . . . 22
3.40. Version 1.4 . . . 23
3.41. Version 1.3 . . . 23
3.42. Version 1.2 . . . 24
3.43. Version 1.1 . . . 24
4. Version history . . . 26
4.1. Version 7/CAX 105 7730 R2A . . . 26
4.2. Version CRL 113 200/7 R1A . . . 26
4.3. Version CRL 113 200/6 R6B . . . 27
4.4. Version CRL 113 200/6 R6A . . . 27
4.5. Version CRL 113 200/6 R5A . . . 28
4.6. Version CRL 113 200/6 R4A . . . 28
4.7. Version CRL 113 200/6 R3A . . . 29
4.8. Version CRL 113 200/6 R2A . . . 29
4.9. Version CRL 113 200/6 R1A . . . 30
4.10. Version CRL 113 200/5 R5A . . . 31
4.11. Version CRL 113 200/5 R4A . . . 32
4.12. Version CRL 113 200/5 R3A . . . 32
4.13. Version CRL 113 200/5 R2A . . . 32
4.14. Version CRL 113 200/5 R1A . . . 33
4.15. Version CRL 113 200/4 R2A . . . 33
4.16. Version CRL 113 200/4 R1A . . . 34
4.17. Version CRL 113 200/3 R2A . . . 34
4.18. Version CRL 113 200/3 R1A . . . 35
4.19. Version CRL 113 200/2 R2A . . . 35
4.20. Version CRL 113 200/2 R1A . . . 35
4.21. Version CRL 113 200/1 R10A . . . 36
4.22. Version CRL 113 200/1 R9B . . . 37
4.23. Version CRL 113 200/1 R9A . . . 37
4.24. Version 1.8.pl7 . . . 38
4.25. Version 1.8.pl6 . . . 38
4.26. Version 1.8.pl5 . . . 39
4.27. Version 1.8.pl4 . . . 39
4.28. Version 1.8.pl3 . . . 39
4.29. Version 1.8.pl2 . . . 41
4.30. Version 1.8.pl1 . . . 43
4.31. Version 1.8.pl0 . . . 43

4.32. Version 1.7.pl4 . . . 45
4.33. Version 1.7.pl3 . . . 49
4.34. Version 1.7.pl2 . . . 50
4.35. Version 1.7.pl1 . . . 51
4.36. Version 1.7.pl0 . . . 55
4.37. Version 1.6.pl5 . . . 58
4.38. Version 1.6.pl4 . . . 61
4.39. Version 1.6.pl3 . . . 66
4.40. Version 1.6.pl2 . . . 70
4.41. Version 1.6.pl1 . . . 71
4.42. Version 1.6.pl0 . . . 74
4.43. Version 1.5.pl8 . . . 78
4.44. Version 1.5.pl7 . . . 81
4.45. Version 1.5.pl6 . . . 85
4.46. Version 1.5.pl5 . . . 86
4.47. Version 1.5.pl4 . . . 88
4.48. Version 1.5.pl3 . . . 89
4.49. Version 1.5.pl2 . . . 91
4.50. Version 1.5.pl1 . . . 93
4.51. Version 1.5.pl0 . . . 94
4.52. Version 1.4.pl5 . . . 96
4.53. Version 1.4.pl4 . . . 96
4.54. Version 1.4.pl3 . . . 97
4.55. Version 1.4.pl2 . . . 99
4.56. Version 1.4.pl1 . . . 99
4.57. Version 1.4.pl0 . . . 100
4.58. Version 1.3.pl0 . . . 102
4.59. Version 1.2.pl4 . . . 104
4.60. Version 1.2.pl3 . . . 105
4.61. Version 1.2.pl2 . . . 106
4.62. Version 1.2.pl1 . . . 108
4.63. Version 1.2.pl0 . . . 109
4.64. Version 1.1.pl10 . . . 110
4.65. Version 1.1.pl9 . . . 111
4.66. Version 1.1.pl8 . . . 112
4.67. Version 1.1.pl7 . . . 113
4.68. Version 1.1.pl6 . . . 113
4.69. Version 1.1.pl5 . . . 114
4.70. Version 1.1.pl4 . . . 114
4.71. Version 1.1.pl3 . . . 115
4.72. Version 1.1.pl2 . . . 116

4.73. Version 1.1.pl1 . . . 116
4.74. Version 1.1.pl0 . . . 116
4.75. Version 1.0 . . . 118
5. Changes of Test Port API . . . 119
5.1. Version 1.8.pl6 . . . 119
5.2. Version 1.8.pl5 . . . 119
5.3. Version 1.8.pl4 . . . 119
5.4. Version 1.8.pl3 . . . 119
5.5. Version 1.8.pl2 . . . 119
5.6. Version 1.8.pl1 . . . 119
5.7. Version 1.8.pl0 . . . 120
5.8. Version 1.7.pl4 . . . 120
5.9. Version 1.7.pl3 . . . 120
5.10. Version 1.7.pl2 . . . 120
5.11. Version 1.7.pl1 . . . 120
5.12. Version 1.7.pl0 . . . 120
5.13. Version 1.6.pl5 . . . 121
5.14. Version 1.6.pl4 . . . 121
5.15. Version 1.6.pl3 . . . 121
5.16. Version 1.6.pl2 . . . 121
5.17. Version 1.6.pl1 . . . 121
5.18. Version 1.6.pl0 . . . 121
5.19. Version 1.5.pl8 . . . 121
5.20. Version 1.5.pl7 . . . 121
5.21. Version 1.5.pl6 . . . 122
5.22. Version 1.5.pl5 . . . 122
5.23. Version 1.5.pl4 . . . 122
5.24. Version 1.5.pl3 . . . 122
5.25. Version 1.5.pl2 . . . 122
5.26. Version 1.5.pl1 . . . 122
5.27. Version 1.5.pl0 . . . 122
5.28. Version 1.4.pl5 . . . 122
5.29. Version 1.4.pl4 . . . 122
5.30. Version 1.4.pl3 . . . 123
5.31. Version 1.4.pl2 . . . 123
5.32. Version 1.4.pl1 . . . 123
5.33. Version 1.4.pl0 . . . 123
5.34. Version 1.3.pl0 . . . 123
5.35. Version 1.2.pl4 . . . 123
5.36. Version 1.2.pl3 . . . 123
5.37. Version 1.2.pl2 . . . 123

5.38. Version 1.2.pl1 . . . 124
5.39. Version 1.2.pl0 . . . 124
5.40. Version 1.1.pl10 . . . 124
5.41. Version 1.1.pl9 . . . 124
5.42. Version 1.1.pl8 . . . 124
5.43. Version 1.1.pl7 . . . 124
5.44. Version 1.1.pl6 . . . 125
5.45. Version 1.1.pl5 . . . 125
5.46. Version 1.1.pl4 . . . 125
5.47. Version 1.1.pl3 . . . 125
5.48. Version 1.1.pl2 . . . 125
5.49. Version 1.1.pl1 . . . 125
5.50. Version 1.1.pl0 . . . 125
6. References . . . 126

Chapter 1. Introduction

This document describes changes implemented in TITAN TTCN–3 toolset and the associated Test Port API from the initial TITAN TTCN–3 release through to the current release.

The document is organized as follows: What's new in this version? gives a short overview about the new features of major releases. Version history contains the detailed list of changes, including new functionalities and solved problems, throughout the history of TITAN TTCN–3 toolset. Changes of Test Port API summarizes the changes of the Test Port API in each version.

Chapter 2. What's new in this version?

Version 11.2.1

This release contains some new features and bugfixes:

- C++ compiler
  - Bugfixes and improvements
- Language Server & NextGen Plug-Ins (VS Code, Theia, LS Eclipse)
  - Bugfixes in Eclipse Plug-Ins, VSCode extension and Language Server

Chapter 3. Version 11.2.0

This release contains some new features and bugfixes:

- C++ compiler
  - Bugfixes and improvements
- Language Server & NextGen Plug-Ins (VS Code, Theia, LS Eclipse)
  - Bugfixes in Eclipse Plug-Ins, VSCode extension and Language Server

3.1. Version 11.1.0

This release contains some new features and bugfixes:

- C++ compiler
  - Bugfixes and improvements
- Language Server & NextGen Plug-Ins (VS Code, Theia, LS Eclipse)
  - Bugfixes in Eclipse Plug-Ins, VSCode extension and Language Server

3.2. Version 11.0.0

This release contains some new features and bugfixes:

- C++ compiler
  - Bugfixes (587, 614, 617, 620, 621, 633, 640, 655, 656, 732, 734, 735, 736, 737, 738, 739, 740, 741, 742, 744, 745)
- Language Server & NextGen Plug-Ins (VS Code, Theia, LS Eclipse)
  - Bugfixes (8, 10, 16, 17, 18, 34, 36, 44, 48, 49, 56, 58, 59, 61, 62)

3.3. Version 10.1.2

This patch release on top of Titan 10.1.1 contains some new features and bugfixes:

- C++ compiler
  - Bugfixes (727, 728, 729, 730)
  - Standard uplift (627)
- Eclipse Plug-Ins
  - Bugfixes (518)

3.4. Version 10.1.1

This patch release on top of Titan 10.1.0 contains some new features and bugfixes:

- C++ compiler
  - Bugfixes (629, 630, 643, 650, 667, 682, 726, 727)
- Language Server & NextGen Plug-Ins (VS Code, Theia, LS Eclipse)
  - Bugfixes (52)

3.5. Version 10.1.0

This release contains some new features and bugfixes:

- C++ compiler
  - Bugfixes (592, 695, 703, 705, 707, 708, 709, 710, 711 ,717, 722, 725)
  - Standard uplift (600, 626, 634, 635, 639, 641, 645, 646, 648, 666, 668, 669, 670, 671, 675)
  - PER Encoder (718, 719, 720, 721, 724)
    - Standalone ASN.1 encoder generation (712, 714, 715, 723)
- Eclipse Plug-Ins
  - Bugfixes (515, 516, 517)
- Language Server & NextGen Plug-Ins (VS Code, Theia, LS Eclipse)
  - Bugfixes (12, 14, 15, 22, 28, 30, 31, 32, 33, 35, 37, 39, 40, 41, 42, 43, 45)
  - General stability and robustness improvements
- Usage of WSL2 as Windows alternative

3.6. Version 10.0.0

This release contains some new features and bugfixes:

- C++ compiler
  - Bugfixes (694, 699, 700, 701, 702)
  - Standard uplift (647, 672, 677, 678, 679, 681, 683, 685, 686, 687, 693)
  - PER Encoder (691, see merge requests for details in this link)
- Eclipse Plug-Ins
  - Bugfixes (513, 514)
- Language Server
  - Bugfixes (1, 2, 3, 4)
- NextGen Plug-Ins (VS Code, Theia, LS Eclipse)
  - Bugfixes (1, 2, 4, 6, 11, 12)

3.7. Version 9.0.0

This release contains some new features and bugfixes:

- C++ compiler
  - Bugfixes (584, 591, 601, 604, 608, 611, 612, 636, 673, 676, 680, 688, 689, 690, 692, 696, 697)
- Eclipse Plug-ins
  - Bugfixes (507, 508, 509, 510)
  - IDE enhancements (417, 427, 439, 458, 477, 480, 484, 488, 489, 491)
- New EXPERIMENTAL features
  - Language Server
  - NextGen Plug-Ins: VS Code and Theia extensions

3.8. Version 8.3.0

This release contains some new features and bugfixes:

- C++ compiler
  - Bugfixes (545, 607, 612, 622, 623, 624, 631 , 651, 662)
  - Object Oriented features improvements – Full compliancy to v1.4.1 (661, 663, 664)
- Eclipse Plug-ins
  - Bugfixes (497, 498, 499, 500)

3.9. Version 8.2.0

To the memory of Elemer Lelik.

This release contains some new features and bugfixes:

- C++ compiler
  - Bugfixes (481, 576, 578, 579, 580, 581, 582, 583, 585, 586, 588, 592, 595, 597, 598, 599)
  - Object Oriented features improvements (533, 577, 596)
  - Use of octetstring in LENGTHTO (554)
- Eclipse Plug-ins
  - Bugfixes (404, 429, 432, 447, 448, 450, 461, 462, 465, 469, 475, 476)
  - New documented limitation (471)
  - Eclipse IDE UI enhancements (371, 442, 449, 451, 452, 453, 454, 457, 459, 460, 464, 466, 468)
  - JSON default attributes upgraded (416)
  - Use of octetstring in LENGTHTO (419)
  - Uplift minimum Java version from 1.6 to 1.8 (463)

3.10. Version 8.1.0

This release contains some new features and bugfixes:

- C++ compiler
  - Bugfixes (495, 506, 544, 555, 556, 557, 558, 559, 560, 563, 572, 573, 574, 575)
  - Object Oriented features improvements (533, 571)
  - Support Dynamic Matching (548, 549, 550)
  - Commit ID appears in version printout (561)
  - INTEGER(BIGNUM *other_value) Titan API call made public (551)
- Eclipse Plug-ins
  - Bugfixes (402, 406, 412, 420, 421, 426, 428, 430, 431, 440, 445, 446)
  - ANTLR version updated to 4.7.1 (409, 410)
  - Object Oriented feature enhancements (423, 424)
  - Eclipse IDE UI enhancements (397, 418, 441)
- Test Ports
  - Java implementation of IPL4 Test Port

3.11. Version 8.0.0

This release contains some new features and bugfixes:

- General
  - Titan source repositories moved to GitLab
- C++ compiler
  - Bugfixes (402, 408, 486, 546)
  - Exception handling implemented in Object Oriented features; the OOP implementation now complies to v1.1.1 of the related standard
  - OOP Standard Collections published
  - JSON default handling optimization (547)
  - XML handling bugfix (545, Backward-incompatible change!)
- Eclipse Plug-ins
  - Bugfixes (390, 399, 408, 411)
  - Java compiler can produce CI-integrable output (EXPERIMENTAL)

3.12. Version 7.2.2

This release contains only a bugfix for TPD file import (572399).

3.13. Version 7.2.1

This release contains some new features and bugfixes:

- 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

3.14. Version 7.2.0 (7/CAX 105 7730 R2A)

This release contains some new features and bugfixes.

New Features:

- Changes in Titan product code(Bug 565875)
- Changes in licensing(relevant for internal users only):
  - License ordering/extension page moved
  - License ordering/extension API to automate license renewal
- Changes in shipping libedit (Bug 565893)
- Titan image added to ADP marketplace(relevant for internal users only)
- Relaxing the C/C++ compiler version check for Titan 7.1.1 and upwards (Bug 564194)
- Object-oriented language features (Bug 563718)
- Allow unsafe universal charstring to charstring conversion (Bug 564585)
- Support for @default in the compiler and in the xsd2ttcn converter (Bugs 564919, 564920)
- Java-based Main Controller implemented
- Java-based JSON codec implemented

3.15. Version 7.1.0 (CRL 113 200/7 R1A)

This release contains major new features, backward incompatible changes and some bugfixes.

With this release we have also upgraded our license from Eclipse Plugin License 1.0 to Eclipse Plugin License 2.0.

New Features:

- The C side of the toolset will now support the Object Oriented Extension of the TTCN-3 standard (with some limitations).
- In the Designer plugin we have parallelized several algorithms resulting in 3-4* faster execution:
  - The semantic checking of modules inside the same project.
  - Projects in a project hierarchy, not dependent on each other, will not be able to be analyzed in parallel.
  - The Code Smell checkers of Titanium will also run in parallel.
- On the Java side of the toolset, we have added support for JSON encoding/decoding (with some limitations).

Backward incompatible changes:

- 'omit' matching mechanism shouldn't be allowed for union fields
- Compilation issue in the Java code generator when used with very large modules
- [UPDATE 3rd February, 2021] Escaping of TTCN-JSON strings now follows standard (escaping of "/" character changed!)

3.16. Version 6.6.1 (CRL 113 200/6 R6B)

This a bugfix release, with the following new features:

- Enhance performance of TIMER::get_min_expiration
- Legitimize compiler option '-F'
- Colorize compiler error/warning messages
- change the internal representation of octetstring from char[] to byte[] for efficiency
- CSN.1 L/H in the RAW codec
- change in how xsd files with NoTargetNamespace should be mapped to TTCN-3
- add support for structured type compatibility to the Java code generator
- add support for module parameter handling to the Java side runtime

3.17. Version 6.6.0 (CRL 113 200/6 R6A)

This release introduces two major new features in the form of two new experimental Eclipse plug-ins:

- a Java codegeneration plug-in, see: https://gitlab.eclipse.org/eclipse/titan/titan.core/blob/master/usrguide/java_referenceguide/JavaReferenceGuide.adoc

and also: https://gitlab.eclipse.org/eclipse/titan/titan.core/blob/master/usrguide/JavaCodegenExecTutorial.pdf

- a Refactoring plug-in, see: https://gitlab.eclipse.org/eclipse/titan/titan.EclipsePlug-ins/blob/

master/org.eclipse.titanium.refactoring/docs/Titanium_Refactoring_Description/Titanium_Refactoring_Description.adoc

Apart from the above, this version has the following new features of the Titan compiler and excutor:

- Printing symbolic version in 'compiler -v'
- Legitimized compiler option '-0'
- New command line option '-p' implemented for TTCN binaries (both in single mode and parallel mode), which lists all module parameters
- Support for real-time testing
- Implementation of map param/unmap param
- Extended 'default' JSON attribute for structured types
- Colorized compiler error/warning messages

3.18. Version 6.5 (CRL 113 200/6 R5A)

This release introduces real-time support according to TTCN-3 Language Extensions: TTCN-3 Performance and Real Time Testing as a major new feature.

This version has the following new features:

- License upgrade to EPL 2.0
- Documentation migrated to asciidoc
- Bugfixes for core, ASN.1, XML, RAW, OER codecs, XSD2TTCN
- New RAW coding instruction 'FORCEOMIT'
- New JSON attribute 'as map'
- Attribute 'text ... as ...' for JSON
- Adopt Makefilegen for the Debian packaging
- Support for real-time testing in TITAN Added compiler option '-I', which enables the real-time testing features mentioned here. The features are disabled by default, and the new keywords ('now', 'realtime' and 'timestamp') can be used as identifiers again (for backward compatibility).

Also added makefilegen option '-i', which activates this option for the compiler, and the makefile setting 'enableRealtimeTesting' in the TPD, which does the same thing.

- New flag for xsd2ttcn -o: generate all definitions into one module (called XSD_Definitions)

3.19. Version 6.4 (CRL 113 200/6 R4A)

This release is mainly a corrective release, it introduces no major new features. No backward incompatibilities are to be expected, with maybe one exception we see as a minor risk: the fix for Bug 533767 - RAW encoder ALIGN(right) is working according of specification of ALIGN(left) (and vice versa) for octetstring may induce under some unlikely circumstances an incompatible

behavior.

This version has the following new features:

- Implement verdict redirect for 'done' statement
- `str2float` should handle special float values
- RT2 record equality
- `string2ttcn` to filter patterns of visible characters in octetstrings
- Syntax to bind a variant attribute to multiple encodings
- TITAN build on Alpine Linux
- new tpd tag `disableUserInformation`
- runs on scope reduction (Titanium)
- Add discarding option to `setstate` operation
- Notify user if port is not mapped in translation mode
- Implement reference to port in translation function
- Implement extendable sequence coding in OER
- TAG and CROSSTAG for JSON encoder

3.20. Version 6.3 (CRL 113 200/6 R3A)

This version has the following new features:

- new compiler options:
  - e: enforce legacy handling of `encode` and `variant` attributes
  - O: disable OER encoder/decoder functions
  - D: disable user and time information generation in the generated files
- Support for multiple encodings
- Implement OER coder in TITAN (with the option to restrict generation of OER codecs)
- Implement OER negative testing
- Allowing to start functions with `out` and `inout` formal parameters
- Enable 'out' parameters for behavior functions in the 'start' operation support for dynamic erroneous attributes
- Allow translation ports to work as internal ports
- Allow sending and receiving during translation functions
- Flag to disable time and user information in the generated files
- Implement mtc and system clauses in `testcase` and `altstep` and functions
- Add runtime configuration setting for plain XML and JSON encodings
- Implement `json2cbor` and `cbor2json`

- Implement `json2bson` and `bson2json`
- JSON enc/dec: encoding enumerated values in number form
- Support `enableLegacyEncoding` in tpd
- Add the encoding legacy switch to tpd TEXT codec
- Add the encoding legacy switch to makefilegen
- Add support for NULL terminated string in RAW
- RAW: add offset option to `LENGTHTO` attribute
- RAW: Support also `⋯ bits` syntax in variant attributes

3.21. Version 6.2 (CRL 113 200/6 R2A)

This version has the following new features:

- new compiler options:
  - J: Compiler (and xsd2ttcn, makefilegen) option to read input files list from a text file
  - N: ignore UNTAGGED encoding instruction on top level unions (legacy behavior)
- support of encvalue/decvalue for ASN.1 types
- support for implicit call of PER codec external functions
- implemented: ports with translation capability
- support for concatenation of templates
- implemented any from clause and index redirects with the use of the @index modifier (see standard, chapters 21-23)
- support for dynamic erroneous attributes
- implemented @fuzzy support
- support for external functions for decmatch and @decoded
- no support of Solaris binaries from this release of Titan (older versions of course will continue to support Solaris)
- makefilegen more restrictive on name attribute of the referenced project
- makefilegen: remove generated headers dependency from all `.c` `.cc` files
- (This will revert the following bugs: Bug 499963 - The generated `Makefile` does not make full build when `-j` switch is present ; Bug 512688 - makefilegen: Incorrect `.c` and `.cc` compiling rule)
- XER: allow `anytype` to be xer enc/decodable
- JSON `as value` attribute extended for records/sets with one field and for the `anytype`
- **make archive** button in Eclipse
- support for `make port` command in Eclipse
- plug-ins upgraded to Jung 2.1

This list is not comprehensive; for details, see document embedded in PRI.

3.22. Version 6.1 (CRL 113 200/6 R1A)

This version has the following new features:

- support for `mctr reconf` command
- command line debugger
- advanced code splitting
- makefilegen capability to handle .xsd files
- makefilegen and compiler to handle file lists in files(`compiler -J` file or `makefilegen -J` file)
- new compiler switch for decreasing variant errorlevel from error to warning
- LTTng logger plug-in
- encvalue/decvalue for ASN.1 types
- Titan build for ARM/Raspberry Pi
- decmatch and @decoded
- istemplatekind
- select union
- @nocase
- Partial @deterministic support
- Storing parts of received messages

Incompatibilities:

- warning changed to error when '*' is used for mandatory elements
- infinity/NaN not allowed at timer start
- receive handling changed (receive(*) and receive(?) not allowed or restricted)

The above is not a comprehensive list; for all details , pls. check the document embedded in PRI.

3.23. Version 5.5 (CRL 113 200/5 R5A)

This version has the following new features:

- type substitutionGroup support
- allow using specific encode attribute strings to identify encode functions
- `ttcn2json`: extra keyword for restricted "as value" unions
- makefilegen shall generate `-Y` if tpd orders it
- user Debug classes
- negative testing with JSON encoder
- new compiler switch for decreasing variant errorlevel from error to warning

- makefilegen supports commenting out `OPENSSL_DIR` based on tpd setting
- activate emergency logging when a test fails
- `makefilegen -I` option
- RAW encoder for universal character string
- ISO 10646-conformant unicode syntaxes
- new internal functions: `encvalue2unichar`/`decvalue2unichar`,`any2unistr`
- `make port` command
- `checkstate port` operation
- clang support in Titan
- Eclipse Designer: implement fast algorithm
- config parser/editor based on ANTLR 4
- A number of TRs related to XML, Eclipse, JSON
- negative and positive conformance tests covering core language part of the standard added
- new document: statement of compliance covering Core language part of the standard
- legacy switches:
  - M: allow 'omit' in template value lists (legacy behavior) (artf692717)
  - B: allow selected union field to be unbound (legacy behavior) (artf717563)

3.24. Version 5.4 (CRL 113 200/5 R4A)

This version has the following new features:

- Refactored xsd2ttcn converter
- Eclipse plug-ins migrated from ANTLR 2 to ANTLR 4.
- 60 Eclipse plug-in related TRs and CRs implemented.
- Function calls with subreferences (artf550360)
- Template(present) accepts complement matching (artf564824)
- Integer to enumerated (artf590888)
- Support for IntX in RAW(artf607782)
- Module parameters can be initialized with module parameters (artf618367)
- Improved logformat to pretty-print XML and JSON

3.25. Version 5.3 (CRL 113 200/5 R3A)

This version has the following new features:

- TEXT codec to support universal character string (UTF-8).
- New Junit Logger plugin with extended logging.

- First version of the coverage/profiler tool.
- Stack trace displayed in case of segmentation fault or abort().
- Allow component and default types in module parameters.

3.26. Version 5.2 (CRL 113 200/5 R2A)

This version has the following new features:

- `Makefilegen -Z` option: Faster than the previous recursive linking method , support for dynamic linking, improved make archive
- `Makefilegen -H` option: support for partial build of hierarchical *.tpd structures.
- `Ttcn2json` improved ASN.1 handling, including parameterized types
- TR HS 34398 revoked.
- As the solution to TR HT 24380 caused performance problems, this was removed from RT1 (the default load test runtime)

3.27. Version 5.1 (CRL 113 200/5 R1A)

This version has the following new features:

- Changes in the assignment of charstring and universal charstring values to permit direct assignment of Unicode characters in editors with UTF-8 support.
- **Out parameter behavior changed: all out parameters are set to <unbound> at the start of the function. As this could cause incompatible behavior, a compiler option enforcing legacy behavior (`-Y`) was introduced.**
- A number of deprecated compiler options (`-E`, `-n`, `-N`, `-B`) were removed.
- New JSON codec variants "as value", "default".
- TTCN-3 type to JSON schema converter compiler option introduced.
- Eclipse plug-in improvements.
- Macro redefinition functionality for TITAN TTCN-3 Test Executor in the `[DEFINE]` section of the .cfg file.
- Nested concatenation operator `&=` in the `[MODULE_PARAMETERS]` section of the .cfg file
- Eclipse plug-in package and bundle id's (including extension point id's) have been changed due to open sourcing Titan. Their names start with **"org.eclipse.titan"** instead of **"com.ericsson.titan"**
- Legacy `mctr_gui` and `logbrowser` (based on Qt3 which lacks support in modern Linux versions) removed. The last version can still be obtained from older Titan packages.
- `Ctags` support removed due to licensing problems (`ctags` files can be obtained from older Titan releases).
- From this release, usage of 64-bit Cygwin is encouraged. A 32 bit version will not be released.
- Correction for newer openssl packages that break Titan license validation.

IMPORTANT	Titan releases previous to CRL 113 200/5 R1A will not work if openssl is upgraded beyond the critical level of release; the exact level depends on the Linux platform and version.

- **A correction for TR HT24380 (Error in manipulating dependent inout parameters - a record of and its element) may cause incompatible behavior (see TR for further details). When Titans' behavior might change compared to previous releases, a warning message- intended to help users to detect sequences of TTCN-3 code that need to be changed- will be displayed.**

3.28. Version 4.2 (CRL 113 200/4 R2A)

This version has the following new features:

- JSON encoding support.
- Support for various universal character string encodings (UTF-8, UTF-16LE, UTF-16BE, UTF-32LE, UTF-32BE).
- Built-in support for base64 encodings.
- Java executor API for Titan.
- Eclipse plug-in improvements.
- Configurable timestamp in console.
- Improved behavior in port congestion situations.
- Superfluous circular warnings for ASN.1 disabled.
- TEXT encoder debug logging.
- Several improvements regarding the XML encoding/decoding.
- T3Doc disabled in the Designer.
- The asciiart directory emptied to prevent interference with automated usage.

Important notes:

- As the referenced TTCN-3 standards for universal character string encodings and for JSON are not finalized yet, details of these (as in exact function names) may change.
- The following new keywords have been introduced in this release: `oct2unichar`, `unichar2oct`, `get_stringencoding`, `remove_bom`, `encode_base64`, `decode_base64`

3.29. Version 4.1 (CRL 113 200/4 R1A)

This version has the following new features:

- Catching Dynamic Test case errors – Adds the ability to survive DTEs in TTCN-3 code, for instance in case of long running load tests. Very similar to exception handling used in other languages.
- Lazy Parameter Evaluation – In formal parameters can be defined to be subject of lazy

evaluation: the expression used as actual parameter shall be evaluated only when the formal parameter is used (not at the function call); the evaluation is only done once.

- Titanium – new Eclipse plugin, a code quality analysis prototype for advanced users, available upon request.
- Usage statistics - Titan compiler, runtime and Titan Eclipse plug-in usages are collected for statistical purposes.
- Change of default error behavior for XML encoding from 'Warning' to 'Error' to align with the other Titan encoders.
- Template Module Parameters - TTCN-3 language extension, module parameters can be both values (standard) and templates (non-standard).
- `Ttcn2string()` predefined function - returns the parameter's value in TTCN-3 string representation. `String2ttcn()` predefined function - `Ttcn2string()` predefined function contrariwise.

NOTE Please make sure that your makefile contains the following part marked with red:

```
SOLARIS8_LIBS =-lresolv -lnsl -lsocket

LINUX_LIBS = -lpthread -lrt
```

3.30. Version 3.2

This version has the following new features:

- Support for distributed build using hierarchical `Makefiles` with new `ttcn3_makefilegen` command line options (`-r`, `-F`).
- New makefile target "library" is implemented. The pre-compiled objects can be collected to a library archive file(.a or .so), useful when the project hierarchy has rarely changing parts.
- Extended _.tpd file handling in makefilegen was introduced. `ttcn3_makefilegen` processes the `MakefileSettings` part of the _.tpd files. Benefits: *.tpd files extracted/created with Eclipse can be used in command line and usage of makefilepatch scripts can be hugely reduced or even eliminated.
- `ORDERED_INCLUDE` in configuration files is implemented; the includes will be strictly ordered.
- Clean-up after unsuccessful makefilegen execution, `symlinks` are now generated only if no errors were found during *.tpd file processing
- _.tpd file validation with `ttcn3_makefilegen`: the _.tpd file is validated with a schema that now is part of TITAN (`file $(TTCN3_DIR)/etc/xsd/TPD.xsd`); validation errors will prevent makefile generation.
- `Makefilegen`: override the working directory in _.tpd file: the working directory of top level project comes from top level _.tpd file by default; when using the –D switch the working directory will be the current directory.
- `Makefilegen` support for OSS Nokalva ASN.1 compiler is implemented. `Makefile` generation from

*.tpd file enables OSS Nokalva support without custom makefilepatch scripts

- Integration of DPMG(Diameter Protocol Module Generator) into the TITAN build system.
- Improved *.tpd file related documentation.
- Reduced nr of supported gcc versions. Supported versions are: 3.4.6 – 4.7.2
- Changes in supported platforms: Solaris versions from 5.10 are supported; Cygwin versions from 1.7 are supported. Earlier Solaris and Cygwin versions are not supported
- Titan Eclipse plugins support Eclipse versions from Eclipse 3.7.2 to Eclipse 4.2
- Java 1.6 is the minimum requirement
- A fourth Eclipse plug-in, Titanium, is released as a prototype. Update and maintenance of Titanium will be the responsibility of the Titanium project until further notice. For details pls. see https://ericoll2.internal.ericsson.com/sites/DUCI_SW_Technology/Titanium/default.aspx

3.31. Version 3.1

Version 3.1 has the following new features:

- Interface implemented for the TestStatistics tool
- All from in value list, subset, superset and permutation supported
- Embedded macro references in the `[DEFINE]` section - runtime (support in command line)
- Structured macro definitions in the `[DEFINE]` section - runtime (support in command line)
- Embedding TTCN-3 functions (limited functionality)

3.32. Version 2.2

Version 2.2 has the following new features:

- XML encoding is now supported for the hexstring and verdicttype TTCN-3 types
- Transparent functions were introduced to allow easier identification of failing tests in case of SourceInfo := Single.

3.33. Version 2.1

Version 2.1 has the following new features:

- The Titan Eclipse Designer's support for preprocessed TTCN-3 files has been improved.
- The performance of TEXT decoding has been improved.
- A logger plugin (JUnitLogger) is now delivered with Titan. It outputs XML files in the same format as JUnit. Using this logger plugin allows integrating of Titan with the Jenkins (Hudson) continuous integration tool.
- To allow JUnitLogger to receive the necessary information, the Titan Logger API has been slightly changed. Existing logger plugins will need to be rebuilt.

- In response to a TR (HP88760), the C++ interface of the OBJID class has been changed. The type of the elements in the internal storage of the OBJID class is now specified with a `typedef`, `objid_component`. Code which uses the indexing operators or directly accesses the element storage will need to be rewritten. It is a backward incompatible change and it affects users of the SNMP test port. A new version of the SNMP test port was released (CNL 113 344 R4B) compatible with the new Titan.
- `ttcn3_makefilegen` has a new flag `-P`, which prints out the list of files found in a given TPD recursively relative to a given directory.
- TTCN-3 level code coverage was implemented.
- Text hover for T3Doc in Eclipse was implemented.
- `mctr_gui`, `ttcn3_logbrowser`, `ctags`, Nedit, XEmacs support is part of the Titan package again.

3.34. Version 1.10

Version 1.10 has the following new features:

- Renaming refactoring was implemented in Titan Eclipse Designer. This feature provides TTCN-3 scope-aware renaming of declarations.
- Selection highlighting was implemented in Titan Eclipse Designer. When a variable name or function name or keyword is selected in the code, all the occurrences of the selected variable name or function name or keyword will be highlighted in the same file.
- Performance of `log2str()` was improved.
- Implicit omit support for module parameters was implemented.
- Append operation (`&=`) for list types in configuration files was implemented.
- Support of executing testcases with default parameters from command line and configuration file was added.
- Improved error recovery for the compiler. E.g. it can now stop on the first syntactic error and skip the semantic analysis.

3.35. Version 1.9

Version 1.9 has the following new features:

- With the release we have decided to change from the proprietary Titan versioning scheme, to the one used by Ericsson. From now on it will be much easier to decide if a new version is forward, backward compatible with a previous version. The versioning is also supported in the attributes of the modules, with some limitations. We only accept version numbers in 3 formats: R9A, CRL 113 200 R9A and CRL 113 200/1 R9A.
- With this release we removed all QT based GUI parts (`mctr_gui`) and `ctags` from the official Titan releases, as they have been in maintenance phase for the last year. NEdit and XEmacs parts are still available as downloadable components from our download pages.
- The import of imports feature declared in the newest TTCN-3 standard was implemented. This way it is now possible to recursively import import statements from other modules.

- IPv6 support for Titan's internal communication was implemented. This way Titan is now able to function properly when the MC and PTCs are located on an IPv6 network.
- The makefilegen tool in the command line package is now able to generate `Makefiles` from the information stored in .Tpd project descriptor files.
- It is now possible to find all reference pointing to a given declaration inside eclipse. Finding all references to a definition was implemented as a new kind of search in the Eclipse platform.
- The Executor plug-in will now be able to automatically merge the generated log files after execution.

3.36. Version 1.8

Version 1.8 has the following new features:

- The `testcase.stop` operation is now supported, allowing for the users to stop the execution of the actual `testcase` raising a dynamic `testcase` error with a custom explanation text.
- The `ispresent` predefined function was extended to operate on all structured types and fields as described in the 4.3.2 version of the TTCN-3 standard.
- The main features of the LogViewer eclipse feature can no be accessed from the Project Explorer too, so it is no longer required to switch to its custom navigator.
- It is now possible to configure the Executor feature and eclipse executed "launch configurations" to automatically merge the log files that were generated during execution. For the case of several consecutive executions it is now possible to configure the system, to remove the previous log files before a new execution.
- Added the negative testing feature allowing to generate invalid messages, and to send them to the SUT, to observe its reaction.
- With the help of emergency logging it is now possible to define different behaviors for logging in normal and in emergency situations.
- The performance of the LogViewer plug-in has been enhanced considerably, to support the processing of arbitrary large log files.
- Titan is no longer depending on the external Readline package. It has been replaced with Editline, which is now compiled into the delivered packages.
- A new project description format has been created to support exporting and importing the data of Titan projects in eclipse into a single file.
- The LogViewer eclipse plug-in was enhanced to work on larger files, with less resource consumption. Also it is now much better integrated with the rest of the toolset.
- Huge increase in the speed of the on-the-fly analysis in the Designer plug-in, with much more efficient memory usage when the incremental parsing option is turned on.
- The Designer now supports build configurations allowing switching between sets of build settings in a consistent way.
- The build action of Eclipse can now be invoked from the command line on two ways. One guaranteeing to build exactly as Eclipse is doing it, and one allowing the user to fine tune all of his settings.

- Support for the launch shortcut feature of eclipse was introduced allowing to create and initialize new launch configurations in an easier way.
- The base of the TTCN-3 standard used to describe the features and limitations of TITAN was changed from version v3.1.1 to v4.1.1
- The build process was enhanced with options for dynamic linking, advanced dependency refreshing, and with splitting the generated code into several files.
- The checking of subtypes in TTCN-3 and ASN.1 modules was enhanced considerably, and the on-the-fly semantic analyzer in the Designer plug-in was brought on the same level as the command line compiler is on.
- Introduced support for the module interface feature, allowing for the user to hide internal parts of a module from the other modules.
- Introduced the `testcasename()` and removed the `sizeoftype()` predefined function in accordance with the standard.
- Support for XML encoding and decoding is introduced, together with a new command line tool that converts XSD files into TTCN-3 modules.
- The `enum2int`, `encode` and `decode` predefined functions were introduced.
- It is now possible to use the `concatenation`, `replace`, `substr`, `lengthof` predefined functions on values of the set of, record of an array types.
- The implicit omit attribute is now supported.
- The TTCN-3 type anytype became supported with some restrictions.
- The runtime was split into two versions: one for function testing where much less code is generated, at the cost of somewhat degraded runtime performance; and one for load testing. Both are compatible with the interfaces of the original runtime.
- Both eclipse plug-ins were enhanced to be able to format and merge log files produced by an execution.
- The on-the-fly semantic analyzer of the Designer plug-in was considerably enhanced.
- The code quality checks done by the on-the-fly in the designer plug-in were extended to detect unused local and module level definitions too.
- The checking of the validity of the license file was introduced in the Designer plug-in, so as to protect it from unauthorized usage.
- The Designer plug-in was enhanced to be able to parse TTCN-3 files in an incremental manner, which should reduce the time required for analyzing a project from a few second, to a few times 10-2 seconds.
- The designer plug-in was extended with its own internal Makefile generator.

3.37. Version 1.7

Version 1.7 has the following new features:

- The naming convention of the generated C++ code has been revised to avoid potential name clashes between definitions. The definitions of each TTCN–3 and ASN.1 module is put into a

separate C++ namespace that corresponds to the module name. This eliminates all problems caused by definitions with identical names in different modules. The scope of C++ enum values that represent the values of TTCN–3 and ASN.1 enumerated types became narrower to avoid conflicts if the same element name appears in two different enumerated types.

- Extension (inheritance) of TTCN–3 component types and compatibility between different component types is now supported by the compiler.
- Dual-faced TTCN–3 ports, which can transform the incoming and outgoing messages, were introduced. Using this feature the compiler is capable of automatic generation of TTCN–3 external functions that perform encoding or decoding based on the built-in codecs (RAW, BER, TEXT).
- The Runtime GUI has become a stand-alone product. It is no longer part of the TTCN–3 Executor package.
- The logging functionality has been significantly enhanced. From now the types of events logged can be set using much finer granularity. Using the name of the component in the name of the log files also became possible.
- From now it is possible to assign actual parameters in a parameter list to a specific formal parameter from the formal parameters of the type.
- It is now possible to use assignment notation with array indices.
- The efficiency of connection handling of the Main Controller, the Parallel Test Components and the testports was greatly enhanced.
- The Eclipse Designer plug-in is now building an AST that is structurally equivalent to the on found in the compiler, and stores about the same amount of data. Thus increasing the amount of semantic errors that can be detected on-the-fly without invoking the build system.
- The logging of the `match` operation was made configurable through the `MatchingHints` logging option. If it is set in "Compact" mode (which is the default) the log record will be only a few lines long, instead of a few hundred lines long. In fact if there is only one field mismatching than the log will contain 1 line regardless of the size and structure of the value and template compared.

3.38. Version 1.6

Version 1.6 has the following new features:

- The semantic check for the TTCN–3 dynamic behavior descriptions (such as functions, altsteps, testcases) have been implemented, which means that all parts of TTCN–3 modules are now analyzed.
- The compiler generates the entire C++ code from the Abstract Syntax Tree, that is, the output of semantic analysis. This makes it possible to add support for some language constructs and perform code optimization in future versions. These were impossible with the old, parser-based code generator.
- The TTCN–3 parser of the compiler supports recovery from syntax errors. This means the compiler does not stop when a syntax error is detected, but it continues to analyze the input to find more errors.

NOTE In some cases it is not possible or worthwhile to recover from a syntax error[1].

- Code generation for in-line compound values and templates (including in-line modified templates) is now supported.
- The initializer sequences of constants and non-parameterized templates are ordered automatically so that forward references do not cause dynamic test case errors anymore.
- Support of TTCN–3 language constructs has been enhanced. There is full support of arrays, groups and attributes. Select-case and interleave statements as well as alive PTCs were implemented.
- Text encoding has been introduced.
- Function, altstep and testcase references are supported in TTCN–3 .
- Non-mandatory parameters (i.e. default values for formal parameters) are supported in TTCN–3 .
- Usage of C preprocessor on TTCN–3 modules is allowed.
- The Makefile generator has been significantly enhanced and moved from the compiler to a stand-alone program.
- The syntax of run-time configuration files has been enhanced to allow the use of macros and environment variables. Modularity (i.e. spreading configuration data over several files) is also supported.

3.39. Version 1.5

Version 1.5 has the following new features:

- The compiler supports the semantic analysis for all TTCN–3 definitions except the dynamic parts (i.e. functions, altsteps, testcases and control parts). This means that new checking routines were implemented for TTCN–3 subtype constraints, signatures, constants, templates and all definitions within component types.
- The compiler produces user-friendly error messages with file name and line number information and supports error recovery. It displays all error messages found in the input modules.
- The time needed for the compilation of generated C++ code was significantly reduced compared to 1.4.pl0. The saving can be more than 50 % in case of large projects.
- Procedure based TTCN–3 ports and the related communication operations are now supported with enhanced Test Port API.
- The run-time environment provides one unified API for both RAW and BER encoder/decoder functions.
- The internal structure of RAW encoder/decoder functions was significantly revised. This results in faster and more robust operation.

3.40. Version 1.4

Version 1.4 has the following new features:

- One integrated compiler for TTCN–3 and ASN.1. This allows the semantic analysis of test suites that import from ASN.1 modules without intermediate files. The command line switches of the previous two compilers were unified.
- The ASN.1 front-end of the compiler was significantly enhanced to handle X.681- X.683 extensions.
- The compiler supports the full semantic analysis of ASN.1 modules and semantic analysis of TTCN–3 type definitions. The output for other TTCN–3 definitions is still generated on the fly without checks.
- The compiler performs automatic reordering in the generated code for TTCN–3 types as well. This means, the generated C++ code will be always valid even if the type definitions use forward referencing.

NOTE	The forward referencing problem between TTCN–3 constants and templates is still unsolved. They must be declared in bottom-up order to get a working C++ code.

- The code generation routines of the previous compilers were fully re-used and no significant changes were made in the Base Library in order to preserve the stability of the executable tests.

3.41. Version 1.3

Version 1.3 has the following new features:

The Main Controller was completely re-designed in this version, which means the following advantages:

- There are no longer static limits on the number of simultaneously active PTCs.
- Improved and more comfortable command-line interface (with history, command completion, etc.).
- More robust and more efficient handling of large number of test components and/or port connections. Graceful recovery from run-time errors.
- Central configuration file handling and automatic distribution of configuration parameters.
- Version checking in MC to avoid inconsistent ETSes in distributed test environments.
- Faster execution of TTCN–3 configuration operations.
- Explicit control of PTC locations with user-defined constraints in addition to load balancing.
- A lot of Main Controller related bugs were fixed, which caused deadlocks in some situations before.
- TTCN–3 address type is supported by the compiler and the Test Port API.
- Lot of bug fixes in the compilers and the run-time environment.
- Re-organized chapters and clarifications in the user documentation.

3.42. Version 1.2

Version 1.2 has the following new features:

- The compiler supports the new, Edition 2 syntax of the TTCN–3 Core Language. The obsolete language elements that were supported in version 1.1 (e.g. named alternatives) are still accepted for backward compatibility, but a warning message is printed.
- The toolset contains a new ASN.1 compiler, which allows the importing of ASN.1 modules into TTCN–3 test suites. Like the TTCN–3 compiler, the ASN.1 compiler translates ASN.1 definitions to C++ code, which shall be used together with C++ output of TTCN–3 modules.
- The ASN.1 compiler performs a semantic analysis on its input and reports errors instead of generating invalid C++ code.
- The ASN.1 compiler may generate additional functions for the equivalent C++ classes of ASN.1 data types that allow the encoding and decoding of data values according to the Basic Encoding Rules (BER) of ASN.1.
- The TTCN–3 compiler has a new feature that may generate additional functions for TTCN–3 data types for direct (RAW) encoding/decoding of messages. This encoding scheme can be efficiently used for protocols that define the encoding of its PDUs in table-based format. The encoding rules shall be specified in special with attributes of the data types.
- The TTCN–3 compiler and runtime environment provides full support for the use of altsteps and dynamic defaults as specified in the (Edition 2 of TTCN–3 standard). Moreover, for backward compatibility, the obsolete named alts can also be used, even in combination with altsteps and defaults.
- The internal handling of TTCN–3 string types (bitstring, octetstring, charstring) has been improved. The runtime environment can copy string values without memory allocation, which may result in 50% performance improvement in some cases. The Test Port API for these types did not change.
- We have a comprehensive regression test suite for the tool itself. It covers almost all basic and user-defined types, built-in operators, template and behavior constructs of the TTCN–3 language. The tests are run before each release to minimize the remaining bugs.
- Lots of minor improvements and bug fixes.
- The tool is no longer called prototype. Quick help to achieve full backward compatibility with version 1.1. For the meaning of these switches please refer to the respective sections of this document.
- Use the `-u` and `-r` flags for the TTCN–3 compiler.
- Use the `-s` flag for the logformat utility.
- Ignore all warnings of the compiler that refer to obsolete TTCN–3 language elements.

3.43. Version 1.1

Version 1.1 has the following new features:

- Support of parallel test execution. Full support of TTCN–3 create, start, stop, running and done

operations.

- Support of distributed test execution, which means scalability. Automatic load balancing between the participating computers.
- Platform interoperability, that is, test components running on any of supported platforms can communicate with each other.
- The total number of parallel test components can be safely increased up to 1000, which enables performance (load) testing with the Test Executor.
- Internal communication between TTCN–3 test components is supported in a transparent way. TTCN–3 `connect`, `disconnect`, `map`, `unmap`, `send (…) to` and `receive (…) from` operations are also fully supported.
- Extended Test Port interface.
- Enhanced command line syntax and functionality of the compiler.
- Many bug fixes.
- Improved User Documentation. For more details, please see the next chapters.

[1] For example, the parser may get confused after a missing opening or closing bracket and ignore the rest of input module.

Chapter 4. Version history

4.1. Version 7/CAX 105 7730 R2A

Release date: 27th of November 2020

This release contains some new features and bugfixes.

New Features:

- Changes in Titan product code(Bug 565875)
- Changes in licensing(relevant for internal users only):
  - License ordering/extension page moved
  - License ordering/extension API to automate license renewal
- Changes in shipping libedit (Bug 565893)
- Titan image added to ADP marketplace(relevant for internal users only)
- Relaxing the C/C++ compiler version check for Titan 7.1.1 and upwards (Bug 564194)
- Object-oriented language features (Bug 563718)
- Allow unsafe universal charstring to charstring conversion (Bug 564585)
- Support for @default in the compiler and in the xsd2ttcn converter (Bugs 564919, 564920)
- Java-based Main Controller implemented
- Java-based JSON codec implemented

4.2. Version CRL 113 200/7 R1A

Release date: 29th of May 2020

This release contains major new features, backward incompatible changes and some bugfixes.

With this release we have also upgraded our license from Eclipse Plugin License 1.0 to Eclipse Plugin License 2.0.

New Features:

- The C side of the toolset will now support the Object Oriented Extension of the TTCN-3 standard (with some limitations).
- In the Designer plugin we have parallelized several algorithms resulting in 3-4* faster execution:
  - The semantic checking of modules inside the same project.
  - Projects in a project hierarchy, not dependent on each other, will not be able to be analyzed in parallel.
  - The Code Smell checkers of Titanium will also run in parallel.

- On the Java side of the toolset, we have added support for JSON encoding/decoding (with some limitations).

Backward incompatible changes:

- 'omit' matching mechanism shouldn’t be allowed for union fields
- Compilation issue in the Java code generator when used with very large modules

And many bugfixes.

4.3. Version CRL 113 200/6 R6B

Release date: 29th of November 2019

This release has the following new features:

- Enhance performance of TIMER::get_min_expiration
- Legitimize compiler option '-F'
- Colorize compiler error/warning messages
- change the internal representation of octetstring from char[] to byte[] for efficiency
- CSN.1 L/H in the RAW codec
- change in how xsd files with NoTargetNamespace should be mapped to TTCN-3
- add support for structured type compatibility to the Java code generator
- add support for module parameter handling to the Java side runtime
- Bug fixes

4.4. Version CRL 113 200/6 R6A

Release date: 17th of May 2019

This release has the following new features:

- a Java codegeneration plug-in
- a Refactoring plug-in
- Printing symbolic version in 'compiler -v’
- Legitimized compiler option '-0’
- New command line option '-p' implemented for TTCN binaries (both in single mode and parallel mode), which lists all module parameters
- Support for real-time testing
- Implementation of map param/unmap param
- Extended 'default' JSON attribute for structured types
- Colorized compiler error/warning messages

- Bug fixes

4.5. Version CRL 113 200/6 R5A

Release date: 7th of December 2018

New features:

- License upgrade to EPL 2.0
- Documentation migrated to asciidoc
- Bugfixes for core, ASN.1, XML, RAW, OER codecs, XSD2TTCN
- New RAW coding instruction 'FORCEOMIT'
- New JSON attribute 'as map'
- Attribute 'text ... as ...' for JSON
- Adopt Makefilegen for the Debian packaging
- Support for real-time testing in TITAN
- New flag for xsd2ttcn

4.6. Version CRL 113 200/6 R4A

Release date: 31st of May 2018

New features:

- Implement verdict redirect for `done' statement
- str2float should handle special float values
- RT2 record equality
- string2ttcn to filter patterns of visible characters in octetstrings
- Syntax to bind a variant attribute to multiple encodings
- TITAN build on Alpine Linux
- new tpd tag `disableUserInformation`
- runs on scope reduction (Titanium)
- Add discarding option to `setstate` operation
- Notify user if port is not mapped in translation mode
- Implement reference to port in translation function
- Implement extendable sequence coding in OER
- TAG and CROSSTAG for JSON encoder

4.7. Version CRL 113 200/6 R3A

Release date: 17th of November, 2017

New features:

- new compiler options:
  - -e: enforce legacy handling of `encode` and `variant` attributes
  - -O: disable OER encoder/decoder functions
  - -D: disable user and time information generation in the generated files
- Support for multiple encodings
- Implement OER coder in TITAN (with the option to restrict generation of OER codecs)
- Implement OER negative testing
- Allowing to start functions with `out` and `inout` formal parameters
- Enable `out` parameters for behavior functions in the `start` operation support for dynamic erroneous attributes
- Allow translation ports to work as internal ports
- Allow sending and receiving during translation functions
- Flag to disable time and user information in the generated files
- Implement mtc and system clauses in testcase and altstep and functions
- Add runtime configuration setting for plain XML and JSON encodings
- Implement `json2cbor` and `cbor2json`
- Implement `json2bson` and `bson2json`
- JSON enc/dec: encoding enumerated values in number form
- Support `enableLegacyEncoding` in tpd
- Add the encoding legacy switch to tpd TEXT codec
- Add the encoding legacy switch to makefilegen
- Add support for NULL terminated string in RAW
- RAW: add offset option to `LENGTHTO` attribute
- RAW: Support also `⋯ bits` syntax in variant attributes

4.8. Version CRL 113 200/6 R2A

Release date: 26th of May, 2017

New features:

- new compiler options:

`-J`: Compiler (and `xsd2ttcn`, makefilegen) option to read input files list from a text file

`-N`: ignore UNTAGGED encoding instruction on top level unions (legacy behavior)

- support of encvalue/decvalue for ASN.1 types
- support for implicit call of PER codec external functions
- implemented: ports with translation capability
- support for concatenation of templates
- implemented 'any from' clause and index redirects with the use of the @index modifier (see standard, chapters 21-23)
- support for dynamic erroneous attributes
- implemented @fuzzy support
- support for external functions for decmatch and @decoded
- no support of Solaris binaries from this release of Titan (older versions of course will continue to support Solaris)
- makefilegen more restrictive on name attribute of the referenced project
- makefilegen: remove generated headers dependency from all `.c` `.cc` files

(This will revert the following bugs:Bug 499963 - The generated Makefile does not make full build when `-j` switch is present ; Bug 512688 - makefilegen: Incorrect `.c` and `.cc` compiling rule)

- XER: allow anytype to be xer enc/decodable
- JSON `as value` attribute extended for records/sets with one field and for the anytype
- **make archive** button in Eclipse
- support for `make port` command in Eclipse
- plug-ins upgraded to Jung 2.1

4.9. Version CRL 113 200/6 R1A

Release date: 18th of November, 2016

New features:

- support for `mctr reconf` command
- command line debugger
- advanced code splitting
- makefilegen capability to handle `.xsd` files
- makefilegen and compiler to handle file lists in files(`compiler -J` file or `makefilegen -J` file)
- new compiler switch for decreasing variant errorlevel from error to warning
- LTTng logger plug-in
- encvalue/decvalue for ASN.1 types
- Titan build for ARM/Raspberry Pi

- decmatch and @decoded
- istemplatekind
- select union
- @nocase
- Partial @deterministic support
- Storing parts of received messages

Incompatibilities:

- warning changed to error when '*'is used for mandatory elements
- infinity/NaN not allowed at timer start
- receive handling changed (receive(*) and receive(?) not allowed or restricted)
- [UPDATE 3rd February, 2021] module parameter references (initializing module parameters with the values of other module parameters in the config file) only allowed in Runtime2 (see artf789088)

4.10. Version CRL 113 200/5 R5A

Release date: 26th of May, 2016

New features:

- type substitutionGroup support
- allow using specific encode attribute strings to identify encode functions
- `ttcn2json`: extra keyword for restricted "as value" unions
- makefilegen shall generate `-Y` if tpd orders it
- user Debug classes
- negative testing with JSON encoder
- new compiler switch for decreasing variant errorlevel from error to warning
- makefilegen supports commenting out OPENSSL_DIR based on tpd setting
- activate emergency logging when a test fails
- makefilegen `-I` option
- RAW encoder for universal character string
- ISO 10646-conformant unicode syntaxes
- new internal functions: `encvalue2unichar/decvalue2unichar`,`any2unistr`,
- `make port` command
- `checkstate` port operation
- clang support in Titan
- Eclipse Designer: implement fast algorithm

- config parser/editor based on ANTLR 4
- negative and positive conformance tests covering core language part of the standard added
- new document: statement of compliance covering Core language part of the standard
- legacy switches:

-M: allow 'omit' in template value lists (legacy behavior) (artf692717)

-B: allow selected union field to be unbound (legacy behavior) (artf717563)

4.11. Version CRL 113 200/5 R4A

Release date: 13th of November, 2015

New features:

- Refactored xsd2ttcn converter
- Eclipse plug-ins migrated from ANTLR 2 to ANTLR 4.
- 60 Eclipse plug-in related TRs and CRs implemented.
- Function calls with subreferences (artf550360)
- Template(present) accepts complement matching (artf564824)
- Integer to enumerated (artf590888)
- Support for IntX in RAW (artf607782)
- Module parameters can be initialized with module parameters (artf618367)
- Improved logformat to pretty-print XML and JSON

4.12. Version CRL 113 200/5 R3A

Release date: 22nd of May, 2015

New features:

- TEXT codec to support universal character string (UTF-8).
- New Junit Logger plugin with extended logging.
- First version of the coverage/profiler tool.
- Stack trace displayed in case of segmentation fault or `abort()`.
- Allow component and default types in module parameters.

4.13. Version CRL 113 200/5 R2A

Tentative release date: 19th of March, 2015

New features:

- `Makefilegen -Z` option: Faster than the previous recursive linking method , support for dynamic linking, improved make archive
- `Makefilegen -H` option: support for partial build of hierarchical *.tpd structures.
- `Ttcn2json` improved ASN.1 handling, including parameterized types

4.14. Version CRL 113 200/5 R1A

Tentative release date: 9th of January, 2015

New features:

- New JSON codec variants.
- TTCN-3 type to JSON schema converter compiler option introduced.
- Macro redefinition functionality for TITAN TTCN-3 Test Executor in the `[DEFINE]` section of the `.cfg` file.
- Nested concatenation operator `&=` in the `[MODULE_PARAMETERS]` section of the `.cfg` file.
- A number of deprecated compiler options (`-E`, `-n`, `-N`, `-B`) removed.
- Correction for newer openssl packages that break Titan license validation.

IMPORTANT	Titan releases previous to CRL 113 200/5 R1A will not work if openssl is upgraded beyond the critical level of release; the exact level depends on the Linux platform and version.

4.15. Version CRL 113 200/4 R2A

Released on the 4th of July, 2014

New features:

- JSON encoding support.
- Support for various universal character string encodings (UTF-8, UTF-16, UTF-32).
- Built-in support for base64 encodings.
- Java executor API for Titan.
- Eclipse plug-in improvements.
- Configurable timestamp in console.
- Improved behavior in port congestion situations.
- Superfluous circular warnings for ASN.1 disabled.
- TEXT encoder debug logging.

4.16. Version CRL 113 200/4 R1A

Released on Jan. 10, 2014

New features:

- Catching Dynamic Test case errors – Adds the ability to survive DTEs in TTCN-3 code, for instance in case of long running load tests. Very similar to exception handling used in other languages.
- Lazy Parameter Evaluation – In formal parameters can be defined to be subject of lazy evaluation: the expression used as actual parameter shall be evaluated only when the formal parameter is used (not at the function call); the evaluation is only done once.
- Titanium – new Eclipse plugin, a code quality analysis prototype for advanced users, available upon request.
- Usage statistics - Titan compiler, runtime and Titan Eclipse plug-in usages are collected for statistical purposes.
- Change of default error behavior for XML encoding from 'Warning' to 'Error' to align with the other Titan encoders.
- Template Module Parameters - TTCN-3 language extension, module parameters can be both values (standard) and templates (non-standard).
- `Ttcn2string()` predefined function - returns the parameter's value in TTCN-3 string representation. `String2ttcn()` predefined function - `Ttcn2string()` predefined function contrariwise.

4.17. Version CRL 113 200/3 R2A

Released on Jul. 5, 2013

New features:

- Support for distributed build using hierarchical Makefiles with new `ttcn3_makefilegen` command line options (`-r`, `-F`).
- New makefile target "library" is implemented. The pre-compiled objects can be collected to a library archive file (.a or .so), useful when the project hierarchy has rarely changing parts.
- Extended _.tpd file handling in makefilegen was introduced. `ttcn3_makefilegen` processes the MakefileSettings part of the _.tpd files. Benefits: *.tpd files extracted/created with Eclipse can be used in command line and usage of makefilepatch scripts can be hugely reduced or even eliminated.
- `ORDERED_INCLUDE` in configuration files is implemented; the includes will be strictly ordered.
- Clean-up after unsuccessful makefilegen execution, symlinks are now generated only if no errors were found during *.tpd file processing
- _.tpd file validation with `ttcn3_makefilegen`: the _.tpd file is validated with a schema that now is part of TITAN (file `$(TTCN3_DIR)/etc/xsd/TPD.xsd`); validation errors will prevent makefile generation.

- `Makefilegen`: override the working directory in _.tpd file: the working directory of top level project comes from top level _.tpd file by default; when using the –D switch the working directory will be the current directory.
- `Makefilegen` support for OSS Nokalva ASN.1 compiler is implemented. Makefile generation from *.tpd file enables OSS Nokalva support without custom makefilepatch scripts
- Integration of DPMG (Diameter Protocol Module Generator) into the TITAN build system.
- Improved *.tpd file related documentation.
- Reduced nr of supported gcc versions. Supported versions are: 3.4.6 – 4.7.2
- Changes in supported platforms: Solaris versions from 5.10 are supported; Cygwin versions from 1.7 are supported. Earlier Solaris and Cygwin versions are not supported
- Titan Eclipse plugins support Eclipse versions from Eclipse 3.7.2 to Eclipse 4.2
- Java 1.6 is the minimum requirement
- A fourth Eclipse plug-in, Titanium, is released as a prototype. Update and maintenance of Titanium will be the responsibility of the Titanium project until further notice. For details pls. see https://ericoll2.internal.ericsson.com/sites/DUCI_SW_Technology/Titanium/default.aspx

4.18. Version CRL 113 200/3 R1A

Released on Jan. 18, 2013

New features:

- Interface implemented for the TestStatistics tool
- All from in value list, subset, superset and permutation supported
- Embedded macro references in the `[DEFINE]` section - runtime (support in command line)
- Structured macro definitions in the `[DEFINE]` section - runtime (support in command line)
- Embedding TTCN-3 functions

4.19. Version CRL 113 200/2 R2A

Released on Aug. 31, 2012

New features:

XML encoding is now supported for the hexstring and verdicttype TTCN-3 types

Transparent functions were introduced to allow easier identification of failing tests in case of SourceInfo := Single.

4.20. Version CRL 113 200/2 R1A

Released on Jun. 27, 2012

New features:

- The Titan Eclipse Designer's support for preprocessed TTCN-3 files has been improved.
- The performance of TEXT decoding has been improved.
- A logger plugin (`JUnitLogger`) is now delivered with Titan. It outputs XML files in the same format as JUnit. Using this logger plugin allows integrating of Titan with the Jenkins (Hudson) continuous integration tool.
- To allow `JUnitLogger` to receive the necessary information, the Titan Logger API has been slightly changed. Existing logger plugins will need to be rebuilt.
- In response to a TR (HP88760), the C++ interface of the OBJID class has been changed. The type of the elements in the internal storage of the OBJID class is now specified with a `typedef`, `objid_component`. Code which uses the indexing operators or directly accesses the element storage will need to be rewritten. It is a backward incompatible change and it affects users of the SNMP test port. A new version of the SNMP test port was released (CNL 113 344 R4B) compatible with the new Titan.
- `ttcn3_makefilegen` has a new flag `-P`, which prints out the list of files found in a given TPD recursively relative to a given directory.
- TTCN-3 level code coverage was implemented.
- Text hover for T3Doc in Eclipse was implemented.
- `mctr_gui`, `ttcn3_logbrowser`, `ctags`, Nedit, XEmacs support is part of the Titan package again.

4.21. Version CRL 113 200/1 R10A

Released on Apr. 13, 2012

New features

- Renaming refactoring was implemented in Titan Eclipse Designer. This feature provides TTCN-3 scope-aware renaming of declarations.
- Selection highlighting was implemented in Titan Eclipse Designer. When a variable name or function name or keyword is selected in the code, all the occurrences of the selected variable name or function name or keyword will be highlighted in the same file.
- Performance of `log2str()` was improved.
- Implicit omit support for module parameters was implemented.
- Append operation (`&=`) for list types in configuration files was implemented.
- Support of executing testcases with default parameters from command line and configuration file was added.
- Improved error recovery for the compiler. E.g. it can now stop on the first syntactic error and skip the semantic analysis.

Fixed bugs

- **HP53582** Calling `Remove_Fd_All_Handlers` after `Remove_Fd_Read_Handler` causes error
- **HP57968** Designer: Running the compiled test without parameters can have unexpected effect

- **HP49044** Error window popup on any `Exclude/Include` operation in the workspace
- **HP70610** Reference search: does not find references in for loop header part
- **HP70600** Reference search: does not find local variables inside alt guard blocks
- **HP63161** Designer: `IllegalArgumentException` when creating TTCN3 files
- **HP40284** On-the-fly checker does not accept timer as log argument
- **HP55541** Single mode launcher runs in an arbitrary directory
- **HP55521** Eclipse Single Mode Launcher ignores config file
- **HP43578** Titan: faulty warning printout during compilation, "statement not reachable"
- **HP43572** Titan: fail to evaluate alt-statement (snapshot) correctly
- **HP22848** Titan compiler 1.8pl7 fails on Solaris10u10 with a "Too many files open "message.
- **HP38572** modulepar description in the Titan help is outdated, and not complete
- **HP39882** On-the fly checker: second imported definition of the same type is not recognized/stored
- **HP39843** on-the-fly checker: faulty transitive behavior of import
- **HP19155** UserGuide does not contain information for `-lutil` flag dependency in Makefile
- **HP38965** On-the-fly semantic checker doesn't accept `sizeof(X)` where X type is record of sth

4.22. Version CRL 113 200/1 R9B

Released on Jan. 24, 2012

Fixed bugs

- HP36538 was fixed. Incorrect handling of the := assignment in the `[DEFINE]` section of configuration files.

4.23. Version CRL 113 200/1 R9A

Released on Dec. 19, 2011

New features

- With the release we have decided to change from the proprietary Titan versioning scheme, to the one used by Ericsson. From now on it will be much easier to decide if a new version is forward, backward compatible with a previous version. The versioning is also supported in the attributes of the modules, with some limitations. We only accept version numbers in 3 formats: R9A, CRL 113 200 R9A and CRL 113 200/1 R9A.
- With this release we removed all QT based GUI parts (`mctr_gui`) and `ctags` from the official Titan releases, as they have been in maintenance phase for the last year. NEdit and XEmacs parts are still available as downloadable components from our download pages.
- The import of imports feature declared in the newest TTCN-3 standard was implemented. This way it is now possible to recursively import import statements from other modules.

- IPv6 support for Titan's internal communication was implemented. This way Titan is now able to function properly when the MC and PTCs are located on an IPv6 network.
- The makefilegen tool in the command line package is now able to generate Makefiles from the information stored in .Tpd project descriptor files.
- It is now possible to find all reference pointing to a given declaration inside eclipse. Finding all references to a definition was implemented as a new kind of search in the Eclipse platform.
- The Executor plug-in will now be able to automatically merge the generated log files after execution.

4.24. Version 1.8.pl7

Released on Oct. 10, 2011

New features

- The handling of XSD minOccurs and maxOccurs was updated to follow the upcoming version of the standard (4.3.2) with regards to the handling of optional alternatives of <choice> elements.
- The `testcase.stop` operation is now supported, allowing for the users to stop the execution of the actual testcase raising a dynamic testcase error with a custom explanation text.
- The `ispresent` predefined function was extended to operate on all structured types and fields as described in the 4.3.2 version of the TTCN-3 standard.
- We have re-implemented the `isbound` predefined function in way that is much more performance efficient than the previous one released.
- The `encode_utf8` function of our universal charstring class became part of our public API, so it can now be safely used from C/C++ codes as well.
- The indexing of string templates became supported.
- The main features of the LogViewer eclipse feature can no be accessed from the Project Explorer too, so it is no longer required to switch to its custom navigator.
- It is now possible to configure the Executor feature and eclipse executed "launch configurations" to automatically merge the log files that were generated during execution. For the case of several consecutive executions it is now possible to configure the system, to remove the previous log files before a new execution.

4.25. Version 1.8.pl6

Released on Maj. 30, 2011

New Features

- With the new negative testing feature it is possible to generate invalid messages, and to send them to the SUT, to observe its reaction. For example mandatory fields can be left out, new data fields appended, value constraints can be violated.
- Emergency logging allows for the users to define logging behavior for normal and emergency situations. For example one could completely turn off logging for the normal case, while still

receiving all needed logs in case of an error.

- The performance of the LogViewer eclipse plug-in was enhanced, so that now it no longer needs to store in memory all data of the log files to be able to display its content, neither in the table based representation nor in the Message Sequence Chart based representation.
- The LogViewer was also extended with support for searching and filtering in Titan generated lo files. Naturally this was also done in a way that blends naturally to the platform, so that users will not have to learn new ways of working.

4.26. Version 1.8.pl5

Released on Dec. 17, 2010

New Features

- The TITAN logging architecture has been re-designed to support dynamic configuration and logger plug-ins. Currently only the legacy logger plug-in is supported, which creates backward compatible log files.
- Titan is no longer depending on the external Readline package. It has been replaced with Editline, which is now compiled into the delivered packages.
- A new feature for importing and converting MCTR_GUI project to Eclipse format was added.
- A new project description format has been created to support exporting and importing the data of Titan projects in eclipse into a single file.
- The LogViewer eclipse plug-in was enhanced to work on larger files, with less resource consumption. Also it is now much better integrated with the rest of the toolset.

Backward incompatibilities

TR number HM60511 raised our attention to the fact that according to the newest standard it is disallowed to index inside a matching different from "?" (See section 15.6.3 of the standard). This might make existing codes cause dynamic testcase errors at runtime.

4.27. Version 1.8.pl4

Released on Aug. 13, 2010

New Features

- Unbound checking has been completely finished according to the standard.
- Huge speed increase and reduced memory usage was achieved in the Designer when the incremental parsing is turned on. Thanks to research efforts done in this field.

4.28. Version 1.8.pl3

Release on July. 02, 2010

New Features

- Subtype checking for ASN.1 subtype constructions was implemented for the command line.
- A feature introduced into the 4.1.2 version of the TTCN-3 standard became supported, which allows the declaration and usage of not completely initialized record and record of values as long as the un-initialized element is not referenced directly.
- The `-v` flag of the generated ETS was enhanced to print the version information attached to the modules it was compiled from.
- Single mode execution was enhanced with automatic control part execution in case there is only one control part in the whole testsuite compiled into the ETS. In this case it is not necessary to provide parameters to the ETS when executed.
- Added support for the exclusive range bounds feature of the TTCN-3 standard.
- The name of the testcase will be displayed in the name of the log files of the MTC and HC if configured to be shown. Previously it was only displayed in the PTC's logs.
- The execution of external script actions will always be logged in the MC, both before the execution and after the execution of the script, to indicate the range where execution has spent its time outside the TITAN generated code.
- The `ttcn3_start` script was extended to accept as an optional parameter the ip address it should start its communication on. This is useful when the computer running the tests is connected to several networks at the same time.
- We have started to re-work the logging of the runtime. At this time this should not have any effect noticeable for the users (Other than taking the name "Titan_Logger_Api").
- The subtype checking done on TTCN-3 modules in the previous release of the command line tools, was introduced into the Designer plug-in.
- When a new TITAN project is created as the last step of the wizard it will present the properties page of the new project.
- Launch shortcuts became supported by the Executor plug-in. This enables the user to create and initialize a new or reuse an old Launch Configuration simply by selecting a TITAN project or a configuration file for execution. The new launch configuration will be created and initialized to default values based on the data found on the project (if the Designer is also installed at the same time) and automatically launch the execution.
- It is now possible the exclude certain resources from the build by providing a global list of regular expression, that will be matched on the file names. If any of the expression matches on the name of a file, that file will be excluded from the build.
- It is also possible to configure the Project Explorer view to exclude the excluded resources and the working directory from its shown elements.
- In order to make it more apparent, why a given resource is not part of the build of the project, the exclusion decoration has been enhanced to describe the reason of exclusion.
- It is now possible to configure the Designer plug-in to do naming convention checks on the source code. The conventions can be configured globally, on project level and even on folder level if needed.
- The way of handling the path of the working directory, the generated executable and the makefile updater script was reworked so, that now it is possible to use environmental variables

and Eclipse path variables in them too.

- As part of the previous item if the working directory is not present when the build is started, it will be created automatically.
- The Designer was enhanced to collect information about the compiler being configured as the actual build environment. If this setting is changed it will offer to rebuild all of the projects.
- The internal `Makefile` generator of the Designer was enhanced to support building a project without using symbolic links.
- It now supported to have several build configurations defined for each project. This way if one has a "debug" and a "release" configuration, one will be able to switch between the sets of build settings configured for each simply with a few clicks.
- The on-the-fly analysis of the Designer was extended to support delayed semantic checking. When this option is turned on, the on-the-fly semantic analysis will be only invoked when the users saves the file he was working on. While he is editing it only the syntactic checks will run. This mode enhances the performance of the tool, when one is editing framework libraries. However as the semantic database is not updated until the semantic analyzer is run, so will the code completion and other higher level functions also work with somewhat outdated data until the next `save` operation.
- The methods for building a TITAN project were introduced. In the first form the user is able to invoke the build process of Eclipse on a project from the command line, without activating any user interface elements. This mode will build the project on the exact same way it is done when the user is calling it from Eclipse. In the second form an xml file generated with all the data that might be needed to call the TITAN provided makefile generator. Using this form the user is able to create his own scripts, allowing to configure his build process in much finer detail.

4.29. Version 1.8.pl2

Released on Jan. 29, 2010

New Features

- The base of the TTCN-3 standard used to describe the features and limitations of TITAN was changed from version v3.1.1 to v4.1.1
- The checking of subtypes in TTCN-3 was improved considerably.
- The semantic checking done by the on-the-fly analyzer in the Designer plug-in was enhanced to be on the same or higher level than present in the command line. A few checks are still missing as a limitation, but if the configurable checks are set several high level bugs/maintenance problems can be detected.
- A version checking mechanism was implemented, where TTCN-3 modules can have version numbers and place version requirements on imported modules, or the TITAN that is used to compile the actual module. Please also note, that as this feature introduces new syntax, earlier TITAN version will report an error for it.
- Support for dynamic linking was introduced into the build system. As in case of incremental modifications, sometimes most of the build time is spent with linking the object files to the final executable, eliminating this step can enhance build times in these cases. However this also

means that the dynamic libraries must be transported together with the executable, as it will no longer work in a standalone manner.

- Dependency checking was enhanced in the build system. If using the new way, dependencies will be refreshed only for those modules that have changed, plus the dependencies on gcc are not tracked.
- At build time the compiler can to split the generated code based on the types present in modules. When using the option "type", TITAN will create separate source files for the implementation code of the following types (for each module): sequence, sequence of, set, set of, union. In this case a common header file and a source file holding everything else will also be created. The amount of the generated files increases on this way, but as each of them is smaller the C++ compiler can compile them easier. As there are more files, the build process can run much more efficiently in parallel mode.
- In the Designer plug-in the behavior of the content assistant can be configured by the user. Sorting of the proposals can be configured to be either alphabetical or relevance based. It is also possible to set the common prefixes of proposals, or in the case there was only 1 proposal found the whole proposal should be inserted automatically.
- The automatic insertion of closing apostrophes can also be configured.
- A new action was added to the TITAN actions toolbar, where the xsd2ttcn converter can be invoked on the selected files.
- The syntactic analysis of files was enhanced to become parallel, allowing several times faster operation on machines having several computational cores. For example a dual core processor (commonly present nowadays) will be able to parse two files in parallel.
- The show view menu of the plug-in's default perspectives was extended with links to views commonly present in the perspectives, to help faster navigation.
- In the internal makefile generator the `OPENSSL_DIR` and the `XMLDIR linker` search paths can be disabled, in case the users wish to set their own libraries.
- The reporting of syntax errors in extension attributes became configurable. According to the standard if TITAN is not able to perfectly understand an extension attribute, it should assume that it was meant for a different tool instead of reporting errors, but in this case typos could not be reported to the user.
- In the build process if the working directory does not exist when the build is started, but is set to be contained directly in the root of the project, it will be created automatically. And after the build has finished its contents will always be refreshed automatically, to represent the contents of the actual file system.
- Also in the build process, just before executing the external command the `derived` flag of the working directory will be set automatically (users could set this by hand till now). Setting this flag should mean for other plug-ins, that the contents of this folder should be treated specially, for example they will be left out of search results, and version handling plug-in should also ignore them. This together with the previous feature allows better interoperability with version handling systems, as in this case the working directory no longer needs to be handled by the version handling system in most of the cases.

Fixed bugs

Several bugs found both in the xsd2ttcn converter and in the XML encoder/decoder were corrected.

4.30. Version 1.8.pl1

Released on Sept 11, 2009

New Features

- Added support for the module interface feature of the TTCN-3 standard (version 4.1). Allowing for the users to assign visibility attributes to definitions.
- Added the `testcasename()` predefined function, which returns the name of the actual testcase or an empty character string.
- The `sizeoftype()` predefined function was removed in accordance with the new TTCN-3 standard.
- Introduced the **FILE** and **BFILE** pre-processor macros, which are replaced with the canonical path of the file, and the name of the file respectively.
- The meaning of the **SCOPE** macro is changed to comply with how it has appeared in the standard. In the new operation it will be replaced with the name of the lowest named basic scope unit in which the macro is used.

Fixed bugs

In the Designer plug-in the `extends` extension attribute was parsed incorrectly.

4.31. Version 1.8.pl0

Released on Jun 12, 2009

New Features

- Support for XML encoding and decoding is introduced, together with a new command line tool that converts XSD files into TTCN-3 modules.
- The TTCN-3 type Anytype is now supported with some restrictions (see section 4.2 of the Programmer Reference Guide).
- A new runtime was introduced, that requires much less code to be generated and compiled at the cost of minor decrease in runtime performance. The original runtime is advised to be used in load test scenarios (for this it is called load test runtime), while the new runtime is advised to be used in function test scenarios (for this it is called function test runtime).
- The internal handling of extension attributes was redesigned. The original analysis of these attributes was dependent on the location where they were found (so the same extension was accepted for a function but rejected for a type). This behavior was changed to accept all extension attributes, and only report an error if the attribute is located at the correct place, but contains some semantic errors in itself.
- Several predefined functions were extended to be able to accept templates as parameters (`encode`, `replace`, `substr`).

- Index assignment notation became supported in base templates
- With the addition of the **SCOPE** macro TITAN will now support all TTCN-3 macros defined by the upcoming TTCN-3 standard (version 3.4)
- The speed with which PTC were created was enhanced. Compared to 1.7.pl3 there was a noticeable slowdown in 1.7.pl4. With this improvement PTC should be created faster than in 1.7.pl3.
- All operations are now supported for big integers too.
- The `enum2int` predefined function was implemented
- The `setverdict` predefined function was extended with an optional `charstring` parameter where the users can specify the reason of setting the verdict.
- The implicit omit attribute feature of TTCN-3 was implemented
- A new option was introduced to the compiler to emulate more precisely the warning/error message format of gcc, so to make it integrate with eclipse much better.
- Concatenation of patterns became supported, and from now on patterns can reference templates too.
- The encode, decode predefined functions were implemented.
- `Inout` parameters became supported when functions are started.
- The automatic postfixing of identifiers was introduced, to be able to refer to assignment in ASN.1 modules which have a name that is a keyword in the TTCN-3 language.
- We added support for several features that operate on list types (set of, record of and arrays) including: `concatenation`, `rotation`, `substr`, `replace` and `lengthof`.
- Both Eclipse plug-ins were enhanced with the ability to format and merge log files, in the form of two new actions available in the TITAN menu.
- The executor plug-in was extended to report an error if an executable was set for a launch configuration that is not able to use it (for example an executable compiled for single mode execution can not be executed in parallel mode).
- It is now possible to set, that when the external TITAN action actions are executed on a set of file, they should not process those that are excluded, or are inside excluded folders.
- It became possible to configure what should happen to the markers reported by the compiler, once an on-the-fly analization was executed.
- It is also possible to handle the on-the-fly reported error markers as fatal for build, meaning that as long as the on-the-fly analyzer is reporting an error on a project it will automatically fail the build process. Running the build in such cases would most probably also end up reporting the very same error, but would take a long time to do this.
- It is possible to configure the severity with which the unused function return value problem should be reported.
- The "go to matching bracket" feature was implemented.
- The Designer plug-in was enhanced to detect the number of processing resources possible to use in a build, and as such is able to drive the build process to use several parallel threads. This should result in the decrease of build times, for user who have not yet manually configured

their system to do so.

- Introduced the "Treat `.ttcnpp` files as `.ttcn`" feature. If this is enabled the on-the-fly analyzer will try to analyze `.ttcnpp` files as if they were ordinary TTCN-3 files. If the `.ttcnpp` files do not contain any pre-processing macros, but can not be renamed for external reasons, this feature will greatly enhance the user experience. If the files do contain pre-processing macros than enabling this feature will only mean a change of reported errors.
- The Designer plug-in is able to check the validity of the license file, to display the data contained within, and to warn the user a few days before the expiration of the license.
- Enhanced the code quality checks to detect unused definitions and assignments, both on module level and in local scopes. These two scopes has to be separated as unused local definitions always indicate an error, while unused module level definitions might be completely valid in library modules.
- The on-the-fly semantic checker of the Designer plug-in was enhanced considerably.
- The Designer plug-in was enhanced with the ability to incrementally parse TTCN-3 files. This means that after the first time there should be no need to syntactically re-analyze the whole file, but the tool will be able to decrease the amount of data to be re-analyzed to about a few lines. This will not only decrease the time required to re-analyze a project from a few seconds to a few times 10-2 seconds, but will also stop the outline from collapsing after each change in the file.
- The Designer plug-in was extended with an internal makefile generator which uses the data collected by the on-the-fly analyzer. Using this makefiles can not only be generated faster, but the way the makefile is generated can be configured very precisely for each project. When used properly makefiles generated this way should not need to be changes later with makefile updater scripts.
- The on-the-fly analyzer was enhanced to adapt to changes in the file system. So if a new file is added to the project it will be analyzed automatically (earlier a file had to be opened in a supported editor).

Fixed bugs

- There was a slowdown in component creation.
- When the `Log match` operation was used, with the matching hints option set to compact, and the mismatch between the value and the template was contained somewhere within a union type, there was actually no information logged by the operation.
- Some special big integers could be encoded or decoded incorrectly in internal communication.
- The `install_handler` function did not handle correctly the case when a user closed a file already having a handler, then opened a file with the very same file descriptor, and tried to install a new handler on it.

4.32. Version 1.7.pl4

Released on October 03, 2008.

New features

- Template restrictions from the coming TTCN-3 standard (version 3.3.1) was implemented, allowing a finer specification of templates.
- A new predefined function called `log2str` was introduced. This function works like the original log function, accepting any number of parameters of any type. But the character string created with the concatenation of the parameters is not logged in a file, but returned as a charstring.
- The `replace` predefined function was implemented for all string types.
- Two new keywords from the coming TTCN-3 standard (version 3.3.1) were implemented : break and continue. Using these constructs it will be easier to create simple to understand loop sequences, as the loop condition can be simplified (INCOMPATIBLE).
- The connection handling on both the Main Controller and the Parallel Test Component side was enhanced with using an epoll based mechanism. On the Linux based platforms where this feature is available the users will be able to create as many connections as they want without the need to use a special build of TITAN. The overhead of using thousands of connections compared to using only a few will be almost non-measurable.
- The testport API was also redesigned to support this new feature gained by using the epoll functionality. This way the above mentioned benefits will also be present for the testport writers. For backward compatibility reasons the old interface is kept, meaning that existing testports does not need to be changed. However, using the old interface the testports will not be able to use the new possibility to its fullest.
- The logging of the `match` operation was made configurable through the `MatchingHints` logging option. If it is set in "Compact" mode (which is the default) the log record will be only a few lines long, instead of a few hundred lines long. In fact if there is only one field mismatching than the log will contain 1 line regardless of the size and structure of the value and template compared.

New features added to the Eclipse plug-ins

- The semantic data stored by the on-the-fly toolset about TTCN-3 files was increased to be about at the same level as the compiler is. Minor items like storing the 'with attributes' is missing, but other than that every structure is in place. This change was used as base for other features, and will serve as the base of the whole infrastructure we are going to build.
- The on-the-fly semantic checker was enhanced considerably thanks to the increased amount of data available. This allows the fast detections of lost of much more semantic errors, reducing the number of builds the users have to have dramatically. Because full semantic checking was not an aim of this project, and storing data coming from ASN.1 modules is not yet fully supported, the on-the-fly semantic checker can not be complete. The missing parts include areas like the checking of actual parameters, or checking the existence of return statements.
- We have implemented a few code quality checks in the on-the-fly semantic checker, which can detect a few inefficient structures: loops whose entry condition never evaluates to true, value shifting or rotation that actually does not change the value, etc...
- Seeing that now there are projects containing hundreds of modules, we implemented a heuristical check for superfluous import statements. In several cases import relations were declared between modules that did not actually import any definition from each other. This only complicated the understanding of the relations between modules, and put an unnecessary constraint on the incremental build system. This function is not a full functionality, as the on-the-fly semantic check is not complete, it can also not be complete. For this reason the reported

severity of such problems was made to be user configurable (it can be set to be an error, or warning, but can also be turned off).

- Even though we have increased the amount of data stored in the memory, we have managed to decrease the overall memory consumption. This is mainly the result of completing the on-the-fly structure for the TTCN-3 modules, as with the whole structure and the better semantic checker in hand we could already implement several optimalizations.
- The jump to definition was also implemented for configuration files. This way it is now possible to jump to definitions inside the configuration files, or to module parameters receiving value in the module parameters section.
- The standard outline view found in Eclipse is now supported for TTCN-3 and ASN.1 modules. This way the user can see an outline of the structure of his module to better understand it, or to find the declaration of definitions much faster. This outline view can not only be used to sort and filter the definitions in a way best suited for the user, but by clicking on an element displayed can be used to instantly navigate to the searched feature.
- An other long existing and wished for feature that we now started to support was what Eclipse calls "project references". In this feature the user can set the dependencies of projects inside Eclipse and from then on both the build processing and the on-the-fly checking of these projects will handle them automatically as dependent projects. This not only allows the partitioning of larger projects into smaller, more concise parts, but also allows to do this in a file system independent manner. For example a new project just existing on the users computer might depend on other projects stored in several different version control system around the world, as long as each project is set up to be working correctly in a standalone manner, they can be connected into much bigger project hierarchies.
- We have introduced two more build levels in the Designer plug-in (level 2.5 and 4.5) which use a heuristic algorithm to decide when the dependency relations of modules needs to be refreshed. Using this feature the users don’t need to choose between the safety of refreshing the dependency hierarchy, and the speed when not doing so. When all of the source code used in the module is handled by the on-the-fly analyzer, the dependency data will be tracked, and the slow external dependency update will only be called if needed. However if not all sources are handled by the on-the-fly analyzer, or the situation is not perfectly clear it will always decide to do the dependency update as otherwise the generated code might not compile correctly.
- We have also implemented a text hover functionality. When the user holds the mouse cursor over a definition for long enough, the information displayed about the definition in code completion, will be displayed in a hover box. This way to find out the type of a definition, the user only needs to hold the mouse above it for a short time, there is no need to actually jump to declaration of the definition.
- Since the on-the-fly toolset started to report syntactic and semantic errors, there was always the problem of different errors being reported. The compiler doing the full semantic checking was doing a much better check, but the on-the-fly toolset was working with the actual state of the file. This resulted in situations where the error marker of the compiler was already outdated, or when the tools had their error markers on single error (detected by both). This was now changed, by making the problems reported by the compiler "outdate" after the user has edited the file. This way the markers will still be there, so the user will be able to find other errors to correct, but the gray color of the outdated markings will indicate that the problem might already have been corrected.

- The `mctr_cli` based execution mode was extended to support automatic execution via tracking the state of the underlying `mctr_cli`, through the command line.
- All executor modes were extended to support the execution of control parts as members of test sets.
- Both the Designer and the Executor Eclipse plug-ins received a graphical refresh. All launch modes, definitions and other outline elements, invocable external actions received their very own distinct icons.

Fixed bugs

- The matching of a value containing the omit value, was not handled correctly when the template had a list or a complemented list in the position. The required functionality was not implemented in the generated code, but only the base library.
- When an interleave was embedded in another interleave the generated code was incorrect. If one branch of the embedded interleave was executed it was handled as if all branches of the embedded interleave would have been executed.
- Although TITAN allowed the referencing of global definitions without specifying the module name inside patterns, but not charstring fields of structured constants or using with the module name prefix.
- In certain situations, when a returning function had a too complex branching hierarchy implemented, sometimes the compiler was not reporting paths without a return statement as an error, but as a simple warning. This caused that even though there was no return statement, the code was compiled without problem, and when the execution of the function finished it returned with some memory garbage. This case, when the compiler noticed that something was wrong, but could not decide if it really was wrong or not, was promoted to an error level, to provide safe operation. This is not a backward compatible change, but well written source code, should not need any changes (INCOMPATIBLE).
- The values assigned to templates of signature types were not checked semantically, and so corrupted code was generated.
- In very complex, self-reflexive type structures the semantic checking of the compiler could mark the start function of startable functions as generated, without actually generating it. For this reason the generated code was sometimes erroneous, as the function call could be generated, but the function itself was not.
- The `isvalue` predefined function was working incorrectly for array templates, as the specific functionality was not implemented, and so the general implementation included in the base libraries was executed.
- The code generated for timer array was very inefficient as the name of each timer was generated separately in the code. In case of a timer array containing 20 million timers this resulted in a so big generated source file, that gcc was not able to compile it.
- In the Executor Eclipse plug-in there was no error report if the command used to create the Host Controllers was erroneous. This was simply caused by the fact, that the output appearing in the console was only reported on the user interface one the Host Controller was started, as in this case it was not able to start the contents of the output reading stream were cleared too early.

- The `ttcn3_start` script had no error handling procedure if the error appeared right after trying to execute the `cmtc` command. Which in some cases caused it to keep waiting for the good results indefinitely long, instead of exiting.
- The self component reference was only usable in function which had a runs on clause. This was too restrictive as the standard allows such usage.
- When the system component was used in the connect operation the semantic checker reported a rather un-intuitive error message, which had to be rephrased.
- The `is_bound` function was not generated for some types when the usage of older naming conventions was specified by the user.
- In the Designer Eclipse plug-in the configuration of TITAN to use default values as option always only implemented in the main build system, but other external operations like the testport generation was not configurable with this option.
- In some case the On-the-fly parser of the Designer plug-in was reporting syntactic error for syntactically correct named parameter constructs as a result of an incorrect grammar rule.
- The compiler was not checking the compatibility of runs on components if the function with incompatible runs on component was called inside a log statement inside a function. This check was simply not implemented for function calls placed in log statements.
- Because of a minor bug the pattern #(,1) was not accepted directly in template patterns.
- Because of an error in the Executor Eclipse plug-in, in single mode execution when the input configuration file was syntactically erroneous it was not reported, and the execution was not stopped, but temporary configuration file was generated erroneously.
- TITAN, as a nice feature, was implicitly concatenating character strings which turned out to cause problem, as in case of list of strings, the missing of comma sign was not reported as a syntax error, but the list was created with less elements (as the string where the comma was missing were concatenated).
- We have found an interoperability problem related to the ClearCase Remote Client in the Designer Eclipse plug-in. As the problem was found to be on the side of the Remote Client an error report was sent to IBM, and a workaround was implemented in our plug-in.

4.33. Version 1.7.pl3

Released on March 10, 2008.

New features

- When calling a function, altstep or testcase it is now possible to provide the actual parameters in a different order than the formal parameters were defined. If the each actual parameter exactly qualify to which formal parameter they should be assigned to.
- Now it is possible to use array indices within assignment notations.
- A new flag `-d` was introduced for the compiler to enhance interoperability with other implementations of the ASN.1 standard. When this option is provided the compiler will handle fields of set and sequence types having a default value as if they were optional. This means that these fields will be omitted when encoded, and will not be expected at decode time.

- A new predefined function called `isvalue` was introduced. Using this feature it is now possible to check if a template can be converted to a value with the `valueof` operation or not. As calling `valueof` on a template which did not contain an exact value resulted in a dynamic testcase error.
- Concatenation of binary strings is now possible in the runtime configuration file.
- To further enhance the logging utilities it is now able to split huge logfiles at the time of generation based on options set by the user in the runtime configuration file.

New features added to the Eclipse plugins:

- An on-the-fly parser for runtime configuration files.
- A basic on-the-fly parser for ASN.1 .
- Low level semantic code analysis for TTCN-3 and ASN.1 modules.
- The "Jump to definition" and "Open declaration" features were enhanced to work in ASN.1 modules too. Now it is also possible to cross the borders between the 2 module kinds, allowing for the user to jump to a declaration in an ASN.1 module, from a TTCN-3 module where it is used.
- The runtime configuration file editor was enhanced to offer not only textual editing possibilities for the user, but also some graphical editing functionalites. The graphical pages of the configuration file editor were organized according the sections in the file format, trying to provide a clean separation for informations that are not directly related. Each graphical page was designed to simplify the most common operations, for example on the logging page the user can change the logging settings with simply selecting the categories they wish to be logged out, or deselecting the ones that should be left out.
- The icons of the different supported file formats, and callable command line operations were re-designed, to provide a much better user interface, where the users can find the oprations they wish to invoke simplier and faster.

Fixed bugs

When a constant universal charstring value was assigned to a charstring the compiler did the assignment with reporting any problems, however if there was a complex expression resulting in a universal charatring on the right side the compiler reported a semantic error. This inconsistent state was resolved by reporting a warning for the first case too. This is only done to give some time to the user to make the necessary changes before an error will be reported for that code, making it un-compilable.

4.34. Version 1.7.pl2

NOTE	This is was an intermediate release, required by the TitanSim project.

Released on November 30, 2007.

New features

- A new function and altstep reference type was introduced called "runs on references". This allows the reference touse resources defined by the runs on clause of the actual function or

altstep, when it is called using the apply statement.

4.35. Version 1.7.pl1

NOTE	This is not a released version, only a delivery, delivered on August 27, 2007.

New features

T* he log event subtypes were introduced, allowing finer log settings.

- Type mapping rule discard has been introduced in dual-faced ports, which allows conditional or unconditional dropping of messages while translating them between the external and internal interfaces.
- Automatically generated TTCN–3 external functions used for encoding and decoding have been enhanced: The functions generate debug printouts with event type DEBUG ENCDEC before and after invoking the codecs. The decoder functions report a warning if superfluous data remained in the buffer after successful decoding.
- The translation of TTCN–3 regular expressions has been significantly enhanced in the compiler and the run-time environment: The character sets are verified and duplicate members are reported. Support of quadruple notation has been added for character codes between \q_{*0,0,0,1*} and \q_{*0,0,0,255*}. The generated POSIX equivalent is optimized to be shorter and simpler.

NOTE	TTCN–3 regular expressions are used by the matching mechanism pattern in templates of type charstring, the arguments of predefined function `regexp()` and the attributes of TEXT encoding.

- Non-standard additional predefined function `unichar2char()` has been introduced.
- The run-time realization of TTCN–3 additional predefined functions has been enhanced. New polymorphic versions have been introduced to eliminate the conversion of arguments in C++. The error messages generated by these functions have been rephrased to make the reason of the failure easier to understand.
- Utility `ttcn3 logformat` supports the indentation depth of zero. Option `-i 0` eliminates the previous indentation made in the file so that each log entry is printed in one line.
- The semantic analyzer of the compiler checks the TTCN–3 and ASN.1 modules in bottom-up order, which means the analysis of a module is started only after the checking of all imported modules is completed (except in case of circular imports). This new checking strategy results in shorter and more straightforward error messages because the irrelevant context information is not printed anymore. The original algorithm processed the modules in the same order as they were given in the command line. So when the first module was referring to a faulty definition in a module given later the context information of the error message pointed to both modules although there was no error in the first module.
- The meaning of metacharacter `%n` within the log file name skeletons has been extended. It is substituted with the string `MTC` in single mode and on the MTC, with string `HC` on the HCs or with the name of the PTC if it was given one when it was created. Formerly, this metacharacter had

useful value only on PTCs.

- The status of module parameter values given without module name in section `[MODULE PARAMETERS]` of the configuration file has been clarified. The ambiguity was introduced in the previous release, 1.7.pl0, in which the new C++ naming rules allow the definition of module parameters with identical names in different modules. If the module name is omitted or substituted with an asterisk character (*) in the configuration file the value will be set in all modules that have parameter with the given name. Error occurs if none of the modules contain module parameter with the that name. Unless the module name is given in the configuration file the run-time environment assumes that all identically named parameters have the same type.
- The following enhancements have been made on the GUI:
- The speed of automatic refresh operations on the execution window has been significantly increased. In former versions the window was refreshed after every change in the TTCN–3 test configuration, which could lead to significant delays in the GUI if the test configuration has changed too frequently (like in case of complex load test setups).

New features added to the Eclipse plugins:

- Code completion:
- Became type structure sensitive in TTCN–3 modules, allowing it to complete the fields of structured types in references.
- Became scope sensitve in TTCN–3 modules offering only proposals which could be used in the actual scope.
- Was enhanced with pre-defined skeletons in `asn1`, `ttcn`, `ttcnpp`, `ttcnin` files.
- Was enhanced with type specific, dynamically generated skeletons in ttcn files (for example function calls can be completed with the short version of the formal parameter list of the function).
- Wizards were introduced to help the creation of TTCN–3 , ASN.1 modules and configuration files.
- Changes done to a document in one editor are reflected in every other editor too, where the same document is being edited.
- Syntax coloring changes no longer need to be applied one by one.
- The help system of the Designer and the Executor plugins was separated.

Fixed bugs

- The generated C++ equivalent of enumerated types could not be compiled with GCC 2.95.x if the new naming rules were in effect. The problem was caused by the C++ enum type that was declared within the scope of the C++ class representing the values of the enumerated type. The old version of GCC accepts the casting operator only if the name of the embedded enum type is prefixed with the name of the C++ class.
- When logging the matching procedure of optional fields in record and set types the field of the value and the template was printed in the wrong order if the field of the value was set to omit. Always the value must be printed first during matching, which corresponds to the order of

arguments in built-in operation `match()`.

- The compiler generated wrong C++ code for repeat statements found within the response and exception handling parts of call statements. If the call statement was embedded into an altstep the generated code assumed that the repeat statement refers to the whole altstep. Otherwise the generated C++ code was erroneous, it could not be compiled.
- The copy constructor of class TTCN Buffer did not work properly in the Base Library. This class is used by the common API for encoding and decoding. The defective copy constructor did not copy the length indicator field of the buffer to the newly created object thus some manually written codec functions and Test Ports reported mysterious internal error messages.
- The semantic analyzer of the compiler reported false error messages while checking procedure-based operations `catch(timeout)`. Although this operation is applicable after calling any blocking signature the compiler accepted `catch(timeout)` only if the regular catch operation was allowed (i.e. the corresponding signature had at least one exception type). Of course, the operation `catch(timeout)` is allowed within the response and exception handling parts of call operations and only if the respective operation has a call timer.
- The compiler generated erroneous C++ code for the construct *value returning done* if the new naming rules were in effect. The invoked C++ function was not prefixed with the appropriate namespace if the done statement and the return type of the PTC behavior function (having attribute with *{extension "done" }*) were defined in different modules.
- Erroneous circular TTCN–3 type references pointing back to themselves with field or array sub-references (like type T[0].f1 T;) caused infinite recursion in the semantic analyzer and consequently the compiler crashed with segmentation fault.
- The utility `ttcn3 logbrowser` mis-interpreted some log entries. If the text of the log entry contained only a small integer number (like 1 or 2) the log browser presented the number as an erroneous component reference and left the field for the event text empty.
- The generated C++ code related to TTCN–3 expressions comparing optional fields of record and set types was erroneous in some cases. If two optional fields were tested for inequality the generated code could not be compiled with GCC 4.0.x or later. GCC complained about ambiguous overloading of operators. Furthermore, if an optional field containing a value of type charstring was compared with an optional field containing universal charstring the C++ code caused infinite recursion at runtime. All these errors were related to the instantiation of template member functions of C++ template classes.
- The semantic analyzer of the compiler did not check properly the value list and value range (i.e. character range) type restrictions of type universal charstring. Even some basic checks, such as the verification of range boundaries and overlapping, were skipped in previous versions.
- The compiler generated incomplete C++ type descriptor structures for some TTCN–3 types, which could lead to segmentation fault in the run-time environment during encoding or decoding using the built-in RAW or TEXT codecs. For example, if a type alias was created for type charstring with a fixed length restriction, but without coding attributes then the type descriptor of the aliased type contained information only for RAW encoding. The information about TEXT coding was not inherited from the built-in type charstring. If this aliased type was embedded into a structured type with appropriate TEXT coding attributes the TEXT encoder and decoder operations on the structured type would crash with segmentation fault.

- The compiler generated erroneous C++ initializers for literal values of type charstring containing NUL characters (i.e. characters with character code zero). The length of the strings was set correctly in the run-time environment, but the characters of the string contained memory garbage after the first NUL character.
- The algorithm that translates TTCN–3 regular expressions to their POSIX equivalents handled the TTCN–3 character set expressions incorrectly. Neither individual characters nor character ranges of the set were mapped properly (using the appropriate escape sequences) to POSIX. The resulting POSIX character set was sometimes faulty or had different meaning than the original TTCN–3 set. This problem affected the matching mechanism pattern in templates of type charstring, the arguments of predefined function `regexp()` and the attributes of TEXT encoding in both the compiler and the run-time environment.
- The TTCN–3 test components could terminate with a dynamic test case error if their communication partner terminated while a `disconnect` operation was in progress on an existing port connection. If sending an internal protocol message requesting the connection termination fails on a socket connection only a warning message is displayed rather than an error. This change makes the shutdown process of complex test configurations more robust.
- Utility `ttcn3 logmerge` could crash with a segmentation fault when it was run on several input files and one of them contained only one log entry.
- The semantic analyzer of the compiler could report false error messages complaining about missing return statements within functions having return type. This was the case, for instance, if the function contained an infinite loop without return statement realized using a goto statement. Such code fragments should not be considered faulty.
- The RAWcodec of the run-time environment crashed with a segmentation fault while decoding an integer value encoded on more than 16 octets (i.e. the value of attribute `FIELDLENGTH` was greater than 128).
- The semantic checker algorithm that verified attribute user of dual-faced port types was incomplete. The compiler did not report any error if a source type of an in or out mapping was not present on the message list of the respective port type.
- The decode type mapping rules of dual-faced port types did not consider the associated `errorbehavior` attribute. The reason was that the C++ equivalents of the errorbehavior settings were left out from the generated code by mistake.
- The default argument of the constructor (NULL pointer) was missing from the generated C++ classes implementing TTCN–3 ports if the respective TTCN–3 port type had attribute provider or user. Because of this the compilation of the generated C++ code failed when a TTCN–3 port array was created from the above port types. The C++ template class that realizes port arrays tried to instantiate its elements using the default constructor (i.e. without parameters).
- The command line version of the Main Controller (i.e. `mctr cli`) crashed with a segmentation fault after encountering the expansion of an invalid macro to a host name (e.g. ${*NonExistentMacro, hostname*}) in the configuration file. The crash occurred after reporting the appropriate error message. The reason was an uninitialized variable.
- The semantic analyzer routines of the compiler that check the correctness of the RAW and TEXT codec attributes did not work properly in some very rare cases. The problem occurred when a field of a structured type was a referenced type pointing to another referenced type, which was an alias of a built-in type and neither types had encoding attributes. After checking this

construct the internal memory structures of the compiler remained in an invalid state, which caused an internal fatal error during code generation.

- The error message of the compiler pointed to the wrong location in the source file if a (named) TTCN–3 constant was assigned to a variable and the actual value of the constant violated the subtype restrictions in the type of the variable. In this case the error message pointed to the literal value of the constant (which was apparently correct) rather than the faulty variable assignment.
- The Base Library lacked the C++ functions and operators that can implicitly convert a template (or template variable) of type charstring to a template of type universal charstring by translating the embedded matching mechanisms character-by-character. TTCN–3 modules using such constructs were accepted by the compiler, but the generated C++ code could not be compiled due to ambiguous overloads of operators. To resolve the problem a new constructor and assignment operator have been added to class UNIVERSAL CHARSTRING template, both taking CHARSTRING template as argument.
- Although the ASN.1 front-end of the compiler ignores all type constraints except the table and component relation constraints of open types, some valid type constraints were rejected by the parser. False syntax errors were generated, for instance, if a single value constraint contained values within brackets (such as `SEQUENCE` or `SET` values) or a permitted alphabet constraint (denoted by keyword FROM) contained single characters using the quadruple notation.
- The following bugs have been fixed in the GUI:
- Configuration file macros were not substituted in sections that are processed locally (such as `[MAIN CONTROLLER]`, `[GROUPS]` and `[COMPONENTS]`). Macro substitution was inefficient in the rest of the sections.
- Fixed bugs in the Eclipse plugins:
- Coloring of multi line comments in ttcn3 files could get corrupted. This was corrected by using the on-the-fly parser created intervals to identify its exact location.
- Faster operations in general.
- Calling errors of the native win32 commands corrected.

4.36. Version 1.7.pl0

Released on March 9, 2007.

New features

- The naming convention of the generated C++ code has been revised to avoid name clashes between different definitions. The use of a C++ namespace for each module eliminates all compilation problems caused by definitions with the same identifier in different modules. The proper scoping of enumerated values excludes the name conflict between two enumerated types and makes the enum-hack option unnecessary and obsolete.
- Extensibility (inheritance) of TTCN–3 component types and type compatibility between different component types is now supported by the compiler.
- The compiler can generate TTCN–3 external functions automatically in C++ to perform encoding and decoding of message types using the built-in codecs of the runtime environments. C++

programming is no longer necessary to create a complete and working protocol module. All options of the encoding shall be given in TTCN–3 as attributes of the external functions.

- Dual-faced TTCN–3 ports are now supported. Such ports have two different interfaces: internal (used when sending and receiving messages) and external (used when connecting the port to another test component or mapping to a system port). The handling of incoming and outgoing messages shall be specified using type mapping rules in the attributes of port types.
- Code generation for mixed port types is now supported.
- The expect script `ttcn3 start` has been improved:
- The script uses a built-in function to obtain the name of the computer rather than launching the command hostname, which results in faster and more reliable operation.
- Zombie processes are no longer left during the script run.
- Error handling has been enhanced, which avoids deadlocks in various error situations.
- It is no longer required to have identical names of formal parameters to the corresponding function, altstep or testcase type for the function, altstep or testcase referenced in a `refers` operation. Only the direction and the type of parameters must be identical. Different parameter names will generate warnings rather than errors.
- Formerly, the semantic analyzer of the compiler reported two consecutive error messages when it found duplicate definitions with the same identifier. Now it reports a single error message because this is a single fault. The location of the clashing definition is given as a note following the error message. Hence the counter printed at the end of compiler run shows a more realistic information about the number of errors.
- The visualization of template matching in the log files had contained the corresponding value and template fields in a misleading order. In built-in operation match of TTCN–3 the first argument is a value and the second is a template, but in the log printout the fields of the template was given first followed by the value. The order of the log printout has been reversed to be consistent with the TTCN–3 syntax.
- The status of TTCN–3 special type address was clarified.
- The BER decoding of ASN.1 type UTF8String was significantly enhanced. The newly written decoder is able to detect all possible violations of the UTF-8 character encoding and to report appropriate error messages. Error recovery is also supported, that is, an incorrectly encoded character will not prevent the decoder from processing the rest of the string. The former algorithm caused buffer over-indexing, which led to non-deterministic results, if one or more octets were missing from the end of the received octet stream.
- The compiler supports the latest official TTCN–3 language syntax according to the BNF published in version 3.2.1 of Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3. Part 1: Core Language. The only significant change is that the new BNF allows multiple external constant definitions with the same type separated by commas.
- The Runtime GUI is no longer part of the TTCN–3 Executor package. It has become a stand-alone product with number CNL 113 437 and its own version numbering.
- The following enhancements have been made on the GUI:
- A red 'X' is drawn on the symlink icons of files in the project if the related symlink does not exist, but it should.

- Excluded files are not passed to the `Makefile` generator, and will not be present in the `Makefile`.

Fixed bugs

- The compiler reported an error if the argument of TTCN–3 built-in operation `ischosen` was a value or template of an ASN.1 open type. ASN.1 open types should be visible from TTCN–3 as union types.
- The compiler generated wrong C++ code for port operation `getreply` if the corresponding signature had return type, but the operation did not specify a value match. The lack of value match means that all possible incoming return values should be accepted by the operation. However, when the referred signature template was nonparameterized the generated code matched the return value against the value match specified in the previous `getreply` operation referring to the same signature template.
- The compiler aborted with an internal fatal error during semantic analysis when a function or altstep type had a timer formal parameter.
- The error message of the run-time environment was misleading when trying to convert a record or set value containing an unbound optional field to a template of the corresponding type because the message referred to built-in function `ispresent`. The reason was that the internal realization of the value to template conversion was based on `ispresent`. The conversion algorithm has been rewritten to give a more straightforward error message.
- The semantic analyzer of the compiler did not verify the compatibility of component types when checking the component references returned by built-in operations `create`, `self`, `mtc` and `system`. For example the compiler was unable to detect if a component reference returned by create was assigned to a variable of a different component type. The C++ code generated from invalid input could be compiled to executable, but the component operations following the erroneous statement could result in run-time errors.
- The implementation of TTCN–3 operation any `component.done` was incorrect in the Main Controller. The MC gave false positive answer to the MTC if there was a PTC that was just created, but not yet started.
- The `Makefile` generator program **`ttcn3 makefilegen`** could not cope with binary files (object files, executable programs, etc.) given as command line arguments. The program entered an infinite loop while trying to determine whether the file contains a valid TTCN–3 or ASN.1 module.
- The configuration file parser of the run-time environment handled string values concatenated from two or more fragments incorrectly in sections `[LOGGING]`, `[TESTPORT PARAMETERS]` and `[EXTERNAL COMMANDS]`. In most cases the entire string was simply substituted with the first fragment and the rest was ignored. Such fragmented strings are used mostly in combination with macro substitution when only parts of the string come from macros.
- The Main Controller printed a strange error message complaining about unexpected message STOPPED KILLED when a PTC terminated because of an error while it was executing a blocking TTCN–3 operation. A typical example for this situation is when a PTC is trying to map its own port to a system port, but the operation fails in the test port for some reasons.
- The compiler crashed with a segmentation fault during semantic analysis while checking the definitions of an erroneous group. This could happen if a group contained valid definitions, but the end of the group (i.e. the closing bracket) was missing from the input file. The syntax error related to the faulty group was reported properly during parsing.

- Unsuccessful BER decoding of ASN.1 string types could lead to memory corruption causing segmentation faults in the run-time environment. The problem occurred if the string variable that was passed to the decoder to store the result had a previously assigned value. First the memory buffer carrying the previous value was deallocated, but if the decoding failed the variable was not updated properly. Thus further operations on the variable or the destructor tried to deallocate the same memory area again.
- The run-time environment created wrong BER encoding for negative INTEGER and ENUMERATED values that were smaller than -8388608. In case of such numbers the size of the value part was set to 4 octets correctly, but the encoded value represented a number that was greater than necessary by one (-8388608 instead of -8388609, -8388609 instead of -8388610, and so on). In case of input -8388609 the encoded value was invalid (overlong) according to the rules of CER and DER.
- Matching of TTCN–3 regular expressions did not work on Cello Packet Platform in parallel mode. The routines converting TTCN–3 regular expressions to their POSIX equivalents referred to macros stdin and stdout, which caused restart of the operating system. The problem occurred when matching templates of type charstring containing matching mechanism pattern or using the built-in predefined function `regexp()`. Executable test suites built in single mode were not affected by this fault.
- The following bug fixes have been made on the GUI:
- The GUI no longer tries to create symlinks for non-existing files in the project. This behavior resulted in creating symlinks in the working directory pointing to itself.
- Crash fixed when loading a FileGroup from file when no other FileGroup was present in the project.

4.37. Version 1.6.pl5

Released on November 27, 2006.

New features

- The meaning of TTCN–3 subtype constraints in nested record of and set of types has been clarified. Formerly the type restrictions were attached canonically to the outermost type, but now they belong to the innermost type embedded in record of or set of construct.
- The TTCN–3 language mode of utility `ctags` has been enhanced. Source line markers in preprocessed TTCN–3 modules are now recognized and interpreted. Recognition of the following TTCN–3 language elements has been corrected or improved: constants, variables, templates, template variables, enumerated types, nested type definitions.
- The subroutine of utility **`ttcn3 logmerge`** that extracts the test component identifier from the name of the log file has been improved. The new algorithm works better when a custom file naming convention is used and the component reference is separated with a dot (.) character rather than a dash (-).
- The `Makefile` generator functionality has been significantly improved and moved from the compiler to a separate utility called **`ttcn3 makefilegen`**. The following changes have been made:
- The program automatically recognizes TTCN–3 include files with suffix `.ttcnin`.

- The detection of file name clashes and filtering of generated files related to TTCN–3 preprocessing and/or central storage has been improved.
- Warnings are displayed if the central storage or preprocessing options are used unnecessarily or the options are not used, but they should be.
- The program detects if the names of input files contain spaces or other special characters that cannot be handled by the make utility.
- The path transformation rules are also applied on the name of the target executable.
- On Windows the `.exe` suffix is appended to the name of the target executable only if the file name does not contain that.
- The compiler and the run-time environment supports the evaluation of some nonstandard TTCN–3 macros. The detailed description of the macros can be found in the Reference Guide.
- Macro substitution modifier hostname was introduced in the run-time configuration files. This modifier allows the substitution of macro values containing a DNS name or IP address into sections `[GROUPS]`, `[COMPONENTS]` and `[MAIN CONTROLLER]`.
- The following enhancements have been made on the GUI:
- Exclude and Include from build functionality has been added to source files included in File Groups.

Fixed bugs

- In some cases the GUI did not load the user-defined test sets that were stored in the project.
- The compiler generated wrong (uncompilable) C++ code for the location information when the name of the input file contained special characters (like the backslash) that require escaping in C++ string literals.
- The compiler aborted with an internal fatal error during semantic analysis if a RAW attribute POINTERTO referred to a non-existent field. The abort occurred after printing the relevant error message.
- The performance of utility **`ttcn3 logformat`** was very poor on log files containing very long lines. It took several minutes even for a powerful processor to format a file containing only a few megabytes of data. The reason was regular expressions that were supposed to match the beginning and end of TTCN–3 test cases were given in an inefficient way so that the parser generated by utility `flex` could not process the input with a linear algorithm. The complete internal structure of **`ttcn3 logformat`** has been redesigned to eliminate the performance critical regular expressions. The block indentation algorithm has been replaced with a more robust one that, for instance, can re-indent previously indented log files.
- The compiled did not issue warnings for some TTCN–3 reserved words that can appear in ASN.1 modules thus making the corresponding ASN.1 value or type field unreachable from TTCN–3 . The values of type verdicttype (like `none` or `error`) and the binary operators of TTCN–3 (like `not4b` or `xor4b`) were not detected and reported by the compiler.
- The compiler crashed with a segmentation fault when generating code for an ASN.1 open type that did not have table constraint. This problem was introduced by a bugfix of the previous release that was about missing C++ classes related to open types. Nevertheless, an ASN.1 open type without table constraint is useless in TTCN–3 because there is no way to create templates

for it. The compiler now reports warnings when encountering such open types.

- The semantic analyzer of the compiler verified the arguments of built-in TTCN–3 conversion functions int2hex and int2oct incorrectly. In some cases when the arguments were constants and the given value did not fit in the given length the semantic analyzer did not report any error, but the compiler aborted with an internal fatal error during constant folding while doing the conversion.
- The compiler generated incorrect C++ code for TTCN–3 for loops if the boolean guard expression was constant false. In this case the entire loop was substituted with an empty C++ statement although the initial variable assignment had to be executed exactly once in all cases. The wrong code could cause problems if the initial assignment had side effects (e.g. it called a function or modified a variable defined outside the for loop).
- The compiler and the run-time environment supports the short-circuit evaluation of logical `and` and `or` operations as it is required by the TTCN–3 standard. If the result of the operation can be determined based on the first (left) operand (i.e. the first operand is false in case of and or true in case of or) the second (right) operand will not be evaluated at all. The writer of the TTCN–3 code can exploit this behavior if the right operand may have side-effects or cause a dynamic test case error.
- The compiler reported wrong location information for some TTCN–3 parse errors. For example, when a literal integer value was too large the error message indicating the overflow referred to the previous token of the input rather than the number itself.
- The compiler detects if the same TTCN–3 or ASN.1 input file is passed to it more than once and reports a single, but appropriate error message about this. Formerly the file was parsed and analyzed several times, the same error messages were repeated and only an error message complaining about duplicate module identifiers referred to the real problem.
- The generated C++ code suffered aliasing problem in case of some TTCN–3 function and altstep calls. If a component variable was passed as in parameter to a function having runs on clause and the called function modified the same variable the value of the in parameter has also changed. This behavior breaks TTCN–3 semantics since in parameters shall always be passed by value. The code generator has been fixed by adding explicit copy constructor invocations at the calling side when necessary.
- The configuration file handler of the GUI reported syntax error and created syntactically invalid configuration file if a compound module parameter within brackets contained a typed macro reference. The problem was caused by a subroutine of the GUI that performed block auto-indentation on all configuration file entries. The algorithm did not recognize the brackets of the macro reference and inserted whitespace between the "$" and "{" characters.
- The compiler did not export the type descriptors of the embedded TTCN–3 and ASN.1 types to the generated header file. Thus the generated code was uncompilable if the embedded type was referenced from another module using field or array subreferences.
- The compiler performed incomplete semantic analysis on the arguments of built-in operations `ispresent` and `ischosen`. For instance, when the operations were used in the value of a constant the compiler did not report an error if the argument referenced to a template or template variable.
- Built-in operation `ischosen` was not implemented properly in the run-time environment. When the argument was an unbound union value the operation returned false rather than causing a

dynamic test case error.

- The macro processor of the run-time configuration files did not allow the substitution of macros with empty values as `bitstring`, `hexstring` or `octetstring` value.
- When a project file was passed to the GUI as command line argument the program did not convert the simple file names or relative pathnames to absolute paths. Thus the same file could appear several times on the list of recently opened projects.
- In some cases the GUI crashed with a segmentation fault when selecting and opening a file from the list of recently used projects. The reason was that the program tried to use a string reference pointing to a value that was already destroyed.
- The compiler checked the actual parameter of a parameterized ASN.1 assignment only when the parameter was referenced through the dummy reference. This allowed syntax and semantic errors in the actual parameters to remain hidden if the dummy reference was not used within the body of the parameterized assignment. The corrected algorithm checks all actual parameters as well as the entire instantiated assignment at the point of instantiation when processing the parameterized reference.
- The generated C++ code could contain name clashes between types and values generated at the instantiation of parameterized ASN.1 assignments. The clash could occur, for example, when a parameterized information object assignment had a field setting and a parameter (dummy reference) with identical names (apart from the leading &).

4.38. Version 1.6.pl4

Released on July 31, 2006.

New features

- The `Makefile` generator of the compiler supports the use of C preprocessor on TTCN–3 code. The TTCN–3 parser of the compiler understands the preprocessor's line directives thus the error messages refer to the right position in the source code.
- The non-standard TTCN–3 language extension about function, altstep and testcase references and the related operations, such as apply, refers and derefers are now supported.
- The non-standard TTCN–3 language extension about non-mandatory parameters are now supported. It is now possible to assign default settings to formal parameters and omit actual parameters when invoking the respective definition.
- The run-time environment supports the use of UNIX domain sockets for the realization of TTCN–3 port connections. This results in more efficient data transfer on native UNIX platforms compared to the former TCP-based method when both endpoints of the port connection are located on the same computer.
- Test execution is now supported on Cello Packet Platform with OSE Delta operating system.
- The memory requirement of the run-time environment has been improved by a few percents. For instance, in older versions every string value allocated 3 bytes more than it is necessary.
- The execution time as well as the memory usage of the compiler has been improved by a few percents. The generic containers of the abstract syntax tree have been optimized.

- In parallel mode it is allowed to connect a new Host Controller to the Main Controller while a test case is running. The newly connected HC gets the configuration information immediately and takes part in load balancing as soon as possible.
- The Main Controller supports the relocation of a newly created PTC to another host if it is impossible to create a new process on the host that was chosen initially. If the Host Controller cannot create a new process the MC will exclude the host from load balancing until it reports that overload has ceased.
- The logging routines of the run-time environment support the handling of several log events in a stack-based concept. It is allowed to call TTCN Logger:: `begin event()`, `log event()` and `end event()` in the middle of another event. This allows the use of `TTCN-3 log()` statements in recursive contexts (e.g. when the return value of a function is being logged and the function itself also contains `log()` statements).
- During load tests it often happens that two test components continuously send data to each other through connected ports. Earlier versions of the run-time environment used blocking send operations on the underlying TCP connections, which could lead to deadlocks if all network buffers were full in both directions. Starting from this version non-blocking send operations are used on port connections and when sending would block the test component tries to increase the size of outgoing network buffer if this is supported and allowed by the operating system. If the buffer cannot be enlarged any further the send operation will block and also try to handle incoming messages. In this case a warning message indicates the unusual operation of the run-time environment (i.e. new incoming messages might appear in the port queues while only sending is performed).

NOTE This improvement prevents the test system from deadlocks in many cases, but there is no warranty that the run-time environment can cope with all possible situations (for example, infinite port queues exist only in theory). The ultimate solution is to restrict the amount of outstanding data on port the connections by well designed flow-control algorithms in the TTCN–3 code (e.g. by introducing an application level sliding-window protocol).

- The compiler's output generation algorithm has been improved. In order to perform selective updates the compiler first writes the generated code into a temporary file and updates the final target only if the contents of the existing file differs from the newly generated one. In earlier versions the temporary files were always created even if the real target files did not exist. Now the output is written directly to the target file if a previous version does not exist, which can result in significant speedup in the compiler's operation when building large projects from scratch.
- The compiler supports error recovery in the RAW and TEXT encoding attributes of TTCN–3 types. Formerly a single syntax error in the attribute text caused the compiler to stop immediately.
- Macro substitution is now supported in section `[MAIN CONTROLLER]` of the configuration file in both command-line and GUI modes.
- Basic arithmetic operations and concatenation of character string values are now supported in the run-time configuration file. This allows the more flexible use of macros since a module parameter or a configuration option can be derived from several macros and/or constant

values.

- The TTCN–3 parser of the compiler allows in-line compound values in the operands of expressions.

NOTE	Compound values can only be used with comparison operators (i.e `==` or `!=`) and the other operand has to be a referenced value.

- Metacharacter `%c`, which denotes the name of the current testcase, is now supported when specifying the names of the log files in section `[LOGGING]` of the configuration file.
- The run-time environment knows about the names of TTCN–3 timers. Log entries and error messages related to timers now contain the name of the corresponding timer, which makes the analysis of logs easier.
- The following enhancements have been made on the GUI:
- Execution Window: When configuring the host controllers with 'Detailed actions' switched on, the configuration file to use can be chosen from a list.
- Project Window / File tree: File Groups section added.

Fixed bugs

- The behavior of the run-time environment could be undefined if the left operand of TTCN–3 concatenation operator (`&`) was a bitstring element (containing a single bit) and the right operand was a bitstring value. If the right operand was long enough the operation could lead to segmentation fault because a programming mistake allowed over-indexing in memory.
- The pre-defined functions `bit2int`, `hex2int` and `oct2int` refused the conversion with a dynamic test-case error if their argument contained more bits than the integer representation of the given platform. Now the conversion is possible if the argument is longer than this limit, but after cutting the leading zero bits, hex digits or octets the result can be represented as a non-negative number in the integer type. On the other hand, the conversion was incorrectly possible if the input had the same size as the integer type, but the result was a negative number. Both the compiler and run-time versions of these functions were affected by this fault.
- The compiler reported an inappropriate warning message when checking a value or template of an empty record or set type. The message stated that all elements were not used symbols (i.e. -) although the value or template did not contain any elements.

NOTE	The only possible value of these types is *{}*.

- The error messages of the compiler contained incomplete or missing location and context information hierarchy if there was an error deeply within a complex TTCN–3 expression.
- The Main Controller stopped in a deadlock in batch mode if the MTC and all alive HCs terminated at the same time and closed their control connections unexpectedly.
- Empty character strings in TTCN–3 pattern templates were not handled properly. The compiler accepted empty strings after the pattern keyword, but generated faulty (uncompilable) C++ code. Moreover, the run-time environment reported dynamic test case error if the pattern was empty (e.g. after variable substitution). Of course, empty charstring patterns are allowed and they match the empty string values only.

- The TTCN–3 parser of the compiler rejected array index sub-references in type references. That is why a nested unnamed type of a record of type could not be addressed.

NOTE	The array indices in type references are handled in a special way in the compiler: it is verified that the value in square brackets should be of type integer, but the number itself is not used for anything.

- The compiler generated wrong (uncompilable) C++ code for ASN.1 open types in some cases. The C++ equivalents of some types and values were missing if an unnamed information object set was used in the open type's table constraint. This was the case, for instance, if the open type was embedded into a parameterized type assignment and the object set was defined in-line in the actual parameter list of the type reference.
- Incremental compilation of ASN.1 modules did not work properly if the instantiation of some parameterized type assignments changed since the last full re-compilation of all modules. Changing the order of files in the compiler's command line could cause similar problems. Module-based incremental compilation uses the concept that the C++ equivalent of a module shall only depend on the module itself and the imported modules. Importing modules that use some definitions from the module must not influence the generated code. The C++ equivalents of parameterized assignments are generated when the assignment is instantiated by a parameterized reference (because only these instances can be visible from TTCN–3). The instances are created in the module where the parameterized reference is because the actual parameters might be visible only from that module. Different instantiations of the same parameterized assignment are identified by instance counters. Earlier versions of the compiler used one instance counter for each parameterized assignment. This can cause problems when there is a parameterized assignment in module A and it is instantiated from B and C. If the number of instantiations in B changes this would change the instance counters of C. Thus the generated files of C became outdated although C does not import from B. Starting from this version the instance counters are associated with target modules. In the above example the parameterized assignment in A has two counters: one for B and another for C. This change required a new naming convention for the C++ classes of instantiated types: the identifier of the target module became part of the name, which resulted in longer C++ identifiers.
- The compiler assigned wrong identifiers to some ASN.1 open type members. Open types are mapped to implicit CHOICE types and the alternatives are the distinct types from the table constraint's object set. The alternative identifiers of the CHOICE are derived from each type's name by changing the first letter (which must always be a capital letter) to lower case. This mapping of names did not work properly if the elements of the object set were the instances of a parameterized information object assignment. For instance, different types got the same identifier (and therefore the generated C++ code was invalid) if the respective object field was a dummy reference pointing to the object's formal parameter. When constructing open types the compiler generates a warning message if a member type has a complicated constructed name and thus the alternative will get a strange identifier. These warnings were missing too from the instances of parameterized types.
- The semantic analyzer of the compiler did not accept value (character) range matching mechanisms in templates of type universal charstring.
- Values of type charstring that were created by indexing a single-character element of another charstring value had incorrect internal representation in the run-time environment. The run-

time representation of charstring values contains a length indicator and a terminating zero byte is also appended at the end of the string. However, the indexing operator set only the length indicator in the result so the builtin functions that relied on the terminating byte (e.g. str2int) worked incorrectly.

- The compiler did not handle the unterminated TTCN–3 comments properly at the end of file. Unterminated line comments (beginning with "//" and when the file did not end with a newline character) were accepted without any error message. Moreover, an unterminated block comment (beginning with "/*") could confuse the compiler so that it was unable to parse further files.
- The compiler aborted with an internal fatal error when detecting an erroneous reference in the argument of built-in operations `ispresent` or `ischosen`. The proper error message was displayed before the abort.
- The compiler generated wrong C++ code (which worked incorrectly) for TTCN–3 function calls in very rare cases. The aliasing problem occurred if the same variable was passed to a function as both in and `inout` parameters. When the function has updated its `inout` parameter (i.e. the variable of the caller) the value of the in parameter has been changed although it should have been constant throughout the entire function call.
- The compiler aborted with an internal fatal error during the comparison of constant values of TTCN–3 type objid and its ASN.1 equivalents (i.e. OBJECT IDENTIFIER and RELATIVE-OID). This could occur, for instance, while the compiler was checking the component relation constraints of some ASN.1 protocol modules.
- The compiler's semantic analyzer rejected the return statements within the body of altstep definitions although the latest version of the TTCN–3 standard allows them.
- Checking of built-in operation `valueof` was implemented incorrectly in the compiler's semantic analyzer. For instance, the compiler did not report error if the argument of `valueof` was of an incompatible structured type. Another bug in the algorithm caused segmentation fault during code generation if the argument was an in-line modified template.
- The compiler's semantic analyzer verified the uniqueness of field names in TTCN–3 record, set and union types and the correctness of embedded types in a single step. This strategy resulted in false error messages as well as memory leaks if a type was part of multiple recursion loops and had RAW encoding attributes that referred to its recursive fields, such as `LENGTHTO` or `CROSSTAG`. The algorithm has been restructured into two distinct steps so that the embedded types are not analyzed until the field names are completely verified.
- The RAWencoder of the run-time environment could crash with a segmentation fault while encoding a union value with TAG attribute. This was the case if the field that was referred by TAG (which is used to select the right alternative during decoding) had an invalid value. The situation was similar if more than one values were listed for the current alternative and the value to be encoded did not contain the first one from the list.
- The compiler could report false errors or abort with internal fatal error if the length restriction of a TTCN–3 template contained an arithmetical expression of type integer rather than a literal value.
- The ASN.1 parser of the compiler classified the information object assignments incorrectly as value assignments if the governor information object class was parameterized. Due to the false recognition the semantic analyzer reported inappropriate error messages and crashed with a

segmentation fault.

- The compiler could crash with a segmentation fault during semantic analysis when a parameterized ASN.1 assignment was instantiated and its right-hand side contained an in-line information object set definition (e.g. in a table or component relation constraint).
- The following bug fixes have been made on the GUI:
- Project Window / File tree: The GUI crashed if included projects contained test sets. Now they are displayed correctly, and are offered for execution in the Execution window.
- Execution Window: Log lines containing only a number are aligned to the left of the table cell instead of aligning to right.
- Project Window / File tree: Fixes in included project handling. There is no more warning about read-only project file for included projects.
- Project Window / `Makefile` generation: Sources under "Misc Files" are not included in the `Makefile`. They get automatically" Excluded from build" when adding them to the project.
- Process Monitor: Normal text color is not changed after clicking on highlighted text.
- Execute Dialog: Controls, Test Cases and Test Sets are placed in right order in the list.

4.39. Version 1.6.pl3

Released on December 16, 2005.

New features

- Arrays of TTCN–3 language are now fully supported. This includes multidimensional arrays, arrays within type definitions and index ranges with lower and upper bounds. Index values in array references are verified against the array bounds both in the compiler (as long as the index value is constant) and in the run-time environment. In case of index overflows or underflows a proper run-time error message is generated instead of the unpredictable behavior of former versions.
- The template matching mechanisms subset and permutation are now fully supported.
- Semantic analysis of TTCN–3 group names has been implemented.
- TTCN–3 with attributes that belong to modules or group definitions are now processed and propagated to the embedded definitions. Consistency checking of attributes and attribute qualifiers has been improved.
- The compiler supports generation of `Makefiles` to be used with central storage of precompiled files. Administrators of larger projects can collect and compile the common TTCN–3 and ASN.1 modules that change rarely (type definitions, test ports, etc.) in a central directory. Test developers can build their projects without re-compiling the common modules by taking the pre-compiled object files and C++ header files from the central storage. Usage of this feature can save compilation time as well as disk space for the individual developers.
- The configuration file of the run-time environment has been enhanced. Two new sections (`[INCLUDE]` and `[DEFINE]`) have been introduced. This allows modular configuration information spanning over multiple files and usage of macros and environment variables within the configuration files.

- Built-in TTCN–3 operations `ispresent`, `ischosen`, `lengthof` and `sizeof` are now applicable to templates and template variables in addition to values (constants, variables, etc.).
- The RAW encoder/decoder supports forward references in `CROSSTAG` attributes. It is allowed that the decoding of an union field depends on another field that is defined later in the same record if the union field is mandatory and all embedded alternative types have the same, fixed size. During decoding such union fields are processed out of order. In the first round the field is simply skipped, it is processed only when the field that determines which alternative to choose is successfully decoded.
- The run-time environment and the MC supports the killing of the MTC process if it is not responding because it is stuck in an infinite loop. Formerly an unresponsive MTC could cause deadlock in the whole test system. The termination of MTC can be initiated from a PTC using operation `mtc.stop` or from the user interface by issuing a `stop` command or pressing the **Stop** button. The MTC process is killed only if it does not respond to the stop request before the guard timer expires.
- The IP addresses used for the control connections between the Main Controller and Host Controllers and test components are now configurable. These options are useful when testing is performed on multi-homed hosts (i.e. computers with multiple network interfaces and/or IP addresses). In case of MC the address of the TCP server can be restricted using a new option in the configuration file. On the HC side the client IP address can be set using a command line switch.
- The expect script `ttcn3 start` accepts absolute or relative path names as the name of the ETS. If the configuration file is not specified explicitly in the command line it is searched in the same directory as the ETS is in.
- The following enhancements have been made on the GUI:
- Configuration Editor: Updated to handle the new sections (`[INCLUDE]` and `[DEFINE]`).
- Project window: Subprojects can be added to the main project. `Makefile` generation is updated to generate `Makefile` supporting the central storage.
- Project window: Project tree now remembers the opened and closed state of the main types.
- Project window: When adding a file to the project with a name which already existed within the project's files, replacing the old file with the added one is offered.
- GUI Preferences: The line number switch for the external editor can be configured.
- Process monitor: Highlighting of the source code file paths with line numbers, errors, warnings and identifiers added. The source code with the line number can be clicked, and if configured well in the preferences, the external editor will be opened at that line.
- Project Properties: Added setting of whether to create symlinks with absolute or relative target path.
- Project Properties: Added setting of whether to use absolute paths in the generated `Makefile`.
- Execution window: **Follow last log line** checkbox added to the toolbar to set the log area to scroll to the last added line or not.
- Execution window: The items in the Execute Dialog are not sorted alphabetically by default. The order of the controls and test cases will be the order they were returned by the test case collection (the order they were written in the source modules).

- GUI Preferences: %active config path added to the selectable GUI variables for use in the User Menus.
- The Log Browser supports searching of strings within the event texts.

Fixed bugs

- Templates with length restrictions produced strange printout in the run-time logs due to a missing break in a C++ switch-case statement.
- The run-time location information of else if statements pointed to the wrong line of the TTCN–3 code. For example, if an error occurred in the boolean expression of an else if statement the dynamic test case error message of the log file contained the line number of the first if statement. This made difficult to find the reason of the error since the first boolean expression, which the error message referred to, was correct.
- The compiler generated faulty C++ code if the last statement of a TTCN–3 statement block was a select-case statement. The C++ compiler complained about that the last label of the block was not followed by a statement. The syntaxerror was corrected by adding an empty C++ statement (i.e. a semi-colon) after the corresponding label.
- The compiler crashed with a segmentation fault during code generation if the boolean expression of a TTCN–3 if, for or while statement contained an in-line compound expression that could not be mapped directly to a C++ expression.
- If the actual value for a module parameter of a record of or set of type contained fewer elements in the configuration file than the default value in the TTCN–3 module then the surplus elements of the default value remained present after processing the configuration file.
- The compiler could produce a segmentation fault when generating the C++ structures for describing TEXT encoding of enumerated types. The reason was that the code generation subroutine addressed internal data structures in a wrong manner.
- The expect script `ttcn3 start` detects if the configuration file instructs the MC to run in batch mode. Instead of causing a deadlock the script terminates with a proper error message.
- The compiler aborted with an internal fatal error if a TTCN–3 template reference tried to address a sub-field or element of a template (or template field) containing a generic wildcard like ? or *. Now a proper error message is printed in these circumstances.
- The value comparison algorithm for set of types was not implemented in the compiler. These functions are applied during constant folding (i.e. when all operands of a TTCN–3 expression are available at compilation time) to substitute the expression with its result in the generated C++ code. In case of set of values the comparison algorithm of record of types was applied, which could give incorrect result because it considered the order of elements.
- The TTCN–3 language mode of program `ctags` did not handle the string literals properly. If a TTCN–3 string constant contained a special character like *{* or *}* `ctags` got confused and ignored the definitions in the rest of the file. Moreover, the program failed to parse character string patterns within templates.
- The run-time environment did not specify the client IP address for the control connections towards the `MC`, thus it was chosen by the kernel independently for every connection. On some multi-homed Sun Grid environments it could happen that a test component got a different local IP address from the kernel than the `HC` (i.e. its parent process). In this case the `MC` refused the

component's control connection and the respective create operation failed with a dynamic test case error. In the new version only the `HC`s can get automatically assigned IP addresses (unless the local address is explicitly given in the command line). The test components (child processes) always assign the HC's local IP address explicitly before establishing their control connections.

- The RAW decoder of the run-time environment contained a memory leak. When the decoding of an enumerated value was unsuccessful a memory block containing a temporary string was not deallocated. This could lead to significant memory growth in case of load testing, especially when the types used the TAG attribute, the decoding of which is implemented using backtracking.
- The run-time environment caused a dynamic test case error when an optional record or set field containing omit value was compared with a mandatory field or a simple value. The false error message complained about unbound operand of comparison. Now those operations return the appropriate boolean result: false in case of operator `==` and true in case of `! =`.
- The dynamically linked version of the TTCN–3 Base Library (i.e. libttcn3-dynamic.so and libttcn3-parallel-dynamic.so) did not contain all functions that are necessary for building the executable. If those libraries were used the linker complained about a lot of unresolved function references. Thus only the statically linked versions of the Base Library (i.e. `libttcn3.a` and `libttcn3-parallel.a`) were usable. The reason was a mistake in the built script (the script did not add some object files to the dynamic libraries).
- The compiler aborted with an internal fatal error if it encountered non-ASCII characters (i.e. character codes above 127) in the argument of predefined function `char2oct()`.
- The expect script **`ttcn3 start`** could hang in a deadlock after issuing the `emtc` command. The reason was that the expected output lines of the Main Controller (MC) came in the wrong order due to a race condition within the MC. To avoid such problems in the future both the script has been enhanced and the race condition has been eliminated.
- The utility **`ttcn3 logformat`** did not separate the log entries belonging to different test cases (using the `-s` option) when the timestamps contained the date (i.e. option `TimeStampFormat` was set to `DateTime` in section `[LOGGING]` of the configuration file). In this case all entries were dumped to standard output. This was because a incorrect regular expression was specified in the parser for those timestamps.
- The utility **`ttcn3 logmerge`** did not detect or handle write errors (e.g. when the disk became full the user was not notified that the output file is incomplete). Now the utility reports the appropriate error message and terminates immediately when it encounters any error related to file operations. Moreover, it was not handled properly when the limit of simultaneously open files was reached on Solaris 2.6. In such situations the program got into an infinite loop of printing error messages.
- The program routine that is responsible for processing TTCN–3 character patterns contained a memory leak. Although this fault affected the compiler as well, it could cause serious problems in the run-time environment during long-duration performance tests. The memory usage of TTCN–3 test components increased continuously if they used charstring templates containing pattern matching mechanisms or the predefined function `regexp()`.
- Utility **`ttcn3 logmerge`** could cause segmentation fault if one of its arguments was a path name containing . or .., but its file name part did not contain dash (i.e. -) character.

- The compiler generated invalid C++ code if the first operand of predefined function `substr` was a charstring element. The C++ compiler complained about an ambiguous function overload.
- The following bug fixes have been made on the GUI:
- GUI Preferences: Successful saving of the settings is checked, and an error dialog is shown when it fails.

4.40. Version 1.6.pl2

Released on August 1, 2005.

New features

- C++ code generation for TTCN–3 interleave statements is now supported.
- The compiler reports a warning if an ASN.1 identifier is a TTCN–3 keyword so it is unreachable from TTCN–3 . The warning is generated only if the ASN.1 identifier should be visible from TTCN–3 (e.g. there is no warning if a named bit or an information object is identified with a TTCN–3 keyword). The names of TTCN–3 predefined functions are also considered to be keywords.
- The following enhancements have been made on the GUI:
- Project window: When creating new configuration file, the default assignments are commented.
- Project window: When creating new TTCN-3 or ASN.1 files, the file and module names are checked against basic mistakes.

Fixed bugs

- The Main Controller updated its internal state table incorrectly when an idle nonalive type PTC was stopped explicitly by another test component. The PTC was instructed properly to terminate, but MC reported unexpected events when the PTC finished. As a consequence of this MC assigned error verdict to the corresponding PTC at the end of the testcase regardless its real final verdict.
- The command line version of the Main Controller (i.e. `mctr cli`) stuck in a deadlock if the MTC terminated unexpectedly in batch mode. This could happen, for example, if a faulty Test Port caused the MTC to crash with a segmentation fault. Now this situation is handled explicitly by the MC, which reports an error and terminates with unsuccessful exit status in this case. The same problem was also present in the expect script `ttcn3 start`, which invokes `mctr cli` in interactive mode. The script was fixed, too.
- The Log Browser displayed those log entries incorrectly that consisted of a timestamp and an integer number. Such entry can be produced, for example, by passing an integer value to the TTCN–3 `log()` statement. When seeing this line the Log Browser got confused and recognized the integer number as a component reference and considered the entire next log entry (line) as the text of this event. * The compiler reported strange error messages on valid TTCN–3 input if an encoding token of a TEXT attribute contained one of the following characters: #,], *}* and a decoding token was not specified for the same attribute. When the decoding token is omitted the compiler derives it from the encoding token automatically. Since the encoding token is a simple charstring value, but the decoding token is a TTCN–3 pattern the characters that have

special meaning in patterns have to be escaped. The above three characters, however, were not escaped during the transformation.

- The BER decoder for one-octet character strings (i.e. ASN.1 character string types that are mapped to charstring in TTCN–3) created incorrect string values. In the internal string representation the length and the content characters were set properly, but the terminating NUL character (zero byte) was missing at the end of the decoded string. In most cases the fault remained invisible because the majority of built-in string operations use the length field. However, the predefined function `str2int` uses the NUL terminated string without the length, which caused undefined behavior if its input came from BER decoding.
- It was impossible to specify literal charstring values in RAW encoding attributes of TTCN–3 types. Since the coding with attribute is delimited with quotation mark characters the beginning and end of the embedded string has to be escaped (i.e. the quotation mark character shall be preceded with a backslash or a double quotation mark character shall be used). However, the attribute parser of the compiler used a wrong regular expression, it expected a single quotation mark character as delimiters in charstring literals.
- The code generator of the RAW enc/dec did not prefix the enumerated values if the `-E` (enum-hack) compiler switch was turned on. Thus the generated C++ code did not compile in this case.
- The documentation of RAW attribute `PRESENCE` contained a mistake in syntax description. The attribute parser of the compiler required a pair of brackets for multiple values, but the documentation does not. Now the parser accepts both syntax variants to preserve backward compatibility.
- The RAW encoder encoded the empty charstring values incorrectly. The result of encoding was nothing (empty string) even if the FIELDLENGTH attribute was set to a positive number.
- The following bug fixes have been made on the GUI:
- Execution window: GUI hung in a deadlock when displaying errors or warnings during test execution.
- Project window / process output on Cygwin: endless loop avoided on Cygwin when executing a process.

NOTE	Cygwin is not officially supported by TCC.

4.41. Version 1.6.pl1

Released on June 30, 2005.

New features

- The TTCN–3 parser of the compiler works based on the final BNF of version 3.1.1 of the Core Language standard. This implies the following minor additions:
- Nested (unnamed) TTCN–3 type definitions can now be used.
- The log statement accepts template instances (including inline and inline modified templates) in addition to values and template references.
- The halt port operation is now supported.

- The TTCN–3 select-case statement is now supported.
- Test case static test configurations, that is, alive TTCN–3 test components and the related operations are now supported. The following new component operations have been added: create alive, kill, killed, alive. The meaning of the following existing component operations has been extended: stop, done, running.
- Error recovery in the TTCN–3 parser has been significantly improved.
- The following enhancements have been made on the GUI:
- Project window: added toolbar icon (hotkey: F9) for executing the `[EXECUTE]` section of the config file.
- Project window: toolbar icons (and separators) can be sorted by adding @place number to the end of their name, i.e. `MAINTOOLBAR_>_Make -j2@5`.
- Project window: file tree accepts the <**Del**> key.
- Project window: hotkeys: CTRL+C stops current process execution, CTRL+L clears current process window, CTRL+W closes current process monitor if it is enabled.
- Project window: executable is offered to be removed when changing the Build/Execution mode.
- Project window: symlinks are created as relative symlinks instead of absolute.
- Project window: log files are refreshed right after loading the project.
- Project window / Process monitor: performance improved.
- Config file editor: save compound values formatted.
- Execution window: improved stability and memory consumption on high load.
- Execution window: button sizes are the same when switching to Detailed.
- Execution window: test execution does not need any config file.
- Execution window: waits until the specified number of HCs (NumHCs in the config file) connect before starting the test execution.
- Execution window: state strings are the same as used in the CLI version.
- Execution window: removes the temporary config file from /tmp when finished.
- Execution window: added timed execution; the start date/time can be set before starting the test in the test case selection dialog.
- Execution window: double click is accepted in the test case selection dialog.
- Execution window: empty selection is not accepted.

Fixed bugs

- The compiler aborted with an internal fatal error when checking a TTCN–3 expression (or sub-expression) if all operands of an arithmetic or logical operation were available at compilation time[2] and one of the operands was a referenced value pointing to a TTCN–3 constant of type float or ASN.1 value of type REAL.
- The generated C++ code was uncompilable if the name of an ASN.1 built-in string type (e.g. UTF8String) was used as identifier in a TTCN–3 module or the name of a TTCN–3 built-in conversion function (e.g. bit2int) was used as ASN.1 identifier. Now the compiler appends a

single underscore character to the C++ equivalents of these identifiers to avoid name clashes with the definitions of the Base Library.

- The compiler caused segmentation fault during semantic analysis if the type of a constant or module parameter was erroneous (e.g. the referenced type was not imported) and the value of the constant or the default value of the module parameter contained an embedded record or set value (i.e. a value with TTCN–3 assignment notation). The problem occurred after reporting the relevant error message and only if when C++ code generation was requested (i.e. the command line switch `-s` was not used).
- The GUI crashed at startup with segmentation fault if the environment variable TTCN3 DIR was not set or the configuration files and images were not found under `$TTCN3 DIR/etc/gui`.
- The compiler analyzed only the top-level expressions in the optional arguments of TTCN–3 `create` operation, that is, recursive checks were not performed on the embedded sub-expressions. Because of this some kinds of errors were not detected and the compiler could abort with internal fatal error during code generation even on valid input (e.g. when a sub-expression contained a simple reference pointing to a variable).
- The semantic analysis did not detect when a variable defined in the header of a TTCN–3 for statement shadowed a definition in an outer scope unit with the same identifier.
- The compiler generated invalid code if the argument of a TTCN–3 `send` operation referred to an optional field of a record or set value (constant, variable, module parameter, etc.). The C++ compiler complained about ambiguous overload of a member function in the port class.
- The Main Controller caused segmentation fault in both command-line and GUI modes if the `stop` command was issued while a control part was running and no testcase was executed before.
- In parallel mode the MTC crashed with internal error if a test case was executed after stopping test execution from MC. When the `stop` command of MC interrupted the test execution the actual test case was not terminated properly and MTC remained in an inconsistent state.
- The compiler rejected valid TTCN–3 input if one operand of the comparison operator was a reference pointing to an imported (constant or module parameter) definition and the other operand was an enumerated value. The error message complained about that the compiler was unable to determine the type of operands in the expression.
- The MC reported internal error (unexpected STOPPED message) when a TTCN–3 map operation failed because the Test Port called TTCN error. This error situation is now handled properly.
- The compiler did not allow to assign omit value from an optional field of a TTCN–3 constant to a template variable. The reason was a mistake in the a semantic analyzer routine because it did not propagate an option recursively.
- The semantic analyzer of the compiler reported wrong circular recursion error message if a template reference in a function or testcase contained an array sub-reference the index of which was not constant. For example, this was the case when a for loop iterated through the elements of a record of template or template variable.
- The compiler accepted any kinds of expressions (not only the boolean ones) in if, for, while and do-while statements because of a programming mistake in the semantic analyzer. Moreover, in most cases the generated C++ code was compilable (thus the problem remained hidden) if the type of the expression was a built-in TTCN–3 type (e.g. charstring).
- The compiler generated wrong (uncompilable) C++ code if the boolean expression of a if, for,

while or do-while statement referred to an optional boolean field of a record or set value.

- The semantic analyzer routine of the compiler that checks the length restriction of templates contained several bugs. In certain cases this could cause segmentation faults or aborts in the compiler or the acceptance of invalid inputs. For example, the bugs could be triggered with the following constructs: length ranges with infinity as upper limit, length restrictions for the universal charstring type, length restrictions combined with specific string values.
- The run-time environment did not pass the test port parameters to the respectivetest port objects if the parameter was assigned to named test components. One part of parameter processing routines that handled named test components was inactive due to a programming mistake.
- The run-time environment produced a misleading warning message if an expired TTCN–3 timer was re-started. The message complained about re-starting a running timer although the built-in running operation returns false on expired timers.
- The semantic analyzer of the compiler was unable to determine the type of string patterns in receiving port operations if the corresponding port type had more than one incoming message types. The statement was accepted only if the explicit type specification was present before the pattern (e.g. `p.receive(bitstring:'1?0□B)`;) although the type of the pattern is unambiguous.
- The GUI did not pass the configuration option `KillTimer` to the Main Controller. The default value was always used even if it was overridden in the configuration file.
- In the run-time environment, after invoking the `TTCN EncDec::set error behavior()` function with ET ALL option, the `TTCN EncDec::get last error type()` returned a wrong result unless you cleared it with TTCN `EncDec::clear error()`.

4.42. Version 1.6.pl0

Released on April 1, 2005.

New features

- The compiler supports semantic checking for the dynamic parts of TTCN–3 modules. This means that TTCN–3 modules are entirely covered with semantic analysis.
- The compiler generates the entire C++ code from the output of semantic analysis, the so-called Abstract Syntax Tree (AST).
- The TTCN–3 parser of the compiler supports error recovery, that is, it does not stop when a syntax error is detected, but it tries to read the further parts of the input and perform semantic analysis as well. However, in some cases it is not possible or worth-while to recover from a syntax error.
- The text encoding has been introduced.
- New RAW encoding attribute `REPEATABLE` has been added and the meaning of the `PRESENCE` attribute has been extended.
- The `AssignmentList` notation also can be used in the parameter redirect of `getcall` and `getreply` operations.
- Non-printable control characters of charstring values are printed using C escape sequences or

quadruple notation into the log files. The raw format of earlier versions could disturb the log processing utilities.

- In addition to the quadruple notation all escape sequences that are available in the C language are accepted for specifying non-printable characters of charstring and universal charstring values in the run-time configuration file.
- The order of steps that are performed when deactivating a TTCN–3 port at component termination have been changed. Now the existing connections and mappings are shut down first and after that the port is stopped and its queue is cleared. In some Test Ports it can happen that the user `unmap` function tries to add a new incoming message into the port queue when destroying the mapping. In former versions this could cause dynamic test case error because the port was already in stopped state while user `unmap` was running. Now the incoming message is accepted and the port is stopped only after user `unmap` is completed.
- As a non-standard extension the compiler and the run-time environment supports the assignment of names to the test components in create operations. The location of the new component can also be specified at creation, which is useful in distributed test environments.
- Script `ttcn3 start` has been enhanced. It supports configuration files that are named differently than the ETS and the execution of multiple test cases given in command line.
- The number of how many times the selected test(s) should run can be set on the GUI when selecting which test to run.
- Input field added to the GUI main window. It is enabled when a process runs. Any input entered here is sent to the process as user input.
- The GUI does not allow the creation or saving of empty test sets.
- The GUI asks whether to recreate symlinks and recreate the `Makefile` if appropriate changes are made on the project's properties.
- If creating new file through the GUI, and not specifying file extension in the file name selection dialog, the first extension listed in extensions for that type will be appended to the given file name.

Fixed bugs

- The compiler aborted with an internal fatal error during code generation if an ASN.1 value assignment contained a `ValueFromObject` reference.
- In some cases the compiler aborted with an internal fatal error when checking modified templates for record of or set of types. This was the case, for instance, if the base template contained a generic wildcard like ? or *.
- The compiler generated invalid (uncompilable) C++ code for `getcall` operations if the corresponding signature was imported from another module. This was because the default arguments of the C++ function that handles `param` redirects for the respective signature were generated into the source file instead of the header file and thus remained invisible for other modules.
- The code generator of the compiler ignored the `NotUsedSymbol` characters in `param` redirect of `getcall` and `getreply` operations if the `VariableList` notation was used. Thus the run-time environment assigned the wrong parameters to the given variables.

- The compiler ignored and did not check the value list subtype constraints for TTCN–3 union types.
- The compiler aborted with an internal fatal error when the `-P` (top-level PDU) command-line switch was used. This was because the switch `-P` disabled the semantic checking of ASN.1 modules including their import lists. But when the given top-level PDU tried to refer to an imported symbol the symbol table for imported symbols was uninitialized.
- The compiler rejected the `call` operations that did not have response and exception handling part with an error message even if the corresponding signature was a nonblocking one (i.e. it was defined with the `noblock` keyword).
- The compiler generated invalid (uncompilable) C++ code for those in-line signature templates the signature of which had no parameters.
- The compiler did not generate the C++ equivalents of some data types thus the generated code was uncompilable because of missing classes if an ASN.1 PDU type was defined using an object set, which was imported from another ASN.1 module.
- All generated header files included `version.h` from the Base Library, which made it impossible to use the precompiled headers created by GCC 3.4 or above. The above file contains some preprocessor statements, which disable the processing of existing precompiled headers in further `#include` directives.
- The compiler did not accept the `NotUsedSymbol` in the default duration of timer arrays. The symbol means that the corresponding array element shall have no default duration.
- The compiler might generate invalid (uncompilable) code for the TTCN–3 functions that had signature template parameters. The C++ compiler complained about missing encode text and decode text functions, which are used when the corresponding function is being started as a PTC behavior.
- The compiler did not recognize properly the default syntax for ASN.1 objects. If an object class did not have user defined syntax the compiler rejected the valid object definitions with strange parse error messages.
- The semantic analyzer did not detect errors in procedure based port types. For example, a reference to a non-existent signature in the incoming or outgoing list resulted in an erroneous (uncompilable) output C++ code.
- Our run-time environment implements the TTCN–3 entity that runs the control part and the MTC in the same process. The lists of active timers and defaults were not separated between the two entities thus the testcase could access and modify the control part timers and defaults. For instance, the statement all `timer.stop` in the testcase stopped the active timers of the control part as well.
- It was impossible to specify value list type restrictions for TTCN–3 enumerated types because the compiler did not recognize the enumerated values within the subtype definition.
- The compiler accepted value range type restrictions for almost all built-in types although value ranges are allowed only for integer and float types.
- The predefined conversion functions str2int and str2float in the run-time environment accepted some invalid input strings. For instance if the numbers were followed by letters in the input string the numbers were converted and the remaining letters were silently ignored. Now these functions accept only valid inputs containing a valid integer or float value. An invalid input

causes dynamic testcase error.

- The compiler generated invalid (uncompilable) C++ code if the component reference in the `connect`, `disconnect`, `map` or `unmap` operation referred to an optional field of a record or set value because there was no conversion function to perform the implicit cast.
- The compiler aborted with a segmentation fault or similar error when trying to generate C++ code for the initial value of a component variable, which contained a `valueof` operation. Such constructs usually appear in the output of the TTCN–2 to TTCN–3 converter.
- The utility `ttcn3 logmerge` reported memory leaks when the operating system's limit for the maximum number of simultaneously open files was smaller than the number of input files. In this case the log merge utility has to create temporary files in order to perform the merge in several steps. When the temporary file was closed the internal memory structure associated to it was not deallocated. Apart from the warning this bug had no other side effects.
- The ASN.1 parser of the compiler did not recognize the EXPORTS ALL directive in the module and reported syntax error on it. This construct was introduced in the 2002 edition of the ASN.1 standard, its meaning is identical to the missing EXPORTS clause.
- The compiler rejected the expressions, in which a constant value was referenced with a non-constant index. The error message complained that an integer value was expected as index even if the referenced value was of type integer. Such constructs can be created, for instance, in the body of parameterized templates where the index value refers to a formal parameter of the template.
- The Main Controller entered an infinite loop and flooded the console with an endless sequence of error messages if the operating system's limit of simultaneously open files was reached. This could happen when accepting the control TCP connections of newly created test components during load testing. The problem occurred on Solaris platform only. The reason was that the error codes of operating system calls were retrieved in an improper way and MC did not recognize the above situation.
- If a literal value of TTCN–3 float type caused an overflow in the internal memory representation [3] the compiler stopped immediately with an internal fatal error during the parsing phase. Such situations are now handled properly and the compiler can continue its run after producing the appropriate error message.
- The compiler aborted with an internal fatal error during code generation when it encountered a reference pointing to an element of a string value within template bodies or in the initial value of component variables. The code generator tried to follow the type of the embedded value and it handled the string element index in the same way as the indices of array and record of types.
- The run-time environment did not handle the variable assignments for record of and set of types properly. If the newly assigned value had fewer elements than the previous value of the variable the extra elements at the end were not erased from the variable. For example, if a variable contained three elements and the newly assigned value had only two, the new value of the variable had incorrectly three elements: the first two from the new value and the third one from the old value. This problem was present regardless whether the embedded type was a built-in or a user defined type or the extra values were bound or not. The effects of this bug were the most strange if a component variable of the MTC was initialized in the component type definition and several test cases were executed after each other.

- The run-time environment caused dynamic test case error if an optional component of an ASN.1 `SEQUENCE` or `SET` type, the type of which was the ANY type, was assigned to an optional octetstring field of a TTCN–3 `record` or `set` type and the source field contained omit value. The appropriate C++ member function was missing and the C++ compiler performed an implicit conversion, which assumed that the value of the source field is present.
- In case of some valid, but sophisticated TTCN–3 or ASN.1 type recursion loops the compiler did not produce the equivalent C++ classes in the right bottom-up order. Thus the generated code was uncompilable because of forward referencing of incomplete types.
- The Main Controller could report memory leaks if the configuration file passed to it contained syntax errors.
- The TTCN–3 parser of the compiler could not handle references pointing to an element of a multi-dimensional array in timer and port operations. It reported syntax error if the reference had more than one array indices, although the input was valid. The situation was the same if a component reference was embedded into a structured type and the `start`, `running` or `done` operation referred to a field of a variable.
- The compiler aborted with an internal fatal error during semantic analysis if the base template of a modified union template did not contain a specific value (e.g. it was a generic wildcard like "?").
- The generated C++ code was uncompilable if an indexed element of a string value was assigned to a template. The C++ compiler complained about ambiguous function overloads in case of all string types (i.e. bitstring, hexstring, octetstring, charstring and universal charstring).
- The generated C++ code was uncompilable if an operand of a built-in TTCN–3 arithmetic, logical or string operation was a reference pointing to an optional field of a record or set value. The problem was present in case of all TTCN–3 operations that are mapped to infix C++ operators.
- The HC logs contained incorrect information about the exit status of the test component if the corresponding UNIX process was terminated by a signal. In this case the log file showed a successful (zero) exit status although the process was aborted by a fatal error. Now the HC distinguishes between normal and abnormal process termination and reports the signal number instead of exit status in the latter case.
- Fixed segmentation fault in RAW encoder. The fault occurs if the length field is an union field and it stores the length of optional fields.

4.43. Version 1.5.pl8

Released on November 5, 2004.

New features

- The `Makefile` generated by the compiler cuts the `.exe` suffix from the name of the executable on Windows platforms when running the command `make archive`.
- The compiler uses more efficient internal data structures, which reduces its memory usage by about 10%.
- The verification algorithm of `USER` limited licenses became more robust. Formerly only a reverse lookup was performed based on the numerical user ID (UID) of the process and the license was

rejected if the UID was mapped to a login name different from that of the license file. Now, if this step fails a forward lookup is also performed on the login name of the license file. If the resulting UID matches the process UID the license is accepted. This makes the user name limited licensing working on UNIX systems where several login names are mapped to the same UID (e.g. portable Linux laptops with both local users and NIS membership). On such systems the reverse lookup returns one of the login names randomly.

- The compiler generates the equivalent C++ code of TTCN–3 templates based on the output of semantic analysis. This means the following enhancements:
- The compiler automatically re-arranges the initialization sequences for nonparameterized templates in case of forward referencing. It is not mandatory anymore to place such templates in bottom-up order in the module.
- If enum-hack is turned on the short form can be used for enumerated values within template bodies. The use of long form in functions, testcases and altsteps is still mandatory.
- The short-hand value list notation can be used in templates of record types.
- In-line compound expressions (including in-line modified templates) are supported as function or template actual parameters within template bodies.
- The run-time environment handles some test port errors in a more robust way. Formerly, if the `user map()` call failed (i.e. `TTCN error()` was called during its run) the test component registered the mapping before the error recovery. After the error recovery a regular unmap procedure was performed causing extra error messages.
- The run-time environment now supports the partial overwriting of generic wildcards ? and * in modified templates and template variables of structured TTCN–3 and ASN.1 types. When transforming a template from ? or * to a specific value the following rules are applied recursively:
- In case of record or set types all mandatory fields are set to ? and optional fields are set to *.
- In case of union types the selected field is set to ?.
- In case of record of or set of types all newly created elements up to and including the referred index are set to ?.
- The run-time environment executes the `EndTestCase` command, which is specified in section `[EXTERNAL COMMANDS]` of configuration file, after writing the verdict of test case into the log. Thus the external command can extract this verdict from the logand act accordingly.
- The C++ mapping of TTCN–3 and ASN.1 import statements has been improved. These updates make the generated code compatible with the limitations of GCC 3.4 or above that are in place when precompiled header files are used. The following changes were implemented:
- The #include directives are generated in the same order as the import statements are placed in the source module. This allows the explicit control which header file to be included first.
- Unnecessary #include directives are optimized out. For instance, if module C imports from both A and B and module B imports from A then `C.hh` will include only `B.hh` since `A.hh` is already included in `B.hh`.
- Base Library header files is not included directly if the module has imports.
- Circular import chains are handled in a special way.

- Logging of timers is now supported. That is, a reference to a timer can be an argument of a `TTCN-3 log()` statement. When executing the statement the attributes of the timer (state, default and actual duration, elapsed time) are printed into the log.
- If the executable tests are built with GCC the C++ compiler will perform extra checks on the arguments of Base Library functions that have printf-like syntax. The variable arguments of `TTCN error()`, `TTCN warning()`, `TTCN Logger::log()`, `TTCN Logger::log event()`, etc. are verified against the format string. New warning messages may be displayed during the compilation of Test Ports and external functions if type mismatches are found.
- Log Browser now is able to determine the component name from the log file name.
- Log Browser saves the configured view (columns shown) in the resource file.
- The Configuration Editor of the GUI has been rewritten. Now it also handles the comments in the configuration file. The comments should be above the assignments. See Part II of the TITAN TTCN–3 User Guide for further details.
- The GUI does not open Process output tabs for the external text editor, Log browser and the font / look and feel configuration programs.
- The icon of the TTCN–3 and ASN.1 modules has changed to be brighter.
- The working directory defined in the GUI project now automatically belongs to the log file directories.
- The compiled executable's path is now configurable for GUI projects.
- The Make archive item has been added to the GUI's project menu.
- If running a command in a dialog window in the GUI (ie View execution statistics in on the Execution window) the dialog can be closed by clicking on the **Close** button in the lower-right corner.
- The %selectedfile names has been added to the GUI variables. This enables to call the text editor for the symlinks in the working directory, and not to open the file referenced by the symlink. Using this variable in the text editor command makes `ctags` program work as intended.

Fixed bugs

- The compiler issued misleading warning messages when it found circular imports between the TTCN–3 and ASN.1 modules. Irrelevant module names were listed and the closing module of the loop was missing.
- Several faults were corrected in the TTCN–3 and ASN.1 object identifier value checking algorithm of the compiler. Referenced values within `NameAndNumberForm` were not verified properly thus faulty input could result in unpredictable behavior during code generation. Referenced values containing module name and module objid caused memory leaks. If a TTCN–3 subtype constraint referred to an `objid` value having a faulty component the compiler stopped with an internal fatal error.
- If a TTCN–3 or ASN.1 integer constant was so large that it did not fit in the internal memory representation of the compiler (actually 64-bit signed integers are used) the compiler stopped immediately with an internal fatal error during parsing. Such overflow situations now produce regular error messages and the compiler can continue analyzing the input.
- The compiler generated erroneous (uncompilable) C++ code for type alias definitions of port

types and signatures.

- The configuration file parser of the run-time environment accepted some invalid DNS names in section `[GROUPS]`. However, some valid names (e.g. names containing the dash character or having parts containing numbers only) were rejected.
- Circular recursions within parameterized template references could lead to infinite recursions in the compiler. A typical example for this situation: the body of a nonparameterized template refers to a parameterized template and one of the actual parameters is a reference to the non-parameterized template itself.
- In some very rare cases the compiler generated invalid C++ code for message based port types. Data was read from an uninitialized memory location and in case of a specific bit pattern the compiler called both the message based and the procedure based code generator for the same port type.
- The compiler generated wrong (uncompilable) C++ code if a TTCN–3 charstring value contained non-printable characters. Such characters were created during constant folding (e.g. the output of function `int2char`, which was evaluated by the compiler).
- Some compiler error messages reporting invalid RAW attributes printed memory garbage instead of TTCN–3 identifiers.
- The parser for the formal parameter lists of ASN.1 assignments was not implemented properly. It accepted all possible valid inputs, but some erroneous parameter lists as well.
- The compiler did not support the value list and value range subtype constraints for type universal charstring. Such constructs caused compilation errors.
- If a running operation was performed on an active, but already expired timer it disappeared from the list of active timers. After that the timer became invisible for operations `any timer.timeout` and all `timer.stop`.
- The RAW decoder did not handle the empty record of and set of values properly. If the octet stream contained no elements the corresponding value remained unbound after the decoding instead of containing an empty value (i.e. *{ }*).
- The GUI now shows a warning message if the symlink creation fails because a file named the same as the symlink exists in the working directory.

4.44. Version 1.5.pl7

Released on September 20, 2004.

New features

- The logformat utility no longer wraps the quadruples of universal charstring values into several lines. In previous versions it inserted a newline character after the commas like in case of compound (e.g. record or set) values. Now the valid quadruples are recognized and kept in the same line as the other parts of the string.
- The semantic analysis was improved for procedure signatures. The error messages related to parameters became more intuitive.
- The semantic checking of modified templates has been improved. For instance, if the derivation

path contained a circular recursion loop with 4 steps, the compiler printed the same error message 4 times after each other. The differences in the base and modified template's formal parameters are reported now in a more intuitive way.

- The generated `Makefiles` distinguish between different Solaris versions to make it easier to create portable test ports. This is necessary because some socket handling function calls use different types of arguments on Solaris 2.6 and Solaris 8 systems. Now the `Makefile` generator of the compiler automatically recognizes the operating system version and sets the variable `PLATFORM` accordingly to `SOLARIS` on SunOS 5.6 (Solaris 2.6) and to `SOLARIS8` on `SunOS` 5.7 (Solaris 2.7) or above.
- The logging of procedure based port operations has been improved. For instance, in previous versions the name of the signature was logged twice when transmitting calls or replies. In case of exceptions only the signature name was logged, the type of the exception itself was missing.
- The snapshot manager of the run-time environment has been improved. It does not call the registered event handlers anymore when taking the first snapshot before evaluating an alternative. This results in higher execution speed and fair scheduling when running performance tests. In a typical case one call of the event handler can insert hundreds of messages into the port queue while an alt statement extracts only the first one. With the improved snapshot manager the event handlers are not called again while the port queue is not empty. The new algorithm has a drawback that it cannot cope with the `[else]` branches. Thus when the first `[else]` branch is executed the algorithm switches back to the original compatibility mode.
- Log browser tool has been added as stand-alone tool to browse log files, or can be used in the GUI.
- GUI Type browser tool has been added.
- User menus and editor for the User menus have been added to the GUI. The editor can be accessed on the Preferences **window/User** menus tab.
- Preferences window's **General preferences** tab has been re-designed.
- New Help menu is added. It contains help index, contents, keyword search, license information and re-designed **About** dialog.
- The Process **output window** has been re-designed. Now it handles multiple running processes, each has its own output tab, and each can be stopped by the **Stop running** button.
- The Make, Clearcase and CVS menus have been rewritten for the GUI. Now they are configurable user menus.
- Test controls, test cases and test sets can be chosen and run by pressing the **Execute...** button in the Execution window.
- Statistics about the test execution can be viewed by using the View **execution statistics** toolbar button on the Execution window.
- The Execution window has been redesigned. Now it can be used both for single and parallel test execution. In parallel mode the default view now contains only the **Execute...** button. The detailed view can be switched on.
- Project properties window has been re-designed. General, log file, build and HC execution properties can be set on different tabs.

- The GUI supports starting multiple Host Controllers on different machines on the network.

Fixed bugs

- The compiler reported memory leak at termination if it encountered a `self.stop` or `mtc.stop` statement in its input module(s). A small memory block was not deallocated during the parsing of these constructs. This mistake had no other harmful side-effects.
- The compiler generated invalid C++ code if a TTCN–3 `activate` operation was used as a function statement. This construct was introduced in the most recent BNF and the newly written code generator forgot to put a semi-colon at the end of the equivalent C++ statement.
- If a TTCN–3 or ASN.1 module contained an enumerated type and it was imported into another TTCN–3 module using the undocumented `"light" extension` attribute the C++ equivalent of the importing module was uncompilable. This option disables the processing of some sections of the imported C++ header file from the importing module by using the C preprocessor to reduce the compilation time for large projects. The masked section contains mainly the C++ equivalent classes of data types and in this case some in-line enumerated conversion functions in another (active) section referred to class members, which were invisible from the C++ translation unit of the importing module.
- The compiler did not accept the hexadecimal string notation in ASN.1 modules for the values of the ANY type.
- The BER encoder did not detect if the octets of the value for an ASN.1 ANY type contained some extra bytes of garbage after a complete TLV. The accepted erroneous octet stream could cause failure during decoding.
- The behavior of the compiler was unpredictable when it processed asymmetric and non-internal procedure based port types. If the in and out lists contained different signatures the compiler usually crashed with a segmentation fault because the code generation routine tried to access the elements of the wrong list.
- In parallel mode the test components aborted with signal SIGPIPE (Broken pipe) if the `ConsoleMask` logging option was set to LOG ALL and the Main Controller has terminated unexpectedly (e.g. because Control-C was pressed on the command line). The components recognized the end of their control connection and therefore they tried to send a log message through the terminated TCP channel.
- The actual parameters of TTCN–3 entities were mapped to an improper data structure in the compiler's abstract syntax tree. The corresponding BNF production is called `TemplateInstance`, which consists of an optional type, an optional derived reference and a mandatory template body. For instance, in the former implementation the type part (if it was present) was excluded from the semantic analysis. At the same time the error messages that are used to report incorrect actual parameters were also improved.
- If the compiled Executable Test Suite was started with the command line switch `-l` (in order to obtain the list of test cases and control parts) and any internal error occurred during the initialization of the run-time environment, the ETS did not print anything and returned a successful (zero) exit status. Seeing this behavior it can be assumed that the modules contain no executable test cases or control parts. Now the ETS prints the error message to its `stderr` and returns a non-zero exit status.
- The handling of the list of active event handlers was implemented incorrectly in the snapshot

manager of the test components, which is a part of the Base Library. This could result in the unpredictable behavior of the test component in some race conditions. For example, if the termination of a port mapping was requested from a remote test component (by using an `unmap` operation) and at the same time data has arrived on the registered file descriptor of the corresponding Test Port, it could happen that the snapshot manager called the `Event Handler()` function of the Test Port after completing the `unmap` operation (i.e. calling user `unmap()`), which already performed the deactivation of the event handler.

- The compiler did not accept the valid null value as initial value for component variables with type default or a component type. It reported semantic errors for those definitions.
- The compiler aborted with an internal fatal error during error recovery (i.e. after reporting the error) if an invalid `SEQUENCE` notation was used for an ASN.1 REAL value.
- The generated C++ code could be uncompilable due to ambiguous function overloads in the class that implements TTCN–3 port types if the support of address type was turned on (i.e. the extension attribute `address` was used in the port type). The problem appeared only if the port type had outgoing messages or signatures and the address type was aliased to a built-in type the C++ equivalent class of which had a constructor with `int` parameter (e.g. `integer`, objid and all string types). The reason was a missing explicit cast in a generated function and the C++ compiler could not distinguish between component references and address values.
- The compiler produced invalid C++ code if the to clause was present in a TTCN–3 `raise` port operation. The generated code contained an unnecessary extra closing parenthesis.
- The semantic analyzer of the compiler did not report error if it encountered a length restriction or `ifpresent` matching mechanism in an actual value parameter of a template or function. It simply ignored the extra matching attributes and analyzed the specific value only. The generated C++ code was uncompilable in these cases.
- The special component references such as `null`, `mtc`, `system` and `self` were not handled in component `start` operations. Thus the run-time error messages were cryptic when the argument of start was one of the above.
- The entries of the alphabetical index in this document contained wrong page numbers. This was because the index was built before the table of contents and the pages occupied by the latter were neglected.
- The RAW type descriptor of charstring type was missing causing segmentation fault during RAW encoding/decoding.
- Utility `ttcn3 logmerge` printed garbage characters to the output if the component identifier part of input file name was longer than 30 characters. By default the component identifier is the numeric component reference, which cannot be so long, but in case of custom filename skeletons that part can contain any string.
- The semantic analyzer of the compiler could reject some valid TTCN–3 constant definitions by printing mysterious error messages. The problem occurred if a referenced value with field or array sub-references pointed to a sub-field of another referenced compound value.
- There were some minor mistakes in the semantic checking algorithms of value ranges within template bodies: The compiler did not check the upper bound if the lower bound was set to minus infinity. The `ifpresent` matching attribute of value ranges was reported twice if it was inappropriate.

- The ASN.1 parser of the compiler did not accept the construct `ObjectClassFieldType` in `FixedTypeValueFieldSpec`.
- The semantic analyzer of the compiler was unable to detect circular recursion loops in template references.
- Infinite type recursions (e.g. when a TTCN–3 record type contains itself as a mandatory field) caused stack overflow and segmentation fault in the compiler. Moreover, the compiler was unable to detect the similar embedded recursions within constants and templates of valid recursive types (e.g. when a field of a structured value refers back to the whole value).

4.45. Version 1.5.pl6

Released on June 11, 2004.

New features

- The compiler and the run-time environment supports the following non-standard TTCN–3 language extensions, which allow the dynamic creation of templates during test execution. In the future these constructs will be added to the TTCN–3 standard in the same or very similar form.
- Functions and external functions may return templates (i.e. all permitted matching mechanisms in addition to specific values) if the return type is defined as template.
- It is allowed to define template variables at any place where the definition of regular (value) variables is permitted. The value of such variables may be dynamically assigned and they can carry any permitted matching mechanism.
- It is allowed to pass template parameters to functions, `altsteps` and `testcases` by reference. Those parameters are denoted by the keywords `out` or `inout` similarly to value parameters. The actual parameters shall be template variables.
- The conversion functions related to enumerated types are no longer global C functions. They were moved to static members of the value class. This makes easier to write generic C++ template functions in Test Ports that can handle any enumerated type.
- The script `ttcn3 start` handles the configuration files automatically. If it finds a file named `<ETS name>.cfg` in its current working directory it passes the file to MC as a command line argument.
- New compiler flag `-B` added to generate browserdata.dat needed by the visual type browser (part of GUI).
- The RAW attribute `HEXORDER` is now applicable to the octetstring type besides hexstring type. The `HEXORDER(high)` settings will twist the nibbles within an octet during encoding and decoding.
- The new command `make check` was introduced in the `Makefile` generated by the compiler, which performs syntax and semantic checks on the TTCN–3 and ASN.1 modules.

Fixed bugs

- The Main Controller behaved incorrectly during the shutdown procedure after the `exit` command. There was an unhandled race condition between the internal threads, which had noticeable effects only on Linux. The MC sometimes reported memory leaks or segmentation

fault occurred or the final *Shutdown complete.* message was missing. On other platforms the problem remained hidden due to the different thread scheduling algorithms of the operating systems.

- The type descriptor needed for the RAW encoding of the boolean type was missing from the run-time environment. This resulted in segmentation fault if a message containing a boolean field was encoded or decoded.
- The compiler accepted an invalid syntax for catching timeout on procedure based ports. The string `catch timeout` was interpreted as a `catch` operation in addition to the standard `PortReference.catch(timeout)` notation. Moreover, the standard syntax produced invalid C++ output, only the irregular variant resulted in working code. Only the standard syntax is accepted from now, and it is translated to a valid C++ code.
- The compiler returned a misleading successful (zero) exit status if it encountered a general problem with an input file (e.g. the file could not be opened or classified as a TTCN–3 or ASN.1 module).
- The Main Controller did not set the close-on-exec flag on its socket file descriptors that were used for communication with the other processes of the test system (HCs and test components). As a result of this all commands launched from the user interface (e.g. the text editor started from the GUI) inherited the open files. If the process did not finish before the end of the test session the MC could not close the TCP connections properly.

4.46. Version 1.5.pl5

Released on April 30, 2004.

New features

- The compiler is able to generate `Makefiles` to be used in single mode with the newly introduced command line switch `-s`.
- The `Makefile` generator of the compiler recognizes if a given C++ source file is generated from one of the TTCN–3 or ASN.1 modules. In this case a warning is issued and the given file is not added to the list of USER SOURCES.
- The compiler became less strict regarding the usage of semi-colons within `TTCN□3modulepar` blocks. The compiler treats the semi-colons optional, but it issues a warning if the input does not conform to the standard TTCN–3 BNF.
- If the Test Port writer tries to register the event handler on a bad (e.g. nonexistent) file descriptor, the run-time environment reports this error immediately when Install Handler member function is called. In earlier versions the error was detected later by the select system call when taking a new snapshot. It was more difficult to determine the origin of the error if more than one Test Ports were used in the same test component.
- The logging of location information has been improved. The line numbers are more accurate (e.g. when a TTCN–3 function is called from a send or receive template). It is possible to log the entire call stack and the name of the current TTCN–3 function, testcase or altstep.
- The compiler prints correct location information for errors and warnings found in ASN.1 `OBJECT IDENTIFIER`, Relative-OID and TTCN–3 `objid` values. The error recovery is also supported in these

values. In earlier versions the compiler stopped after finding the first error in these values.

Fixed bugs

- The compiler terminated with a segmentation fault during error recovery if it encountered more than one opening brackets (*{*) without closing pairs in ASN.1 modules.
- The compiler aborted with an internal fatal error if a referenced value or template of wrong type was used as a template for an enumerated type. The error occurred when the compiler tried to print the 'type mismatch' error message.
- The compiler aborted with an internal fatal error instead of printing the appropriate error message if the empty record/set value (i.e. *{ }*) was given as a TTCN–3 template for a union type. If the union template contains (incorrectly) more than one selected fields, the compiler checks now every field template against the corresponding field type after printing the error message.
- If the compiler generated a `Makefile` to be used with GNU make and all user source files had the `.cc` suffix, but one or more of them did not have its own header file, the suffix substitution rule was not used to obtain the list of object files. In this case only the user header files have to be explicitly enumerated, the names of object files do not need to be enumerated.
- The `sizeof` built-in function could not be used on optional record or set fields (or their ASN.1 equivalents) if the type of the field was a `record of` or `set of` type. The generated C++ code did not compile in these cases, the compiler complained about no matching function.
- If the body of a parameterized template used a formal parameter with field or array sub-references, the compiler evaluated the type of the reference expression incorrectly and thus reported type mismatches on valid templates. For instance, if the type of a formal parameter was a record with an integer field then the compiler assumed that the type of the formal parameter reference with the field sub-reference is the record type and not integer.
- The compiler reported fake circular value recursion if an expression within an array index contained a nested array reference. The reason was that the names of some nodes in the compiler's internal syntax tree were not set and the recursion detection algorithm saw empty strings instead of the real names.
- The ASN.1 parser of the compiler did not consider the default tagging method correctly. If an embedded block (e.g. a `CHOICE` type) contained tagged types, the compiler decided whether to use implicit or explicit tags based on the default tagging method of the lastly parsed ASN.1 module instead of the current module. The problem was visible only if the input ASN.1 modules did not use the same default tagging method, which happens rarely.
- The type descriptor structures in the Base Library were not declared for TTCN–3 type universal charstring. This caused compilation errors in the generated C++ code if the universal charstring type was embedded into a record, set or union type. The problem did not affect the ASN.1 character string types.
- The precedence requirements for RAW attributes `PRESENCE` and `CROSSTAG` were not checked at compilation. Such kinds of errors caused dynamic test case errors (i.e. when the field that determines the presence or selection of another field is encoded at a later position in the message than the field it points to). These problems are now detected by the compiler and no C++ code is generated for faulty types.
- The utility `ttcn3 logformat` did not interpret the command line switch `-s` properly (i.e. it did not

split up the log of each test case into separate files) if the input log file contained location information. The logformat utility supports now also the newly introduced location information formats (stack traces, etc.).

- Some weird and recursive use of `COMPONENTS OF` construct in ASN.1 modules could cause the unpredictable behavior of the compiler (segmentation fault, bus error, etc.).
- The compiler aborted with an internal fatal error during semantic analysis in the following case: The operand of an expression in a template body or component type definition referred to a module parameter or a formal parameter of the template, that is, the value of it could not be evaluated at compilation. Moreover, the type of the corresponding module parameter or formal parameter was a referenced type pointing to another referenced type (i.e. the chain of type references contained at least two elements).
- The compiler aborted with an internal fatal error during semantic analysis if an expression of type universal charstring contained references that cannot be evaluated at compilation time (e.g. references pointing to module parameters or formal parameters of the template).
- The Main Controller reported memory leak at exiting if its connection towards MTC had terminated unexpectedly (e.g. due to the crash of MTC).
- The compiler reported memory leak if a string element was referred in an operand of an expression.
- The compiler generated wrong type descriptors for BER decoding of an ASN.1 `SEQUENCE OF` or `SET OF` types. The user-defined tags of the enclosed type were ignored during decoding.

4.47. Version 1.5.pl4

Released on March 19, 2004.

New features

- The variables and templates of some types in the run-time environment occupy less memory than before.
- The expect script `ttcn3 start` launches the Main Controller from `$TTCN3 DIR/bin` if the environment variable is set.

Fixed bugs

- The internal encoder of the run-time environment stuck in an infinite loop when it tried to encode integer value -2147483648. This could occur when the above value was sent over connected ports or passed to functions to be started on PTCs. * The run-time environment crashed with a dynamic test case error when if a template was initialized with an omitted optional field containing an optional ASN.1 `NULL` type.
- The compiler generated implicit tags instead of explicit ones when "Tag Type" was used for a "DummyReference" and the referenced type was not `CHOICE` or open type. (For details, see Information Technology, Abstract Syntax Notation One (ASN.1): Specification of basic notation clause 30.6c.)
- The ASN.1 parser did not accept the {} value for `SEQUENCE` and `SET` types.

- The mapping of some ASN.1 character string types were wrong: TeletexString, VideotexString, GraphicString and GeneralString must be mapped to universal charstring (and not charstring) according to Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3. Part 7: Using ASN.1 with TTCN–3.
- The semantic checker accepted (and checked) named numbers for ASN.1 integers in TTCN–3 templates (but the generated code was invalid).
- The RAW coding of empty record/set types went wrong in version 1.5.pl3 (the object files could not be linked because of missing member functions).
- The RAW decoding of octetstring values that are not beginning at octet boundary was erroneous (no error message but incorrect value).
- The Main Controller caused memory corruption (which could result in segmentation fault, bus error or other unpredictable behavior) if its control connection towards MTC terminated unexpectedly during the testcase. The MC tried to read and modify some internal memory areas during the error handling procedure that were previously deallocated.

4.48. Version 1.5.pl3

Released on February 27, 2004.

New features

- The built-in TTCN–3 type universal charstring and ASN.1 character string types with multiple-byte character sets are now supported.
- The unrestricted character string type and selection types are now supported by the ASN.1 front-end of the compiler.
- The RAW encoder/decoder allows to use referenced values in attributes `PRESENCE`, `TAG` and `CROSSTAG`. The references shall point to constant definitions of the right type. In earlier versions only literal values were accepted in RAW attributes.
- If a value of a set of type does not match the respective template the run-time environment produces more verbose results in the log. Formerly only the nonmatching template and value were logged as a whole. In addition to this the newly implemented heuristic algorithm prints the following:
- Each element of the value (including its index) that has no matching pair in the template.
- Each element of the template (including its index) that has no matching pair in the value.
- Every matching value *$* template index pairs.

NOTE Although it is trivial to show a possible successful matching, it is very complicated to develop an exact algorithm to present the reason of mismatch. It can be theoretically proven that the matching fails if and only if there exists a subset of values (or templates, symmetrically) with k elements, the elements of which has less than k pairs in the template (or value) altogether. The exact algorithm should show the minimal ones of these sets with the respective pairs. Our heuristics present only those trivial subsets that have $k = 1$ elements, which give useful hints in the majority of practical cases.

- The static run-time memory usage of TTCN–3 port instances was significantly reduced (from about 1000 bytes per port to 60 bytes). The majority of this memory block contained file descriptor bit masks that were moved to the dynamically allocated heap area. This is an important improvement since all UNIX processes that implement TTCN–3 test components contain all port instances of all component types defined in the test suite. In complex test setups (with many component types) the majority of port instances are inactive since they belong to other component types. On operating systems that do not support the copy-on-write mechanism each inactive port instance on every test component used about one kilobyte of memory unnecessarily.
- The first line of all log files contain the version number of the TTCN–3 Test Executor in both single and parallel modes.
- The predefined conversion functions int2char, char2int and str2int behave according to the new (still unpublished) TTCN–3 semantics. That is, they cause a dynamic test case error if their output is invalid instead of returning a dummy result. These changes were necessary to be consistent with the newly implemented int2unichar and unichar2int functions.
- Value range templates are now supported for both charstring and universal charstring types. The upper and lower bound of such ranges shall be one character long string values since they denote the smallest and largest character positions of the permitted character set.
- Some anachronistic TTCN–3 syntax variants were removed from the compiler: The to keyword cannot be used in range definitions either in subtype constraints or in template bodies instead of two dots anymore.
- The compiler did not allow negative integer values to be assigned to TTCN–3 enumerated values due to a BNF bug in the standard. The negative numbers are now accepted just like in ASN.1 (or the embedded ASN.1 tables of converted TTCN–2 test suites).
- The operation `mtc.stop` is now supported. If it is executed on a PTC the execution of the current test case (including all active PTCs) will be interrupted immediately. The execution continues with the next test case or the next statement of the control part.

Fixed bugs

- If the `-E` (enum-hack) switch was used the compiler generated invalid C++ code for those ASN.1 enumerated values that are keywords in both TTCN–3 and C++ languages (e.g. true).
- If an ASN.1 `SEQUENCE`, `SET` or `CHOICE` type definition contained a syntax error the error recovery routines of the compiler did not work properly. Thus the compiler could fail with internal error messages or produce memory leaks later during the semantic analysis.
- The infinite type recursion detection algorithm of the compiler could cause internal errors on some kinds of recursive type definitions. The algorithm was re-designed so that it is started only from top-level types. In addition to solving the above problem this makes the compiler a bit faster.
- If a field of an ASN.1 object class had a default setting the parser did not allow the field to be omitted in object definitions.
- Some mis-encoded data streams could cause segmentation faults in the BER decoder. If the length of some strings was encoded in a faulty way the decoder interpreted it as a very large number and tried to allocate a too large buffer for the value.

- In some cases the run-time environment called the `Event Handler` of a Test Port unnecessarily twice within the same snapshot. This might happen if the first call of `Event Handler` used the `Uninstall Handler` and `Install Handler` primitives after each other. At re-installation the event handler was inserted at the end of active event handlers' list thus the list iterator reached it again. Now the run-time environment keeps track of each event handler whether it was called in the current snapshot.
- If a TTCN–3 string element was assigned to the owner string the assigned value might be incorrect. The problem occurred with all string types: bitstring, hexstring, octetstring and charstring. The reason was the faulty implementation of the respective operators in the run-time environment. The operators deallocated the original string's memory area first and took the value of string element after that.
- If a RAW encoding attribute of a TTCN–3 type contained a syntax error the file name in the compiler error message pointed to the lastly parsed TTCN–3 file instead of the faulty one. Only the line number information was correct. Moreover, the error message did not contain the reason of the error.
- The compiler generated invalid C++ code if the `PRESENCE` RAW attributes contained nested field references.
- If a TTCN–3 default reference was logged, which has already deactivated, the `log()` statement could have unpredictable results (e.g. segmentation fault, irrelevant printout, etc.)
- If a parameterized ASN.1 object assignment was referenced without an actual parameter list the compiler crashed with a segmentation fault after printing the relevant error message.
- If an ASN.1 object assignment has referenced itself recursively the recursion detection routine of the compiler turned into an infinite recursion and printed endless error messages.
- If the component type initialization failed on a PTC (because, for instance, the initial value of a component variable was unbound) its control connection towards MC was closed unexpectedly. This was reported on the MC console, but no error message refer to the component type. Now the PTCs do graceful termination even in such cases.

4.49. Version 1.5.pl2

Released on January 23, 2004.

New features

- The TTCN–3 parser of the compiler no longer recognizes the identifier parameters as a keyword. This is an obsolete variant of keyword `modulepar`, which was used in earlier versions of the TTCN–3 standard.
- The elements of bitstring, hexstring, octetstring and charstring values, which can be accessed using an array-like syntax, can now be logged by TTCN–3 `log()` statements. In previous versions the `log()` statement printed nothing for string element arguments.
- The compiler now gives warning messages if there are circular import chains at module level, e.g. module A imports things from module B, module B imports another things from module C and module C imports something from module A. This is legal in both ASN.1 and TTCN–3 , but the generated C++ code might be uncompilable.

- `GeneralizedTime` and `UTCTime` have been prefixed with ASN in the library because also OSS library uses these identifiers. This prefix is consistent with some other type names (e.g. `ASN NULL`).
- The behavior has changed to greedy while parsing identifiers in objects if there is a *{* after the identifier. This is because that identifier can be a parameterized reference. The old behavior was to leave the block for the next element in the object class syntax because that can be object set or value set.

Fixed bugs

- The compiler issued irrelevant error messages for modified templates if the base template had formal parameter list. The error message complained about missing actual parameter lists, but the actual parameters of the base template shall not be specified according to TTCN–3 syntax.
- The compiler aborted with an internal fatal error if the base template reference of a modified template pointed to a definition other than a template. • The compiler generated invalid C++ code for some ASN.1 constructs if the enumhack (`-E`) option was used. If an ASN.1 type contained an embedded (unnamed) ENUMERATED type the C++ `enum` values were incorrectly prefixed in value assignments for such types.
- The compiler crashed with a segmentation fault if an actual parameter of an altstep invocation contained an in-line compound expression. The C++ mapping of such constructs is still unsolved due to the lack of full semantic analysis. The compiler now continues the parsing and substitutes the unsupported parameter with the comment /* NOT SUPPORTED */ in the output code, which will cause a C++ compilation error.
- The compiler generated erroneous C++ code for certain legal TTCN–3 constructs. Let us assume that the name P was used to identify a component variable in component type A. If a function, altstep or testcase had a runs on clause other than A and a formal parameter named `P` the resulting C++ code did not compile if parameter `P` was used in the function body. The reason is that the TTCN–3 component type scoping is solved using tricky C preprocessor macros in the generated code. Due to the wrong placement of preprocessor statements different name substitution rules were applied on formal parameter list and function body.
- If the user `unmap` function of a Test Port called Install Handler the event handler could remain active after the deactivation of the port. This could result in unexpected Event Handler calls and warnings when the test component terminated.
- The functions of TTCN `logger` caused segmentation fault if one tried to create log entries when the logging was already shut down at test component termination. These late log events are now silently ignored by the logger.
- The RAW decoding of record with optional elements may fail if TAG attribute used to identify the optional elements.
- Fixed the incorrect RAW encoding of HEXSTRING if HEXORDER(high) was specified and the field is started at the 4th bit of the octet.
- The internal hostid calculation algorithm worked unstable on Windows XP. The hostid of a given computer could change between successive reboots, which made host licensing impossible on computers running XP. The algorithm included a registry key in the hostid calculation that could be modified at system startup. This problem did not appear on other Windows versions such as 2000, NT or 9X.

- There was a fatal error if a module imported a symbol with the same name more than twice.

4.50. Version 1.5.pl1

Released on December 19, 2003.

New features

- TTCN–3 external constants and external functions are now included in semantic analysis, which means that they can be referred from template bodies.
- The Main Controller supports the constraints about the location of newly created PTCs. This means the sections `[GROUPS]` and `[COMPONENTS]` of the configuration file are properly interpreted. The load balancing is done only within a subset of hosts (the so-called candidates) that fulfill the constraints set in the above two sections.
- Type recursion loops that are terminated with optional record or set fields are now handled properly. For instance, it is now possible to define a record type, which contains itself as an optional field. Such types are useful to implement the TTCN–3 equivalents of linked list containers, where omit denotes the end of list. In former versions this construct caused the compiler to enter an infinite recursion loop, which finally resulted in a segmentation fault.
- The run-time environment provides a more efficient implementation for local TTCN–3 port connections. If the two endpoints are located on the same test component a software loop is used between the two port objects instead of a TCP connection. This results in faster connection establishment or termination and more efficient data transfer through the connection.
- As a special case of local port connections it is now supported that both endpoints of the connection is the same port. Although this is allowed by the TTCN–3 standard, the establishment of such looped connections failed with mysterious run-time errors in previous versions.
- The `connect` and `disconnect` TTCN–3 operations are now supported in single mode as well. Of course, both endpoints of the port connections must reside on the MTC in single mode.
- The run-time environment is now able to terminate those PTCs that entered an infinite loop without any receiving operation (e.g. by doing an infinite calculation). If a PTC does not respond to a stop request within a configurable amount of time the Main Controller instructs the Host Controller of the corresponding host to kill the uncontrollable UNIX process.
- The ? and * wildcards in bitstring, hexstring and octetstring templates are accepted and the correct matching of those patterns is implemented.
- The pattern construct in charstring templates and the additional predefined function `regexp` are now implemented. This means that the matching of TTCN–3 regular expressions and substring extraction based on them are now supported.
- The ASN.1 types `EXTERNAL` and `EMBEDDED-PDV` are now supported including their BER encoding.
- Associated SEQUENCE notation for REAL values in ASN.1 is now supported.
- Metacharacter substitution was introduced in log file names, which means that the names of the log files are determined during execution. At the same time the log file naming convention became configurable in parallel mode as well.
- Enhanced error recovery during ASN.1-parsing.

- The RAW encoder/decoder contains significant improvements.

Fixed bugs

- The member function `cut()` of class TTCN Buffer in the run-time environment caused damages to the remaining contents of the buffer because of an incorrect memory handling technique. For instance, if the buffer contained two correct BER-encoded messages after each other then after decoding and cutting the first message the decoding of second one failed.
- If the execution of the `[EXECUTE]` section was launched in parallel mode, that is, the `smtc` command was invoked without arguments the Main Controller did not give the prompt back until the execution is completely finished. This made it impossible, for instance, to monitor the actual test configuration (using the info command) or interrupt the execution (using the `stop` command) at any time.
- The load balancing algorithm in the Main Controller did not count the MTC when choosing a location for a newly created PTC. For instance, if a test configuration, which required 3 PTCs, was distributed on two hosts it happened that the first host ran 2 PTCs in addition to the MTC while the second host had only one PTC.
- The compiler generated invalid C++ code for TTCN–3 constants that referred to a sub-field of another constant (or ASN.1 value), which was imported from another module.
- Use of TTCN–3 subtype constraints (e.g. value lists or length restrictions) caused memory leakage in the compiler if it was invoked with the `-p` (parse only) option. The memory responsibilities were incorrectly assigned in the syntax tree if the semantic analysis was bypassed.
- The parse errors in ASN.1 modules (or embedded blocks within ASN.1 definitions) caused memory leaks within the compiler.
- Compound templates (both with value list and assignment notation) were accepted by the compiler for objid and ASN.1 NULL types. The C++ code generated from such invalid input was also erroneous, of course.
- When a modified template was written for a union type, which contained an embedded record the compiler did not allow missing fields for the inner record even if the same alternative of the union was selected as in the base template.
- References to parameterized TTCN–3 entities (i.e. templates or functions) were accepted by the compiler without actual parameter lists in template bodies.
- If a statement block within a TTCN–3 altstep contained local variable definitions the generated C++ code did not compile because of a missing pair of brackets.
- Circular references within TTCN–3 constant expressions (that is, if the right-handside value of a constant definition referred back to the same constant) caused the compiler to abort after reporting the relevant error message.

4.51. Version 1.5.pl0

Released on October 31, 2003.

New features

- The compiler supports the semantic analysis of TTCN–3 templates including the parameterized and modified ones. Signatures and signature templates are also checked.
- The constant, variable, port and timer declarations within TTCN–3 component types are now checked.
- The run-time environment has a new option in the configuration file to append log files instead of overwriting.
- The evaluation order of activated defaults has been reversed to be compliant with an accepted CR to TTCN–3 operational semantics. That is, the lastly activated default is tried first when matching snapshot in alt statements or stand-alone receiving operations.
- In ASN.1 Component Relation Constraint, you can use multi-level parent reference (See Information Technology, Abstract Syntax Notation One (ASN.1): Constraint specification X.682 clause 10.10b). If the Component Relation Constraint is broken, the decoder logs the value(s) of the constraining component(s).
- The BER coder has many enhancements. The sorting of SET and SET OF components (needed by CER and DER) are done. `SET` values can be decoded regardless of the order of components. `REAL` decoding is supported (only base 10).

Fixed bugs

- When a TTCN–3 enumerated type was forward referenced within a module the compiler aborted with an internal fatal error if the enum hack (`-E`) option was used.
- The compiler generated incorrect C++ code for some procedure-based port types. This caused memory corruption when an incoming call, reply or exception was extracted from the port queue, which could result in unpredictable behavior (segmentation faults, infinite loops, etc.) on some platforms.
- When an individual testcase was executed in parallel mode (i.e. without the control part) the name of the module and testcase were swapped in the log (like this: `Executing test case MyModule in module MyTestCase.`).
- The additional predefined function `int2hex` produced incorrect results. Every second hexadecimal digit of the output was zero regardless the input.
- The concatenation operator (`&`) did not work properly for bitstring values if the length of the left operand was not a multiple of eight. The last few bits of the result contained memory garbage instead of the correct value.
- If the Main Controller encountered a socket error in one of the control connections (e.g. a broken TCP link because of a network failure) it entered an infinite loop and flooded the console with error messages.
- The log filter utility performed seek operations on its input file. Thus it could handle only regular files properly as input and produced incorrect (truncated) results if its input was a UNIX pipe.
- The select system call could fail on Solaris with an Invalid argument error code in the snapshot handler of the run-time environment when extremely large timer durations were used[4]. The reason of the failure is a limitation in the select implementation of the operating system because it does not accept timeout values that are larger than an undocumented limit. Now the

timeout values are truncated to a safe limit (which is about 26 days), that is, the test components never block for longer time than this value. In case of truncation a warning message is issued. If nothing happened within this time interval the process will wake up and block again.

- The test execution could stop with strange internal error messages in parallel mode if guard timers were used in execute statements. This happened if the MTC was performing a configuration operation (e.g.`create` or `connect`) at the moment when the guard timer expired. This situation is now explicitly handled in the internal control protocols, which start an error recovery procedure to interrupt the current test case and continue with the control part.

4.52. Version 1.4.pl5

Released on October 2, 2003.

Fixed bugs

The ASN.1 front-end of the compiler crashed with an Internal Error in some cases when checking the tags of open types. For instance, this made impossible the parsing of MAP protocol specification.

4.53. Version 1.4.pl4

Released on September 19, 2003.

New features

- The generated C++ code no longer uses template classes for the realization of enumerated, record of and set of types and their ASN.1 equivalents. This results in significantly faster compilation of the generated code especially in case of large projects. The generated flat code provides the same (or backward compatible) API for the Test Ports.
- The compiler supports semantic analysis and automatic ordering for TTCN–3 module (global) constants. The semantic checks include the folding of expressions, which results in more compact and faster C++ code. The component constants and local constants are still unchecked.
- The compiler error messages include location information (file name and line number) for the faulty language elements. This helps a lot in finding the faults in TTCN–3 and ASN.1 modules.
- The compiler supports error recovery, that is, it does not stop after the first error message, but it goes further to find more errors. At the same time it implements advanced error masking techniques to avoid snowball effect.
- Signatures, procedure based ports and related port operations (i.e.`call`, `getcall`, `reply`, `getreply`, `raise`, `catch`) are now supported. The in-line signature templates are still not supported like the in-line message templates.
- It is now possible to combine value lists and value ranges in templates for integer and float types.
- It is now possible to give a `match` operation as an argument to the TTCN–3 log statements. This logs the matching process field-by-field (like in case of failed receive statements) instead of the boolean result.
- The following new Main Controller commands were introduced: `info`, `stop`, `pause`, `continue`, `log`.

See Section 12.3.1 of the TITAN TTCN–3 User Guide for details.

Fixed bugs

- The compiler no longer gets confused during `Makefile` generation if it finds a file with a name identical to a TTCN–3 or ASN.1 module. An executable program, which was built with a previous `Makefile`, can have such name.
- The run-time environment now detects if the file descriptor returned by the operating system is larger than the system limit `FD_SETSIZE` [5]. The internal event handler mechanism of the test components uses the *select* system call to handle messages on port connections. Only the file descriptors that are smaller than FD SETSIZE can be used with *select*, so if a newly created socket file descriptor reaches this limit a proper error message is printed instead of the unpredictable behavior of former releases.
- The run-time environment put misleading location information into the log if a TTCN–3 function was called from a non-parameterized template. The location info pointed to the last line of the called function.
- The logging of template matching (which is done at failed receive events if the logging of event `TTCN MATCHING` is enabled) did not work properly for record of and set of types and their ASN.1 equivalents. The algorithm did not compare the sub-fields of the elements even if the template had the same number of elements. Now the field-by-field comparison is done, but only if the template and the received value has the same number of elements.
- The compiler did not translate the TTCN–3 identifiers properly if they ended up by more than one underscore character. For example, 2 underscores at the end became 3 underscores in C++ instead of 4.
- The error messages given by the compiler when detecting syntax errors in the RAW encoding attributes referred to the wrong file. The error messages always contained the file name of the lastly parsed TTCN–3 module, but the line numbers were correct.
- The template matching algorithm for the set of type construct did not work properly in some rare cases. Sometimes it did not find the right pairs when sophisticated graphs were needed for this, thus it returned false instead of true.
- The >> (shift right) operator produced invalid results for some bitstring and hexstring values. If the string operand was longer than 1 byte (8 bits or 2 nibbles, respectively) the last few bits or nibbles of the result could contain memory garbage instead of the correct value.
- TTCN–3 identifiers `stdin`, `stdout` and `stderr` are handled by the compiler as if they were C++ keywords to avoid interferences with the libc macros.

4.54. Version 1.4.pl3

Released on July 23, 2003.

New features

- The compiler has a new option, which allows to parse its input modules without performing semantic analysis or code generation. Consequently, the command line switch `-s` has been introduced and the meaning of `-p` changed. Now `-p` means parsing only and `-s` includes

semantic checks as well.

- The size of the C++ classes that the compiler generates for record/set/union types or their ASN.1 equivalents was reduced by about 10 %. This results in smaller executables and faster compilation (especially in case of incremental builds).
- The compiler is now capable of selective code generation when doing incremental builds. It means that after performing a relatively fast parsing and semantic analysis on all modules the C++ code is generated only for those modules that have changed since the last build or import from changed modules. This feature can substantially reduce the time needed for an incremental compilation in case of large projects. Of course, the capability of selective updating has been preserved.
- The interrupt signal (which can be raised, for example, by pressing Control-C while the executable tests are running) is now handled explicitly in single mode. If the signal is received the run-time environment tries to clean up all resources (destroys all existing port mappings, calls the destructors of global objects, etc.) before terminating instead of exiting immediately. This was necessary for implementing test ports on the top of some poorly designed APIs, which require the application level connections to be terminated explicitly (although the communication is carried by TCP).
- The Main Controller now supports batch mode execution if the variable `NumHCs` is set in the section `[MAIN CONTROLLER]` of the configuration file.

Fixed bugs

- The values of empty record/set types (e.g. type record `MyRecord` *{ }* were always logged as *{ }*, even if the value was unbound. This could result in misleading behavior during test port development.
- The RAW decoder decoded the fixed length bitstring fields with incorrect length if the `FIELDLENGTH` value was greater than 8 and the field did not start on octet boundary.
- The accuracy of internal encoding mechanisms that are used for the transmission of TTCN–3 float and ASN.1 REAL values has been enhanced. Formerly the encoding considered 6 decimal digits only, which was too few in some cases. Now the encoded value can carry up to 16 decimal digits which provides lossless transmission for the 48-bit mantissa of 64-bit floating point values.
- The run-time environment could produce misleading error messages in parallel mode at initialization. If the initialization of non-parameterized templates failed (e.g. due to an unspecified field or forward referencing) the error message that was displayed on MC console complained about an error in the configuration file.
- The `start` and `stop` port operations are now implemented exactly according to the TTCN–3 semantics. That is, not the `stop` but the `start` operation clears the remaining messages from the queue.
- The Main Controller did not give the prompt back if the `smtc` command was issued without arguments until all items of the `[EXECUTE]` section have been executed.

4.55. Version 1.4.pl2

Released on June 30, 2003.

New features

- Value returning by started PTC functions and storing the returned values in `done` operations are now supported.
- Improved argument processing (file and module auto-detection) when generating `Makefile` skeletons.
- Support of obsolete TTCN–3 language constructs (such as named alts, old-style notation for imports and module parameters) were removed from the compiler. Consequently the old keywords are no longer reserved.

Fixed bugs

- There was a memory corruption bug in the ASN.1 front-end of the compiler, which could result in invalid C++ identifiers in the generated code. The phenomenon was libc dependent and appeared only on some Linux platforms.
- The Main Controller did not stop if the given configuration file did not exist or contained syntax errors. It assumed that the configuration file is empty in this cases.
- The handling of configuration file errors in parallel mode was improved. That is, no deadlock occurs in this case.
- The make archive rule of the generated `Makefile` passed incorrect arguments to tar.
- Handling of ObjectFieldSetting and ObjectSetFieldSetting.
- Chains in InformationFromObject(s), ObjectSetFromObjects.
- Circular references through InformationFromObjects construct caused segfault in some rare cases.
- If `-P` was used, some assignments were missing from the generated code that should not.
- The Main Controller on Linux stopped with an error message indicating a failure in `poll()` system call when its X terminal window was resized.
- The ASN.1 front-end of the compiler did not check the existence of imported definitions if they were not used. For example, if module A imports X from module B, and module B exports all (or simply doesn't have an exports statement), then it wasn't checked that the module B really has an assignment or imported symbol with name X. So, when X was missing, but wasn't used in module A, there was no error message.

4.56. Version 1.4.pl1

Released on June 13, 2003.

New features

- Hexstring related additional predefined (conversion) functions were implemented. Namely:

`hex2int`, `int2hex`, `hex2str`, `str2hex`, `bit2hex`, `hex2bit`, `hex2oct`, `oct2hex` and `substr` with hexstring as first argument.

- The obsolete TTCN–3 type char was removed from the run-time environment (including the Test Port API). For backward compatibility all occurrences of char is substituted with type charstring.
- The compiler command line switch `-P` was introduced to specify top-level PDUs in order to disable the code generation for unreachable data types.
- Anachronistic timer duration units are no longer supported. Therefore the words `min`, `s`, `ms`, etc. can now be used as regular identifiers.

Fixed bugs

- Sub-fields of optional record/set fields can now be directly referenced in TTCN–3 expressions and assignments.
- The template matching mechanism for the set of type construct (which uses sophisticated graph-pairing algorithms) could stuck at infinite loops in some cases due to an improper variable initialization. The infinite loop included memory allocation thus the memory consumption of the executable grew rapidly after reaching this deadlock situation.
- The `Makefile` generated by the compiler behaved incorrectly in case of incremental compilation because of a missing empty rule. The build process stopped immediately after the translation of TTCN–3 and ASN.1 modules and the C++ compiler was not invoked. The `make` command had to be issued again for the complete build. Moreover, in case of the non-GNU version the build procedure did not stop if the TTCN–3 /ASN.1 compiler returned unsuccessful exit status.
- The `-E` (enum-hack) option generated invalid C++ identifiers if the enumeration was a C/C++ keyword (e.g.`class`).
- The default syntax (i.e. when `WITH SYNTAX` is not defined) of object classes was implemented incorrectly.

4.57. Version 1.4.pl0

Released on June 4, 2003.

New features

- One new, integrated compiler for TTCN–3 and ASN.1.
- X.681-X.683 extensions of ASN.1 are now supported by the compiler.
- Semantic analysis and automatic ordering for TTCN–3 data types.
- TTCN–3 altsteps containing only an [else] branch are now accepted by the compiler as a language extension.
- The PDF version of this User Documentation contains hyperlinks to the referenced sections.
- The new structure for the generated `Makefile` eliminates the unnecessarily repeated compiler invocations when doing incremental builds.

Fixed bugs

- TTCN–3 predefined functions are distinguished from other user defined functions during parsing. As a consequence of this the number of arguments is checked at compilation and the statement `mytimer.start(int2float(5))`; is no longer interpreted as a component start operation.
- The license verification procedure caused segmentation fault in all programs if the real UID of the program had no associated login name. This could happen on Cygwin or if the system administrator did some evil things. A proper error message is printed now in this case.
- The RAW encoder/decoder functions were not generated for some structured TTCN–3 types due to a human mistake in the compiler source code.
- TTCN–3 `execute` statements could not be used as expressions, meaning the final verdict of the corresponding testcase. This construct is now supported even if the testcase has a maximal duration.
- Test cases can no longer be called as simple functions, only the execute statement is allowed.
- An improper memory initialization in the templates of TTCN–3 type hexstring could cause segmentation faults in the run-time environment.
- The member function user start of Test Ports is called implicitly by the run-timeenvironment when a testcase is started. If a fatal error occurred in this function (which was signaled with TTCN error) the error recovery routines have interrupted the execution immediately. Moreover, the error verdict was not counted in the final statistics line. Now in this situation only the corresponding testcase is aborted with error verdict and the execution will continue with the next one.
- When execution was started in parallel mode and the MC detected a version or module checksum mismatch for one of the HCs, the error message appeared only in the log file of the corresponding HC. Now the error message given by the MC (indicating the reason of the failure) is printed to the standard error of the HC as well.
- The expect script called **`ttcn3 start`** (see Section 12.3.4 of the TITAN TTCN–3 User Guide for details) was missing from the binary packages and the scripts delivered with versions 1.2.plx are not compatible with the command line interface of the new MC. A new version of the script was added to the packages. This seems to be more robust when handling console messages coming from the HC or TCs because the former one could cause deadlocks if the messages arrived too frequently.
- When the `-l` command line switch was used with the ETS (in order to obtain the list of test cases and control parts) and the initialization of constants has failed the program exited with abort signal abnormally in both single and parallel modes. The exception caused by this fatal error is now caught and the program terminates with non-zero exit status after printing the error message.
- If the initialization of constants or templates has failed at test startup the error message could include an inappropriate operating system error message. This was because some library calls have set the global variable `errno` to an inappropriate error code even if the library call was successful. As a work-around the `errno` variable is explicitly set to zero after such calls.
- If an empty message (empty string) was logged from the test suite the logger signaled a fatal error message and exited immediately.

4.58. Version 1.3.pl0

Released on April 18, 2003.

New features

- The new MC supports the execution of individual test cases (without control part) in parallel mode as well.
- The new MC supports the automatic distribution of the configuration file to multiple HCs over the network.
- There are no longer static limits on the number of simultaneously active PTCs. The new MC is able to handle any number of PTCs if the limit on simultaneously open files is set to large enough in the operating system.
- The new MC explicitly terminates the remained active PTCs at the end of each test case.
- The checksums in the ETS are checked by the MC to avoid launching of inconsistent HCs in case of distributed execution.
- The configuration operations in parallel mode are executed significantly faster because the Nagle algorithm is switched off on the control TCP connections.
- The internal messages of the TTCN–3 run-time environment (control messages or messages sent on connected TTCN–3 ports) use a more compact encoding for integer attributes. The size of typical messages sent over TCP channels was reduced by 50%.
- TTCN–3 `action` statements are now implemented. The argument is placed to the log with a dedicated severity. Similarly to the log statements it is allowed to pass not only fixed strings, but variables or templates to the action statements. Multiple arguments (separated by commas) are also supported.
- Compound expressions are now allowed in TTCN–3 return statements.
- Statement blocks following altstep instances are now accepted by the TTCN–3 compiler.
- The special TTCN–3 `address` type is now supported (i.e. the user can define it). Address values can be used in port operations (in to or from clauses or in sender redirects) only if the corresponding port type has a special extension attribute.
- Parameterized modified templates may have more parameters than the base template, even if the base template has no parameters. Additional parameters can be appended to the parameter list anywhere in the modification chain.
- The TTCN–3 configuration operations `create`, `stop`, `connect`, `disconnect`, `map` and `unmap` produce more verbose information in the log files of all components that are involved in the operation.
- The option `-e` (enum-hack) was introduced in the ASN.1 compiler to resolve name clashes between different enumerated types. Using this the ENUMERATED values seem like this: TYPENAME enum ENUMID.
- The ASN.1 compiler translates large ASN.1 modules significantly faster than in the previous version due to the improved type checking algorithms.
- The statistics line at the end of execution contains more information than before (e.g. the percentage of passed, failed, etc. test cases). The statistics information is also available in

parallel mode. It is logged when the MTC terminates.

- More robust error checking is performed in the run-time environment during logging. If writing to the log file fails at any time the execution will terminate immediately with a proper error message.

Fixed bugs

- The incorrect behavior of the old MC caused deadlocks if a `connect` or `map` operation was performed on an already established connection or mapping, or similarly if the `disconnect` or `unmap` operation referred to a non-existent connection or mapping. The situation was the same if a `stop` operation was performed on the component reference of an already terminated PTC. The new MC is robust enough to handle such situations.
- The TTCN–3 `done` and `running` operations with `any component` or all component were incorrectly implemented in the old MC.
- The calculation of test case verdicts was implemented in a non-standard way in the former versions (i.e. a successful `done` operation meant an implicit `setverdict`). Now the MTC receives and processes the final verdicts of all PTCs at the end of each test case in compliance with the standard.
- The non-standard predefined function `float2str` used exponential notation if the argument was zero thus it resulted 0.000000e+00 instead of 0.000000.
- The snapshot manager of the run-time environment now signals a dynamic testcase error if it has to block for infinite time. In former versions the execution stopped forever without any warning or error message. This situation can happen only in single mode when there are neither active timers nor active event handlers.
- If an inactive timer was stopped an inappropriate duration (i.e. a memory garbage) was printed to the log after the warning message. Now only the warning is displayed in such cases.
- Negative timer durations were not handled in the previous versions of the run-time environment. A negative default value for a timer produces now a warning. If a timer is started with a negative duration (which can be either explicit or the default one) a dynamic testcase error will now happen.
- The incoming messages had incorrect line information in the log file if they were received during the evaluation of an alt statement. Those messages always pointed to the last branch, which could be in a default as well. Now the line information is set to the first line of the alt statement.
- The TTCN–3 behavior statement `self.stop` was refused by the compiler, although it is allowed in the standard text. The reason for this was a bug in the official TTCN–3 BNF that disallows `self.stop`. The compiler now accepts `self.stop`, which has identical meaning as stop.
- The TTCN–3 compiler generated invalid C++ code if an altstep was instantiated from the top-level alternative of another altstep in combination with a boolean guard expression. A closing bracket was missing in this case.
- The ASN.1 compiler produced an inappropriate error message if in a module with AUTOMATIC TAGS a CHOICE type was included in a SEQUENCE with tagging.
- The ASN.1 tag descriptor structures in the run-time environment were initialized in wrong order. Therefore dynamic testcase errors could happen during encoding/decoding with the

error message ”The innermost tag is not explicit”.

- Some parts of the C++ header file generated by the ASN.1 compiler was reordered to resolve some import related problems.
- In parallel mode the log files are always placed in the current working directory of the HC even if the executable is started with a full pathname. In former versions if the HC was started with a pathname the logs where placed to the directory where the HC resided. Moreover, on Windows platforms the `.exe` suffix of the HC executables is also cut from the names of the log files.
- The `make archive` command of the generated `Makefile` follows symbolic links when creating backups. If the TTCN–3 and/or ASN.1 modules are linked from another directory, the real files are stored in the backup instead of the meaningless symbolic link.
- The identifiers that are reserved for internal purposes in the Base Library and do not contain underscore characters (such as `INTEGER`, `OCTETSTRING` or `PORT`) could not be used as TTCN–3 identifiers because of name clashes. The conflict is now resolved by appending a single underscore at the end of such identifiers thus they are freely usable in TTCN–3 modules.

4.59. Version 1.2.pl4

Released on February 10, 2003.

New features

- Host limited licensing are now supported on Windows platforms as well.
- New non-standard conversion functions float2str and str2float were introduced.
- The internal control protocols of the parallel test architecture were updated. The updates, which are the first steps in the migration toward version 1.3, are not backward compatible. Therefore the ETSes built with 1.2.pl4 must use the Main Controller of version 1.2.pl4 only. Similarly, the Main Controller of 1.2.pl4 does not work with ETSes built with earlier versions. Using incompatible versions may result in run-time errors or deadlocks.
- The TTCN–3 compiler emits the repeated string (bitstring, hexstring, octetstring, charstring) and object identifier literals into the generated C++ source file only once for each module. This can reduce the size of binary object code by about 20-30% for modules containing mainly TTCN–3 templates.
- All ASN.1 identifiers are printed in TTCN–3 form into the log file (underscores are used instead of hyphenation characters).

Fixed bugs

- The RAW encoder and decoder did not handle the `integer` fields properly if FIELDLENGTH was not a multiply of 8. For instance, 7-bit fields were mis-interpreted during both encoding and decoding.
- The RAW decoder has left record of and set of fields unbound if no elements were received.
- If a PTC has terminated with an error verdict a successful `done` operation on it caused Dynamic Testcase Error on the parent component.
- Inappropriate source code information was reported at the end of PTC logs if the `LogSourceInfo`

logging option was turned on.

- The ETS could die with a segmentation fault during error recovery if the assignment of union, record of or set of value or template fields failed because of unbound sub-fields.
- Float values that are either smaller than 10_−_4 or larger than 1010 in absolute value are logged in exponential notation. In previous versions all float values were logged in decimal dot notation so the small float values seemed as 0.000000.
- The internal handling of TTCN–3 port mappings could cause the unpredictable behavior of the ETS on some platform/compiler combinations (e.g. on Debian Linux with GCC 2.95.4) because of memory corruption due to incorrect memory allocation methods.
- If more than one empty record or set type was defined as incoming or outgoing type on a TTCN–3 port type the send or receive operations with in-line templates (like `EmptyRecordType`: *{}*) resulted in erroneous C++ code. The remained limitation is that the type name must be always present for in-line empty record templates even if the type is unambiguous.
- The caching of status values in stand alone receiving statements (such as `receive`, `timeout` or `done` operations) was incorrectly implemented. If an activated default has returned with a repeat statement the ETS could hang in a live-lock or report a dynamic testcase error incorrectly.
- The run-time configuration file parser did not accept ASN.1 identifiers (enumerated values, field names, etc.) that contained hyphenation character. Such identifiers can now be used in the configuration file either in the original ASN.1 form or in TTCN–3 form (i.e. the hyphenation characters are replaced by underscores). Both formats are equivalent.
- The TTCN–3 compiler generated invalid or incorrectly working C++ code if a variable of a set type was initialized at the definition and the fields were not in the same order as in the type definition.

4.60. Version 1.2.pl3

Released on December 16, 2002.

New features

- TTCN–3 templates, even parameterized ones, can be printed to the log using `log(…)` statements like constants, variables. All wildcard combinations are handled properly.
- TTCN–3 functions that have template parameters can be started on PTCs using component `start` operations.
- The `hexstring` data type is now supported. Unfortunately the RAW encoder/decoder still does not work for `hexstring` fields.
- If a `receive` port operation fails because the incoming message does not match the given template, the matching process can be logged field-by-field to make it easier to find the difference. This option is switched on if the event type `TTCN MATCHING` is enabled in logging filters.
- Location information (that is, the name of the source file and line number) can be included in the log file for each TTCN–3 test event.

- The TTCN–3 compiler accepts the latest published BNF (Rev. 12.7, v2.2.1). Of course, it is backward compatible and still accepts the obsolete keywords and constructs, such as named alts.
- The cross-referencing between different kinds of TTCN–3 language elements was clarified. At the same time a two-phase module initialization scheme was implemented. The first phase is performed before, the second one is after processing the configuration file. At the same time the syntax of initializer functions for importable modules has changed.

Fixed bugs

- The precedence between TTCN–3 operators was revised. In former 1.2.plx versions the parser worked exactly as specified in ETSI's TTCN–3 BNF. However, that BNF is wrong (it contradicts the standard text), because – for example – it assigns the same precedence level to operators `and`, `xor` and `or`. Now our parser works as it is specified in the standard text (i.e. as an average user expects).
- The ETS crashed with segmentation fault if a TTCN–3 function with runs on clause was called directly from control part. A proper error message is printed now.
- The compiler and the run-time environment accepted invalid range templates (i.e. if the upper bound was smaller than the lower) for integer or float types. The run-time environment prints now a proper error message during initialization.
- Templates of built-in type charstring could not be initialized with string literals containing exactly one character. The compilation of the generated code failed because of a missing conversion operator.
- The tag in the output of BER encoder could contain invalid primitive/constructed indicator bit in some rare cases. If an ASN.1 CHOICE type contained a built-in type (e.g. OCTET STRING) as field with implicit tagging, the encoder set the indicator bit to 1 (constructed) in the tag, but the value was a single octet string.
- The RAW encoder did not handle properly the pointers to omitted optional fields. In this case the pointer must be set to 0, but it pointed to the next field.
- The error and warning messages displayed during the processing of the with attributes of TTCN–3 data types for RAW encoding contained an invisible carriage return (CR) character at the beginning (like other error messages, which were fixed in the previous version).
- The log formatter utility did not place a newline character into the formatted output when an opening bracket character (*{*) was followed immediately by a string literal (for example, when logging a `record of charstring` value).

4.61. Version 1.2.pl2

Released on October 11, 2002.

New features

- The internal string handling routines were rewritten in both TTCN–3 and ASN.1 compilers. The old functions were inefficient, especially when generating large C++ output files. The improved TTCN–3 compiler runs significantly (about 4 times, the ASN.1 compiler 2 times) faster on a

typical input module.

- The C++ header and source files generated from TTCN–3 and ASN.1 type definitions were restructured so that the header files became smaller. This means faster compilation (with lower memory usage) from C++ to object module in case of the importing modules of those type definitions.
- The error messages related to templates became more talkative for record/set/SEQUENCE and union/CHOICE types. This helps locating the error when debugging faulty template definitions. The type and field names are incorporated into the error strings.
- The compiler is able to generate a smaller and less redundant `Makefile` to be used with GNU make.
- The compiler incorporates source file and line number information into the dynamic testcase error messages of alt statements and stand-alone receiving operations.
- The missing documentation of RAW and BER PDU encoders/decoders was added.
- Two new log processing tools: `logmerge` and `logfilter`. See sections 13.1 and 13.2 of the TITAN TTCN–3 User Guide.
- The ETSes can print the list of the test cases and modules.

Fixed bugs

- One header file (Message `types.hh`) was missing from the binary distribution. This caused C++ compilation errors for startable TTCN–3 functions.
- The `stop component` operation caused idle PTCs (i.e. PTCs that were just created and no `start` operation was performed on them) to abort with a core dump because of an uncaught C++ exception.
- The `create`, `done`, component `start`, `stop` and `running` operations report proper error messages in single mode.
- Operation any `component.done` caused a segmentation fault if no done operations were performed on that component before.
- The executable test programs (both in single and parallel mode) caused segmentationfaults during the logging of the first message if the opening of log file was not successful. Now, a proper error message is printed and the test executor process exits immediately after a log file opening failure.
- The ASN.1 `NULL` values were not interpreted properly by the TTCN–3 compiler.
- Component `stop` operation hangs if it is performed on a PTC that is already terminated. This is due to bug in the Main Controller, which will be fixed later. A work-around for this was added to the Base Library: the `stop` operation returns immediately if the TC knows locally the termination of the target PTC (for example, if the `stop` operation is used subsequently after a done for the same component).
- The error messages of the TTCN–3 compiler contained an invisible carriage return (CR) character at the beginning. This disturbed, for instance, Emacs when finding the location of the error.
- The year and date were in the reverse order in the timestamps of log files when `TimeStampFormat`

was set to `DateTime`. For example, 28/Sep/2002 was printed instead of 2002/Sep/28, as specified in this document.

- Character strings that resemble to a format string of C function `printf` (containing %d %s, etc.) no longer cause problems in TTCN–3 log statements.
- A `charstring` value that contained only one character (which can also be considered as a char value) could not be used as a template for receive operations on ports those incoming type was charstring.

4.62. Version 1.2.pl1

Released on August 16, 2002.

New features

- The `set of` type construct is now supported, including template matching.
- The component type scoping units are supported. Different component types may have identically named ports, timers, constants and variables.
- The runs on clause is properly interpreted for test cases, functions and altsteps. Component instances (ports, variables, etc.) are accessible only if the proper runs on clause is present. If a function or altstep is called on a test component, the component type is checked against the runs on clause. Component type mismatches cause dynamic testcase error.
- The chapter about the usage of the ASN.1 compiler was added to this document.
- System related Test Port parameters were introduced.
- Event severity TTCN DEBUG was introduced in logger.
- String literals (constants) are translated to static C++ objects, that is, they are listed in the source file only. In previous version when a string constant was added to or removed from the source TTCN–3 module, the compiler re-generated the C++ header file as well and therefore it was necessary to re-compile several C++ files.

Fixed bugs

- The HTML report generator `repgen` started `logformat` with invalid command line arguments.
- If some modules of the ETS were compiled on Solaris 8 while others on Solaris 2.6 (but using the same GCC 3.0.3), it might have happened that the functions of RAW enc/dec were unable to catch an internal exception and therefore the TC was killed by an abort signal. The reason of this problem might be the incompatibilities in the system header files of the two operating system. However, making the copy constructor of the exception class trivial has solved the problem.
- The compiler generated invalid C++ code for nested altsteps (i.e. when an altstep was instantiated from another one).
- The TTCN–3 standard specifies that if the action list of a default terminates without reaching a stop or repeat statement, the test executor has to skip the alt statement or receiving operation that the default was called from and jump to the next statement. In the previous version, however, this situation was handled incorrectly and resulted in a dynamic test case error.

4.63. Version 1.2.pl0

Released on July 29, 2002.

New features

- Importing from ASN.1 modules and BER encoding/decoding of ASN.1 types are now supported.
- Direct (RAW) encoding and decoding of TTCN–3 data types are now supported.
- The compiler accepts the latest available BNF of TTCN–3 (V2.2.0, Rev 12.5). The obsolete language elements that were removed from Edition 2 of TTCN–3 (e.g. named alts, old style import statements and module parameters, `verdict.set` and `verdict.get` operations, `goto` alt statements, etc.) are still accepted with warnings to provide backward compatibility.
- Altsteps and (dynamic) defaults are now supported.
- The internal handling of basic TTCN–3 string types (bitstring, octetstring, charstring) has been changed to a reference counter based method. This means that memory allocation and copying is no longer necessary for value assignments or parameter passing. This change in combination with the load-time initialization of string literals can result in 50-100% improvement of overall execution performance, especially in case of test suites dealing with long string values.
- The extra matching attributes (i.e.`ifpresent` or `length` matches) can be used for compound templates as well. Because of a mistake, the TTCN–3 BNF allows them only with single values.
- New command line options of the compiler: `-n`, `-o`, `-p`, `-u` and `-w`. See the TITAN TTCN–3 Programmer´s Technical Reference for details.
- The format of timestamps in log file can be configured. The event type names can also be logged for each event. These options are useful for log post-processing.
- The numeric values are assigned properly (i.e. as it is described in the Edition 2 standard) to enumerated values that have no assigned value.
- The object identifier type of TTCN–3 (i.e. objid) is properly supported, it is no longer an alias to charstring.
- The TTCN–3 compiler has a nice manual page.
- The `Makefile` generation was extended to support ASN.1 modules.
- Some additional predefined functions were renamed according to the latest TTCN–3 specification. Non-standard predefined functions `string2bit` and `string2oct` were renamed to `str2bit` and `str2oct`, respectively. For backward compatibility built-in functions can still be referred using old names as well.
- The generation and use of empty test port skeletons can be omitted for port types that are used only for internal communication between test components. If the with attribute extension `internal` is appended to the port type definition, all necessary code will be included in the C++ output files of the module. This feature reduces the compilation time and the total size of ETS.
- A positive integer identifier number is assigned to all incoming messages when the message is appended to the port queue. The extraction of messages (e.g. in case of a successful `receive` operation) is also logged with a reference to these identifiers. This addition may help the test writers to trace continuously the actual state of port queues when debugging TTCN–3 code.

- The `setverdict` operation displays the old and the new value of the local verdict in the log.
- A new `expect` script makes the testing in parallel mode easier. See Section `expectscript`.

Fixed bugs

- The line numbering (i.e. `-l`) compiler option generated invalid C++ code for the named alternatives.
- The line continuation backslashes in the generated `Makefile` caused syntax errors with some non-GNU versions of make.
- The TTCN–3 `connect` operation sometimes failed and caused a run-time error on Windows platforms. The reason was a small difference between the native UNIX and Cygwin socket APIs. Namely, under Cygwin a *bind()* operation must be always performed before calling *listen()*, even if listening to an automatically assigned ephemeral TCP port.
- The execution of MTC failed in a select system call if some port connections of MTC remained active at the end of the previous test case.
- The NotUsedSymbol (-) is now properly interpreted in array or record of values both in TTCN–3 modules and in the configuration file of module parameters.
- The `map` and `unmap` port operations did not work in single mode. Moreover, `connect` and `disconnect` operations produce proper error messages in single mode.
- Escape sequences in single character string literals (e.g. in a charstring containing only a quotation mark character) resulted in invalid C++ code.
- All ports of test components are started implicitly according to the TTCN–3 specification immediately when the component is created. The explicit start statements are no longer necessary.

4.64. Version 1.1.pl10

Released on February 11 2002.

New features

- The char built-in type is now supported, including the conversion functions `int2char` and `char2int`.
- Bitwise operators `not4b`, `and4b`, `or4b` and `xor4b` are implemented for types bitstring and octetstring.
- Bitwise shifting and rotating operators are implemented for all string types and integers.
- Length restrictions are supported in string templates.
- The template attribute `ifpresent` is supported for optional record/set fields.
- An HTML report generator was added to the official package. See Section 13.4 of the TITAN TTCN–3 User Guide for details.

Fixed bugs

- A UNIX signal can no longer cause dynamic testcase error when the ETS is waiting for a new

snapshot. This could happen in some conditions during the profiling of ETS.

- The assignment operator did not work properly for bitstring elements.
- The size of equivalent C++ code was reduced by about 10 % in case of compound TTCN–3 data types because of revised template realizations.
- The Main Controller no longer creates the log file Parallel MC log, which contained debug messages only.
- The built-in TTCN–3 type verdicttype was not recognized by the compiler. The compiler supported the older syntax that called this type as verdict.
- The compiler generated invalid C++ code if you used an enumerated type in a record of type construct (within the same module). The resulting C++ definitions were in the wrong order.
- The octetstring values that contained ASCII control characters (e.g. newlines or tabulators) were incorrectly logged in ASCII format.

4.65. Version 1.1.pl9

Released on December 21 2001.

New features

- Indexing (i.e. accessing of individual bits) is now supported for type bitstring as well.
- The `Makefile` generated by the compiler was significantly improved. For example, you can easily archive your source files with time stamping.
- It is possible to execute external programs (shell scripts) by the ETS at the beginning or end of each test case or control part.
- TTCN–3 constants and module parameters cannot be modified from TTCN–3 code. Such attempts result in erroneous C++ code.
- The compiler inserts source code information (i.e. the name and line number of TTCN–3 source code) as comments into the equivalent C++ code of TTCN–3 functions, test cases and control parts. This feature hopefully can help when locating syntax errors in TTCN–3 code. It still does not work for other definitions (such as constants or templates), but it will be extended in future versions.
- The `-l` command line switch instructs the compiler to include this line information as #line directives instead of C++ comments. Using this the error messages of the C++ compiler will point to the lines of the original TTCN–3 source code. However, be extremely careful with this option because sometimes the error messages can refer to invalid line numbers. In such cases turn this switch off and analyze the C++ code manually.
- The TTCN–3 `rem` operator is now supported. In the Test Port API it is mapped to global function `rem`, which takes two arguments, either `INTEGER` or `int` in any combinations.
- Non-standard conversion operations `string2bit` and `string2oct` were introduced.
- Default values for module parameters are supported.
- The module name can also be specified when setting module parameters in the configuration file.

Fixed bugs

- There were linking problems with the OpenSSL shared library on some Solaris systems. In the meantime the downloadable packages were updated, so these problems should disappear using the current 1.1.pl8 version.
- The compiler generated invalid C++ code for the initializers of constant and variable arrays.
- The string concatenation operation in Test Port API was changed from '&' to '+' for all string types. The `&` operator will be used for the TTCN–3 and4b operator in future versions.
- In the Test Port API the TTCN–3 `mod` operator is no longer mapped to the C++ operator `%` because they have different semantics in case of negative operands. Instead the global function `mod` was introduced, which takes two arguments, either `INTEGER` or `int` in any combinations.
- It was impossible to set negative integer numbers as module parameters in the configuration file.
- Some symbolic constants worked incorrectly when setting the log filtering bitmasks in the configuration file.
- Logging can be completely disabled by using the filtering mask value `LOG NOTHING`.

4.66. Version 1.1.pl8

Released on November 7 2001.

New features

- All parts of the test executor system can only be used with a valid license key.
- The input language of the parser complies with the latest official BNF of TTCN–3 (Version 1.1.2, published on June 2001).
- With attributes do not cause parse errors, but they are not interpreted by the compiler.
- Modified templates are now supported. The only limitation is that a modified parameterized template must have the same formal parameter list as its base template.
- Explicit type casting for generic wildcard templates in receiving operations is now supported. For example, `MyPCO.receive(ICONreq:?)` is accepted.

Fixed bugs

- The else branch was executed incorrectly in alternatives. Formerly, it was executed only if the guard operations failed for all other branches. Now, the else branch is executed immediately if none of the other branches are successful for the first try (according to the TTCN–3 operational semantics).
- Predefined functions `oct2int` and `bit2int` returned wrong integer values, because they interpreted the octets or bits in the wrong order.
- Invalid C++ code was generated for TTCN–3 functions that had formal port parameters but were otherwise startable on PTCs.

4.67. Version 1.1.pl7

Released on October 8 2001.

New features

- The *any value or none* (*) wildcard is now matched correctly in templates of record of types. Formerly the * was interpreted as *any value* (?), that is, it matched exactly one element.
- The memory handling of record of type construct was improved.
- External TTCN–3 constants and functions are now supported. They are translated to external C++ constant definitions or function prototypes in the target header file.
- The compiler does not stop when it encounters an unsupported type definitions, instead it outputs a warning message. Value list and length restrictions are simply ignored while the set of type construct is currently substituted with record of.
- The unsupported `sut.action` statements are skipped by the compiler.

Fixed bugs

- Overlapped component creation (initiated from different TCs) sometimes caused segmentation fault in Host Controllers.
- Performing a running operation on an already terminated component sometimes resulted in dynamic test case error instead of returning false.
- Passing of simple value parameters (i.e. without `tt in`, `out` or `inout` keywords) in TTCN–3 functions resulted in invalid C++ code in some cases. Now these parameters are passed by value in case of data types and by reference in case of port types.
- Concatenation (`&`) operators were not implemented for string elements returned by indexing.
- Expressions were not allowed in timer start operations.

NOTE	Simple function instances (i.e. their return values) are still not allowed in timer start operations, because the compiler cannot distinguish `MyTimer.start(MyFunction())` from `MyComponent.start(MyFunction())`, but they should be handled in different ways. In such cases please use `MyTimer.start(MyFunction() + 0.0)` as a workaround, which yields always valid C++ code.

- Value list and complemented list templates for record of types resulted in invalid C++ code.

4.68. Version 1.1.pl6

Released on September 26 2001.

Fixed bugs

- The `running` timer operation returned always true for a started timer even if it has already expired.
- The `enumerated` module parameters remained unbound even if they were set correctly in the

configuration file.

- Test execution terminated abnormally in parallel mode if the `disconnect` port operation was performed on a component other than the connection was requested from by a `connect` operation.
- The compiler generated invalid C++ code for some templates that contain specific values for enumerated types.

4.69. Version 1.1.pl5

Released on September 17 2001.

New features

- The compiler does not overwrite the target C++ header or source file if its content does not change. This can speed up incremental compilation because the header files usually do not change if the user only modifies a TTCN–3 definition and therefore only one C++ module shall be re-compiled. This feature can be disabled by passing the `-f` switch to the compiler.
- Non-standard conversion functions `oct2char` and `char2oct` were introduced.

Fixed bugs

- The return statements in TTCN–3 functions accept expressions without parentheses as well.
- A reference to a non-existent test case in the `[EXECUTE]` section of the configuration file did not cause any error message.
- Translation of named alts having parameters caused memory leakage in the compiler.
- Translation of TTCN–3 functions having port parameters resulted in invalid C++ code.

4.70. Version 1.1.pl4

- Released on August 27 2001.

New features

- The stand-alone instances (i.e. like a function call, not within an alt statement) of a named alt within one function are no longer limited. (It is a good exercise to understand how it works with the C preprocessor.)

Fixed bugs

- Log file naming discrepancies were fixed in configuration file.
- Function `main()` was missing from `libttcn3-dynamic.so`.
- The compiler reported memory leakage or sometimes crashed with segmentation fault after translating an array of constants.
- Logging did not work properly when Base Library was linked dynamically.
- There were compilation problems with Base Library on FreeBSD/NetBSD because of missing

included header files.

- Comparison of charstrings with NULL pointer no longer causes segmentation fault. The NULL pointer is interpreted as an empty string.
- The `trigger` port operation did not work correctly, because it dropped all nonmatching messages for the first try. TTCN–3 operational semantics says to drop only one message in one round (i.e. it introduces an implicit receive).
- Port operations any `port.receive`, any `port.trigger` and any `port.check` are now translated correctly.
- The compiler no longer generates code with quadratic size for incoming queue handling of ports.

4.71. Version 1.1.pl3

Released on August 1 2001.

New features

- Run-time support of TTCN–3 module parameters. They are read by the test executor from a configuration file.
- Individual execution of test cases that have no parameters (independently from module control part).
- Test port parameters.
- Additional predefined functions (except hexstring related ones) specified in TTCN–3 standard are now supported.
- Pre-defined TTCN–3 functions (except char, universal char and hexstring related ones) defined in Annex C of Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3. Part 1: Core Language are now supported.
- The enumerated types are provided with a *value) string* and *string) value* conversion functions.

Fixed bugs

- Some C++ templates of Base Library generated erroneous C++ code.
- The bug in Main Controller that caused some map operations to fail was fixed.
- C/C++ keywords that are not keywords in TTCN–3 are now translated to valid C++ identifiers, i.e. the compiler appends a trailing underscore character to them.
- C style comments longer than 16 kbytes in TTCN–3 modules caused the compiler to fail during translation.
- The translation of TTCN–3 `union` type constructs was improved. The former quadratic algorithm was replaced with a linear one, which results in both faster translation and smaller generated C++ code, especially for unions containing a large number of fields.
- Some error messages of the compiler printed the identifiers with double underscore characters.
- Logging of enumerated constants also used duplicated underscores.

4.72. Version 1.1.pl2

Released on July 23 2001.

New features

- Preparations for supporting TTCN–3 module parameters. They will be read runtime by the test executor from a configuration file.

Fixed bugs

- Port operations `connect`, `disconnect`, `map` and `unmap` now works for members of port arrays as well.
- The compiler generated erroneous C++ code for setting the initial value of variable and constant arrays.
- Functions were startable only if their parameters were explicitly denoted by in keyword.
- Variable `LD FLAGS` was defined twice in the generated `Makefile`.

4.73. Version 1.1.pl1

Released on July 10 2001.

Fixed bugs

- TTCN–3 operations all `port.start` and all `port.stop` were not supported by the compiler due to a bug.
- Value redirects in port receiving operations without value template did not cause syntax error during compilation.
- Compilation of the Base Library failed under FreeBSD due to a wrong `#ifdef` statement.
- White spaces were removed from the beginning of comment lines in `Makefile` template generated by the compiler. This confused the make utility on FreeBSD.
- C++ compiler flag `-O2` was removed from `Makefile` template generated by the compiler in order to decrease compilation time.
- In test port function `incoming message` logging is now performed before adding message into the port queue. This facilitates Test Port debugging (e.g. finding unbound fields).
- The user `log` statement in TTCN–3 now accepts more than one arguments separated by commas.

4.74. Version 1.1.pl0

Released in July 2001.

New features

- Parallel and distributed test execution is now supported.
- Internal communication of test components is supported in a transparent way.

- Explicit addressing in `send` and `receive` operations is supported. The sender's address in `receive` operations can be matched and stored.
- `Makefile` template generation by the compiler.
- The compiler can translate more than one modules given in command line.
- Test Port skeleton generation can be controlled by a `-t` command line switch of the compiler.
- The executor does not walk through the data structure to be sent or received, if the logging of port events is disabled. This increases the execution speed with one magnitude compared to logging.
- If a value (e.g. variable) is given as parameter to a send operation instead of a template, the *value) template) value* conversion chain is eliminated, which also results in higher performance.
- execute statements (either with or without timeout value) are supported. The old style (function-like) test case calls are still also supported.
- TTCN `error(…)` now accepts printf-style format string and variable number of arguments.
- New function `TTCN warning(…)` was introduced.
- The semicolon is an optional separator between function statements according to the newest TTCN–3 BNF.

NOTE	The parser may screw up if the omission of semicolon causes ambiguity.

- The compiler still accepts the obsolete duration units and the dash character (-) instead of the omit keyword, but gives a warning message during translation.

Fixed bugs

- In-line array initializers were not permitted by the compiler in the right values of local constant or variable definitions.
- The generated C++ code was erroneous when using single initial values for union constants or variables.
- Execution failed with a dynamic test case error in a send or receive statement when converting a record or set value with an omitted optional field to template.
- The bits of a bitstring constant were in the reverse order within the bytes.
- The constructor of class CHARSTRING with explicit length initialized the string only until the first NUL character in initial value.
- The message type contained duplicated underscore characters in the log entry of send and receive events.
- In case of port arrays, the name of each instance was not set properly.
- The value omit was permitted only in template bodies.

4.75. Version 1.0

First published release. Issued in January 19, 2001.

NOTE	The programs in this release do not support any version printout claiming 1.0. By this time the tool was simply called TTCN–3 Test Executor Prototype.

[2] The operands were literal values or references pointing to constants.

[3] TTCN–3 float and ASN.1 REAL values are represented in the double type of the C/C++ language,which is mapped to 64-bit floating point numbers on most platforms.

[4] This case is different from infinite blocking, that is, when no timers are running at all.

[5] FD SETSIZE is usually set to 1024. This implies that the test components cannot have more than1024 simultaneous port connections by default. This limitation is not applicable to the Main Controller (it uses poll instead of select) so you can work around this situation by using hierarchical test configuration with proxy components. Moreover, if you want to exceed this limit FD SETSIZE can be increased on some operating systems (e. g. Linux or Solaris 8). In this case you will need a special binary package, which is available on request.

Chapter 5. Changes of Test Port API

This section gives you a summary of changes on the Test Port API between the different versions of the test executor. You should check this list carefully if you want to use a Test Port developed for an older version with a newer version. Sometimes you have to change some pieces of code to perform a successful upgrade. The changes that result in incompatibility are denoted by word INCOMPATIBLE.

WARNING	The classes of data types or the Test Port base class may have some member functions that are not described in this document. These functions are written or generated only for internal purposes of the test executor. You should not use the undocumented functions because improper calls may cause the instability of your ETS. In addition, the interface of these functions may change in future releases without notice.

5.1. Version 1.8.pl6

No changes

5.2. Version 1.8.pl5

No changes

5.3. Version 1.8.pl4

No changes

5.4. Version 1.8.pl3

No changes

5.5. Version 1.8.pl2

Although there were no direct changes done to the testport API, the version handling feature introduced some ways for testport writers to constrain the use of their testports by:

- The version of TITAN using the TTCN3_VERSION macro
- The version of gcc using the GCC_VERSION macro

For more information please consult section 4.23.2 of the TITAN programmers reference guide.

5.6. Version 1.8.pl1

No changes

5.7. Version 1.8.pl0

No changes

5.8. Version 1.7.pl4

The Test Port was enhanced with the handling of the epoll system call present on recent linux systems, making it possible to handle connections in a much more efficient way. The already existing Test Port interface was kept, but it is no longer a direct interface, but a mapping to the new interface, which provides several new functions, that can be used to handle connections in a simplier manner (for example there are specific callback functions to handle new data appearing on a registered port connection).

5.9. Version 1.7.pl3

No changes.

5.10. Version 1.7.pl2

No changes.

5.11. Version 1.7.pl1

Log event subtypes were introduced and the generated codes were updated to use the new log events by default. The logger was enhanced to log the actual logging options. Some log messages were moved to other log event categories. (INCOMPATIBLE)

5.12. Version 1.7.pl0

- Test Port classes as well as C++ classes of user defined data types are put into a C++ namespace that corresponds to the TTCN–3 module. (INCOMPATIBLE)
- The C++ equivalents of TTCN–3 and ASN.1 enumerated values were moved into the scope of the C++ class that implements the enumerated type. The enum-hack option became obsolete. (INCOMPATIBLE)
- The C++ enumerated type that describes the selected field of a TTCN–3 union or ASN.1 CHOICE type was moved into the scope of the value class. The naming rules of the possible enumerated values have changed too. (INCOMPATIBLE)
- The C++ realization of TTCN–3 predefined function `ischosen` has changed. There is one common function, which takes the enumerated field identifier as argument instead of the former dedicated functions for all possible fields. (INCOMPATIBLE)
- Useless C++ classes and port operations are no longer generated for procedure based communication. Signatures with `noblock` keyword do not allow reply or `getreply`, signatures without exception types do not allow raise or catch. (INCOMPATIBLE)
- The C++ class that represents signature exceptions provides accessor functions for exception

types using a new naming convention. The naming convention of the enumerated type that describes the selected type has also changed. (INCOMPATIBLE)

- The address extension no longer works with imported address type. (INCOMPATIBLE)
- New API was introduced for provider port types. (ADDITION)

5.13. Version 1.6.pl5

No changes.

5.14. Version 1.6.pl4

- C++ representation of external function parameters without the in keyword has been changed. (INCOMPATIBLE)

5.15. Version 1.6.pl3

- Array types were introduced. (ADDITION)

5.16. Version 1.6.pl2

No changes.

5.17. Version 1.6.pl1

No changes.

5.18. Version 1.6.pl0

- The constructor of test port classes now has a default argument, which is a NULL pointer. In the newly created skeletons the default argument is already present, but it has to be added to the existing test port header files manually. Otherwise, if the port type is used in a port array the generated C++ code will not compile. The updated and newly created test ports also work with older versions. (INCOMPATIBLE)
- The member function `set size()` was added to the C++ equivalents of record of and set of types. (ADDITION)

5.19. Version 1.5.pl8

No changes.

5.20. Version 1.5.pl7

No changes.

5.21. Version 1.5.pl6

The global C-like functions for the conversion of enumerated values were moved into the class scope. They became static members of the value class. The old function names are still preserved for backward compatibility. (ADDITION)

5.22. Version 1.5.pl5

No changes.

5.23. Version 1.5.pl4

No changes.

5.24. Version 1.5.pl3

- Type universal charstring is now supported. Its equivalent C++ class named UNIVERSAL CHARSTRING was introduced. (ADDITION)

5.25. Version 1.5.pl2

No changes.

5.26. Version 1.5.pl1

No changes.

5.27. Version 1.5.pl0

- The C++ API for invoking the RAW and BER encoding/decoding functions has significantly changed. The purpose of changes was to provide a common, unified and more flexible interface for both encoding methods. See the TITAN TTCN–3 Programmer´s Technical Reference for more details. (INCOMPATIBLE)

5.28. Version 1.4.pl5

No changes.

5.29. Version 1.4.pl4

- The procedure based ports are now supported with an enhanced Test Port API. (ADDITION)
- The C++ code generated by the compiler has changed for record of, set of and enumerated types. These are not realized as C++ template classes any more, but as regular C++ classes. These changes should not cause any incompatibilities in properly written test ports.

- The parameter type and/or return type of some helper functions used for enumerated types has been changed from `int` to `enum`. This may require explicit casting in some Test Ports. (INCOMPATIBLE)

5.30. Version 1.4.pl3

- The interrupt signal is handled in single mode. (ADDITION)

5.31. Version 1.4.pl2

No changes.

5.32. Version 1.4.pl1

- Class CHAR, which was the C++ equivalent of the obsolete TTCN–3 type char was removed. The compiler substitutes all occurrences of type char with charstring and CHAR is now a typedef alias to class CHARSTRING, which provides partial backward compatibility. (INCOMPATIBLE)
- Hexstring related conversion functions were added. (ADDITION)

5.33. Version 1.4.pl0

No changes.

5.34. Version 1.3.pl0

- The symbolic constants NULL ADDRESS, MTC ADDRESS and SYSTEM ADDRESS were renamed to NULL COMPREF, MTC COMPREF and SYSTEM COMPREF, respectively. (INCOMPATIBLE)
- Usage of TTCN–3 `address` type is now supported. See the TITAN TTCN–3 Programmer´s Technical Reference. (ADDITION)

5.35. Version 1.2.pl4

No changes.

5.36. Version 1.2.pl3

- TTCN–3 type hexstring is now supported. Its C++ equivalent is class HEXSTRING. (ADDITION)

5.37. Version 1.2.pl2

No changes.

5.38. Version 1.2.pl1

- System related Test Port parameters were introduced. Their value is passed to the Test Port during test run when executing a map statement. (ADDITION)
- The set of type construct is now supported. The API is exactly the same as in case of record of. (ADDITION)

5.39. Version 1.2.pl0

- The new OBJID built-in type was added. (ADDITION) It is not compatible with CHARSTRING, which was OBJID aliased to before. (INCOMPATIBLE)
- The default mapping of underscore characters within file names (including Test Port header and source files) has been changed. (INCOMPATIBLE) Use the command line option `-u` of the compiler to switch the compatibility mode on.
- Member functions VERDICTTYPE::set and VERDICTTYPE::get were removed. Use the functions `TTCN Runtime::setverdict` and `TTCN Runtime::getverdict` to modify or get the local verdict. (INCOMPATIBLE)

5.40. Version 1.1.pl10

- The CHAR built-in type was added. (ADDITION)
- The bitwise and rotating overloaded operators were added. (ADDITION)

NOTE	You should not upgrade a Test Port developed for version 1.1.pl8 or earlier directly to 1.1.pl10 because operator `&` meant concatenation in older versions but it means now bitwise and operation. You should upgrade these old Test Ports to 1.1.pl9 first and after the successful compilation you can proceed to 1.1.pl10.

5.41. Version 1.1.pl9

- The concatenation operator has been changed from `&` to `+` for all string types. (INCOMPATIBLE)
- The modulo division overloaded operator has been removed from the class INTEGER. The `mod` and `rem` TTCN–3 operations can be performed by polymorphic global functions called `mod` and `rem`, respectively. (INCOMPATIBLE)

5.42. Version 1.1.pl8

No changes.

5.43. Version 1.1.pl7

No changes.

5.44. Version 1.1.pl6

No changes.

5.45. Version 1.1.pl5

No changes.

5.46. Version 1.1.pl4

No changes.

5.47. Version 1.1.pl3

- Member function set parameter was introduced. (ADDITION)
- The enumerated types are provided with a *value) string* and *string) value* conversion functions, i.e. `<enum type>` to `str` and `str` to `<enum type>`. (ADDITION)

5.48. Version 1.1.pl2

No changes.

5.49. Version 1.1.pl1

No changes.

5.50. Version 1.1.pl0

- The constructors of Test Port base classes take the port name as a parameter. In version 1.0 the port name was directly assigned in the constructor of the user code, but now it has to be passed to the constructor of base class. (INCOMPATIBLE)
- The function `Event Handler` has now four parameters containing the triggering file descriptors and the time elapsed since the last call. (INCOMPATIBLE)
- Functions user map and user `unmap` were introduced. (ADDITION)
- Function TTCN error accepts printf-style arguments and TTCN warning was introduced. (ADDITION)

Chapter 6. References

1. Methods for Testing and Specification (MTS);The Testing and Test Control Notation version 3.Part 1: Core Language European Telecommunications Standards Institute. ES 201 873-1 Version 4.1.1, July 2009
2. Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3. Part 4: TTCN–3 Operational Semantics European Telecommunications Standards Institute. ES 201 873-4 Version 4.1.1, June 2009
3. Methods for Testing and Specification (MTS);The Testing and Test Control Notation version 3.Part 7: Using ASN.1 with TTCN–3 European Telecommunications Standards Institute.ES 201 873-7 Version 3.1.1, June 2005
4. ITU-T, X.680, Information Technology Abstract Syntax Notation One (ASN.1): Specification of basic notation International Telecommunication Union, July 2002
5. ITU-T, X.682, Information Technology Abstract Syntax Notation One (ASN.1): Constraint specification International Telecommunication Union, July 2002
6. Installation guide for TITAN TTCN-3 Test Executor
7. Programmer's Technical Reference for TITAN TTCN-3 Executor
8. User Guide for TITAN TTCN-3 Test Executor