



OneM2M Meeting
Introduction to TTCN-3
Overview

Speaker



Theofanis Vassiliou-Gioles
Founder and CEO of Testing Technologies /
Director Business Development Automation
Platform Technologies
theofanis.vassiliou-gioles@spirent.com
www.spirent.com/ttcn-3

- Master in Electrical Engineering
- Started communication testing 1996
- ATM test specification standardization
- ETSI TTCN-3 standardization
- Application of test automation in new domains

Spirent Communications PROPRIETARY AND CONFIDENTIAL 18

Agenda

Our path through the workshop



TRI & TCI

What is required to make a test specification executable?

TRI / TCI

TTCN-3 Tools
What you get when you are using a TTCN-3 tool

Tools

TTCN-3
Introducing TTCN-3 with a short example to define a common language.

Spirent Communications

PROPRIETARY AND CONFIDENTIAL

19

OneM2M Meeting Introduction to TTCN-3

Basic Concepts of TTCN-3



What is TTCN-3?



- Testing and Test Control Notation
- Internationally standardized testing language for formally defining test scenarios. Designed purely for testing
- In its essence it can be considered as a kind of scripting language that includes tons of testing specific features!

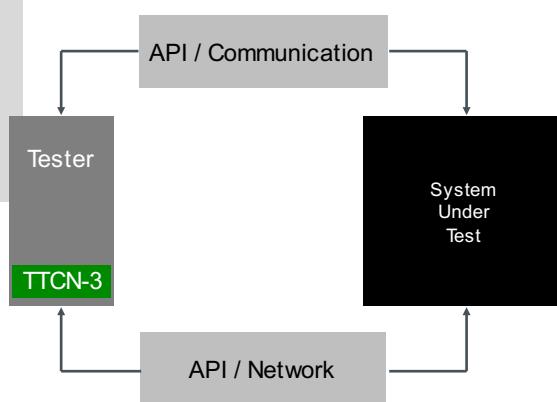


```
 testcase tc_Hello_Bob () {
    p.send("How do you do?");
    alt {
        [] p.receive("Fine!") {
            setverdict( pass );
        }
        [else]{
            setverdict( inconc );} //Bob asleep!
    }
}
```

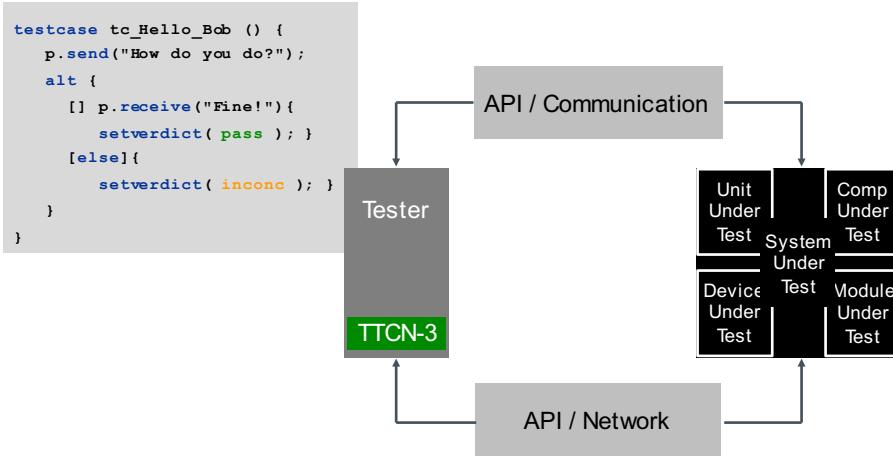
TTCN-3 execution



```
 testcase tc_Hello_Bob () {
    p.send("How do you do?");
    alt {
        [] p.receive("Fine!") {
            setverdict( pass );
        }
        [else]{
            setverdict( inconc );
        }
    }
}
```



TTCN-3 execution

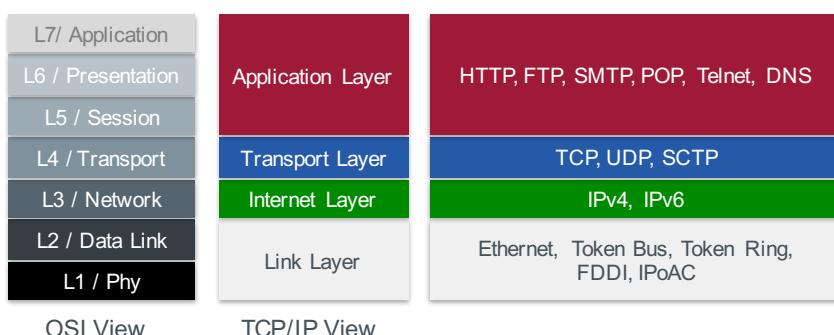


Spirent Communications

PROPRIETARY AND CONFIDENTIAL

23

Generic protocol architecture(s)

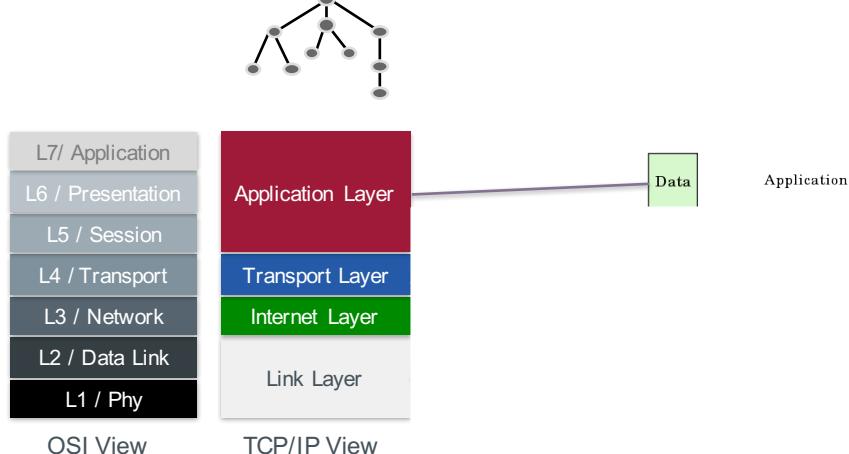


Spirent Communications

PROPRIETARY AND CONFIDENTIAL

24

Generic protocol architecture(s)



Spirent Communications

PROPRIETARY AND CONFIDENTIAL

25

The Example

We need an application that



- We all know or can understand
- Is simple enough but not trivial
- Is complex enough but let's not get lost in details
- Should be testable
- Should be fun

DNS Domain Name Server

Spirent Communications

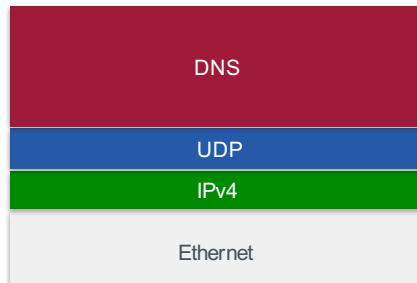
PROPRIETARY AND CONFIDENTIAL

26

When we test we...



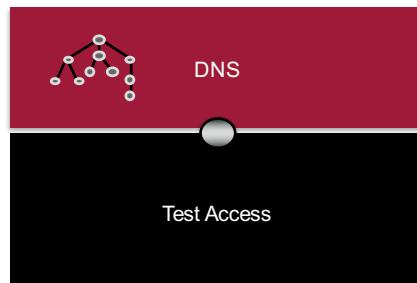
- Select the **protocol or application** to test
 - DNS
- Select the **test access**
 - UDP, IPv4, Ethernet



When we test we would like to ...



- Concentrate on the protocol (application) on an **abstract** level
- Do not care for the concrete technical details like test access



The DNS Server Example

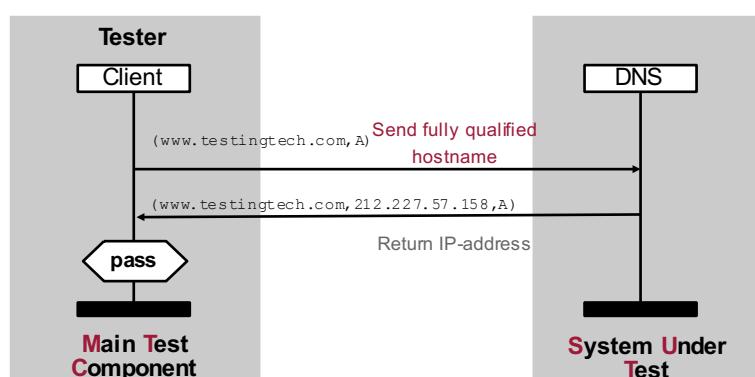


Spirent Communications

PROPRIETARY AND CONFIDENTIAL

29

The DNS Server Example



Spirent Communications

PROPRIETARY AND CONFIDENTIAL

30

TTCN-3 modules



- Main building block of TTCN-3 is a module
 - Unit of compilation
 - Contains definitions
 - Optional control part

```
module DNS {  
    // module definitions  
  
    // module control (optional)  
}
```

Module definitions



- Contains descriptions for
 - What type of data the System Under Test understands
 - How the System Under Tests can be accessed and what environment a test component needs
 - When to communicate what with the SUT and why
 - Dependencies between test cases, if any

Module definitions (1)



- Module definitions

- Type definitions
- Port definitions
- Component definitions
- Templates
- Test case

- Control part

- Controls the execution of test cases

```
type record DNSMessage {
    charstring hostname,
    AnswerType answer optional,
    QueryType qtype
}

type union AnswerType {
    Byte ipAddress[4],
    charstring hostname
}

type integer Byte (0 .. 255);

type enumerated QueryType {
    A, NS, CNAME, MX
}
```



Module definitions (1)



- Module definitions

- Type definitions
- Port definitions
- Component definitions
- Templates
- Test case

- Control part

- Controls the execution of test cases

```
type record DNSMessage {
```



Module definitions (2)



- Module definitions
 - Type definitions
 - **Port definitions**
 - **Component definitions**
 - Templates
 - Test case
- Control part
 - Controls the execution of test cases

Port definitions

```
type port DNSPort message {
    inout DNSMessage;
    // a port may send/receive messages
    // of more than one type
}
```

Component definitions

```
type component DNSTester {
    port DNSPort P;
    timer t := 3.0;
    // a component may have more than one port
}
```



Spirent Communications

PROPRIETARY AND CONFIDENTIAL

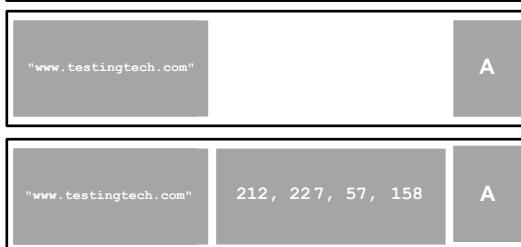
35

Module definitions (3)



- Module definitions
 - Type definitions
 - Port definitions
 - Component definitions
 - **Templates**
 - Test case
- Control part
 - Controls the execution of test cases

```
template DNSMessage query := {
    hostname := "www.testingtech.com",
    answer := omit,
    qtype := A
}
template DNSQuery reply modifies query := {
    answer := { ipAddress := {212,227,57,158} }
}
```



Spirent Communications

PROPRIETARY AND CONFIDENTIAL

36

Module definitions (4)



- Module definitions

- Type definitions
- Port definitions
- Component definitions
- Templates
- **Test case**

```
 testcase tc_testcase1() runs on DNSTester {  
    P.send(query);  
    P.receive(reply);  
    setverdict(pass);  
}  
  
// there may be more than one in a module
```

- Control part

- Controls the execution of test cases



Spirent Communications

PROPRIETARY AND CONFIDENTIAL

37

Module definitions (5)



- Module definitions

- Type definitions
- Port definitions
- Component definitions
- Templates
- Test case

- Control part

- **Controls the execution of test cases**

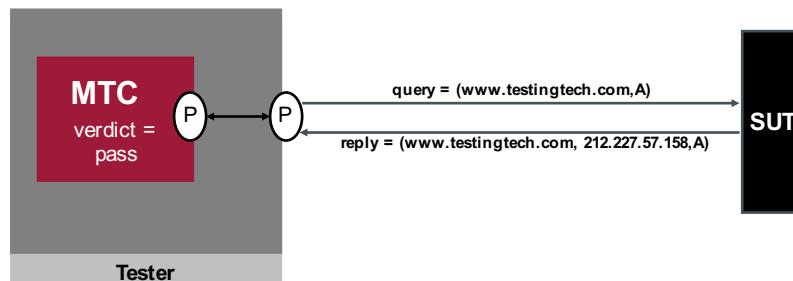
```
control {  
  
    execute(tc_testcase1(), 5.0);  
    while( /* condition */ ) { };  
  
    // more testcases might follow  
    // C-like control structures available  
}
```

Spirent Communications

PROPRIETARY AND CONFIDENTIAL

38

Execution of a test case

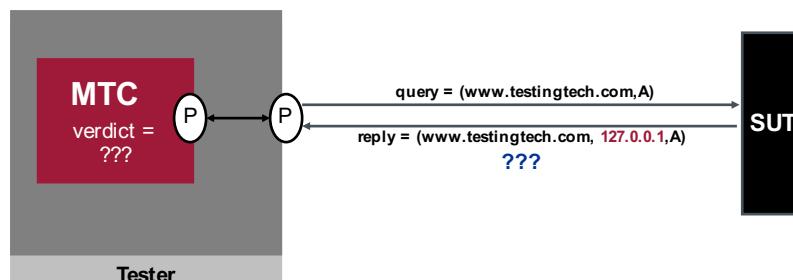


```
testcase tc_testcase1() runs on DNSTester {
    P.send(query);
    P.receive(reply);
    setverdict(pass);
}
```

Is this test case definition adequate?

Is this an effective test case definition?

Dealing with erroneous behavior (1)

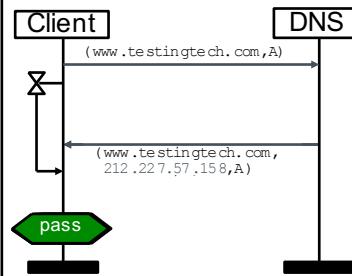


- `P.receive (reply)` blocks until it receives a message that matches the reply.
- If unexpected message is received, any other correct message does not unblock the tester, which then blocks forever.
- If no message is received, the tester will also block forever.

Dealing with erroneous behavior (2)



```
 testcase tc_testcase2() runs on DNSTester {
    P.send(query);
    t.start;
    alt {
        [] P.receive(reply) {
            setverdict(pass);
        }
        [] P.receive { // any message
            setverdict(fail);
        }
        [] t.timeout {
            setverdict(inconc);
        }
    }
    stop;
}
```

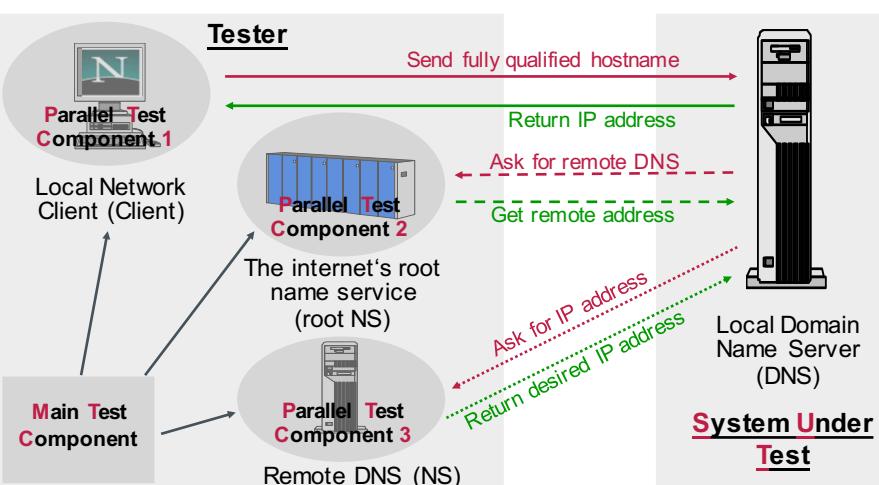


Spirent Communications

PROPRIETARY AND CONFIDENTIAL

41

Non-Local DNS Query (1)



Spirent Communications

PROPRIETARY AND CONFIDENTIAL

42

From Simple To Complex Test Scenarios



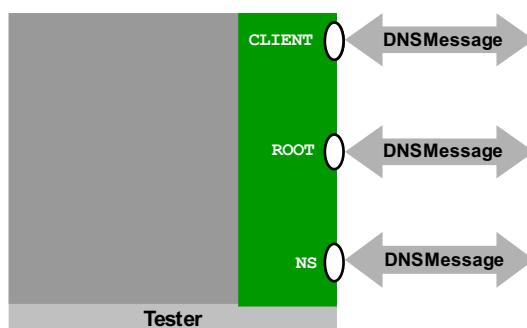
- Test system needs more interfaces
 - Test System Interface has to be extended
- Test behavior required at additional test interfaces
 - Behavior of Local Network Client already covered in `tc_testcase2`
 - Behavior of RootNS and NS required
- Test case that combines all parts

Parallel Test Components



- Test system interface

```
type component TestSystemInterface {  
    port DNSPort CLIENT;  
    port DNSPort ROOT;  
    port DNSPort NS;  
}
```



From Test Case to Behavior Function



- Functions can be used to define the behavior of the parallel test components

```
 testcase tc_testcase2() runs on DNSTester {  
    var default d := activate(a_refactoredAltstep());  
    P.send(query);  
    t.start;  
    P.receive(answer);  
    setverdict(pass);  
    stop;  
}
```

becomes

```
function f_clientBehavior() runs on DNSTester {  
    var default d := activate(a_refactoredAltstep());  
    P.send(query);  
    t.start;  
    P.receive(answer);  
    setverdict(pass);  
    stop;  
}
```

Additional Test Behavior



- Simple „react-on-request“ behavior

```
function f_rootBehavior() runs on DNSTester {  
    alt {  
        [] P.receive(rootquery) {  
            P.send(roootanswer);  
            setverdict(pass);}  
        [] P.receive {  
            setverdict(fail);}  
    }  
}
```

```
function f_nsBehavior() runs on DNSTester {  
    alt {  
        [] P.receive(nsquery) {  
            P.send(nsanswer);  
            setverdict(pass);}  
        [] P.receive {  
            setverdict(fail);}  
    }  
}
```

Dynamic configuration



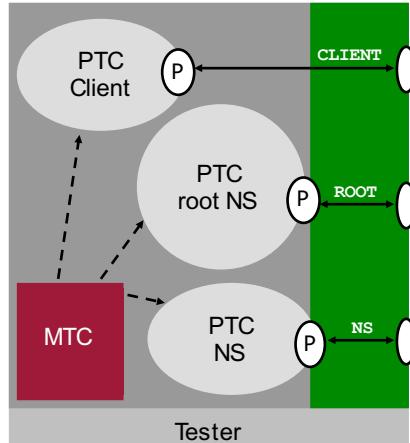
```
 testcase testcase3() runs on DNSTester
 system TestSystemInterface {
    var DNSTester ClientComp, RootComp,
        NSComp;

    ClientComp := DNSTester.create;
    RootComp := DNSTester.create;
    NSComp := DNSTester.create;

    map(ClientComp:P, system:CLIENT);
    map(RootComp:P, system:ROOT);
    map(NSComp:P, system:NS);

    ClientComp.start(f_clientBehavior());
    RootComp.start(f_rootBehavior());
    NSComp.start(f_nSBehavior());

    ClientComp.done;
    // block until ClientComp is done
    stop;
}
```



- Re-configuration during runtime is possible

Spirent Communications

PROPRIETARY AND CONFIDENTIAL

47

What's next on our Agenda?

Our path through the workshop



The Example

- We all know or can understand
- Is simple enough but not trivial
- Is complex enough but let's not get lost in details
- Is testable
- Is fun

TTCN-3
Introducing TTCN-3 with a short example to define a common language.

TRI & TCI

What is required to make a test specification executable?



Spirent Communications

PROPRIETARY AND CONFIDENTIAL

48

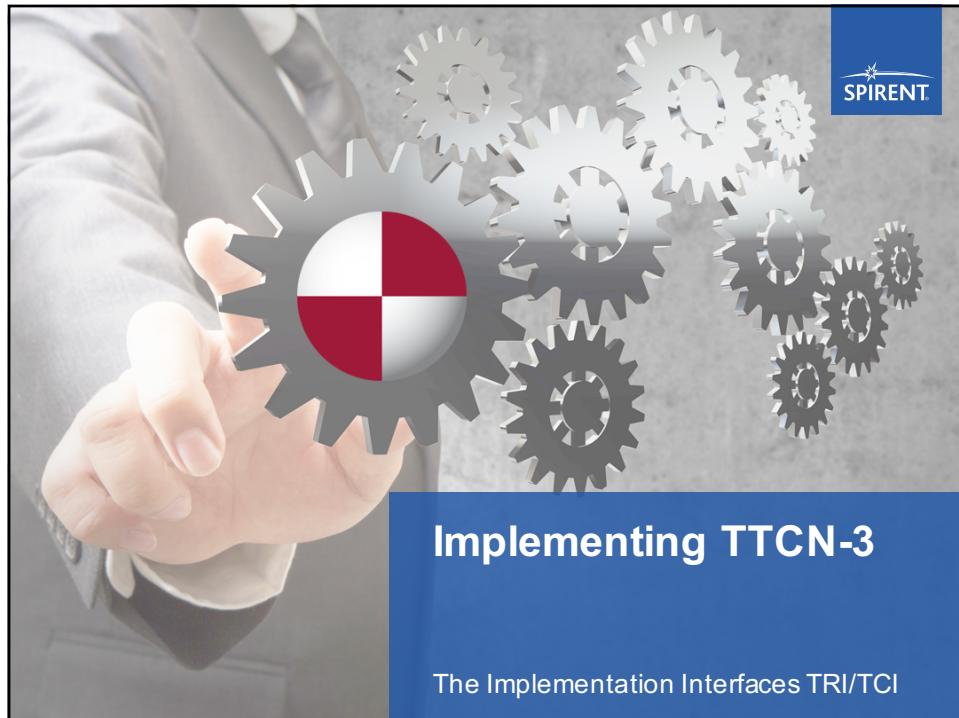
Questions at hand



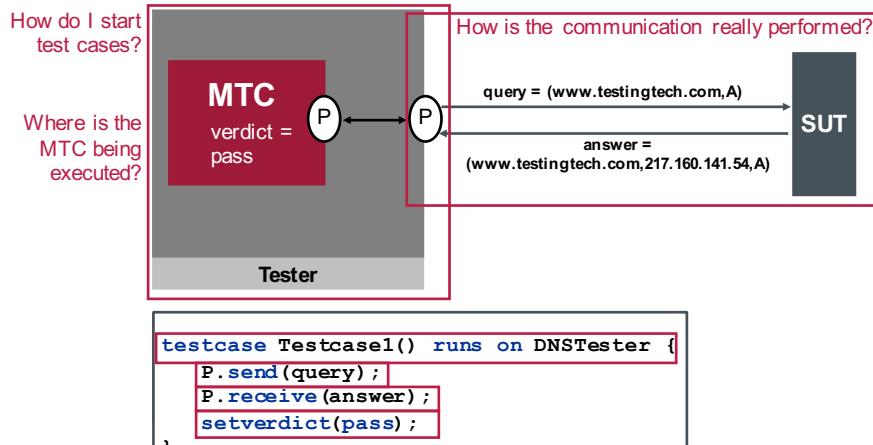
- We have a test specification
- We have been concentrating on the abstract description of our tests
- But ... What do I really need to make it happen?
- In other words
 - What is the architecture of the full test system
 - How and when are messages really been sent
 - When is the data transformed into a format the SUT understands?
 - Who is responsible in providing the software pieces?
 - How can this pieces be implemented?
 - Can I reuse existing software components?
 - Can I document what I am doing?

Implementing TTCN-3

The Implementation Interfaces TRI/TCI



Implementation Interface ?

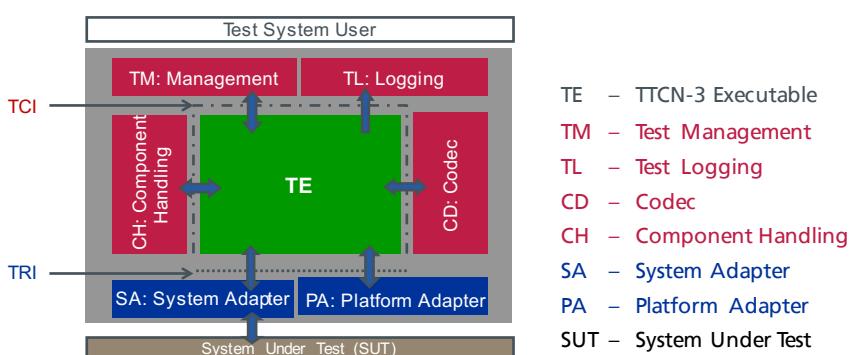


Spirent Communications

PROPRIETARY AND CONFIDENTIAL

51

A TTCN-3 Test System



ETSI ES 201 873-1 TTCN-3 Core Language (CL)

ETSI ES 201 873-5 TTCN-3 Runtime Interface (TRI)

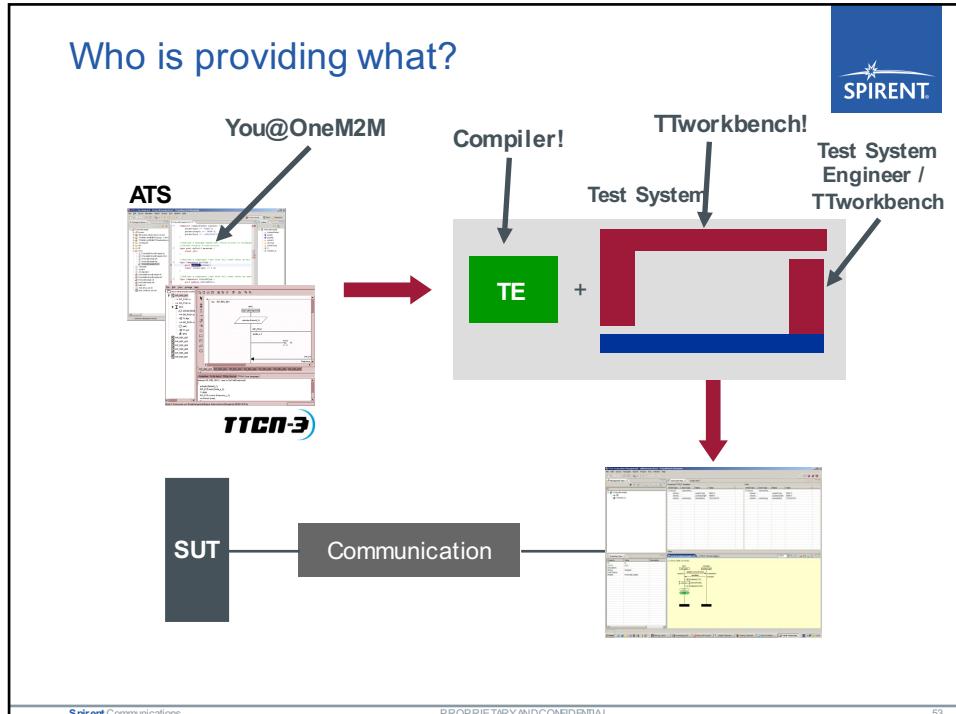
ETSI ES 201 873-6 TTCN-3 Control Interfaces (TCI)

Spirent Communications

PROPRIETARY AND CONFIDENTIAL

52

Who is providing what?



Steps to Implement TTCN-3

- Translate TTCN-3 into executable code
- Adapt runtime environment to test management
- Implement communication and test platform aspects

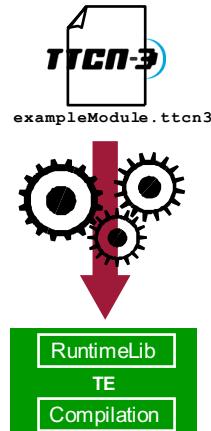
Steps to Implement TTCN-3



- Translate TTCN-3 into executable code
- Adapt runtime environment to test management
- Implement communication and test platform aspects

F : \AB>TTthree DNSTest

- Reads module definitions written in the TTCN-3 core notation
- Generates code and compiles it into executable code
- Runtime support through runtime libraries

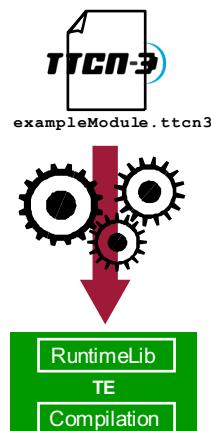


Translate TTCN-3 Into Executable Code



F : \AB>TTthree DNSTest

- Reads module definitions written in the TTCN-3 core notation
- Generates code and compiles it into executable code
- Runtime support through runtime libraries



Steps To Implement TTCN-3



- Translate TTCN-3 into executable code
- **Adapt runtime environment to test management**
- Implement communication and test platform aspects

Spirent Communications

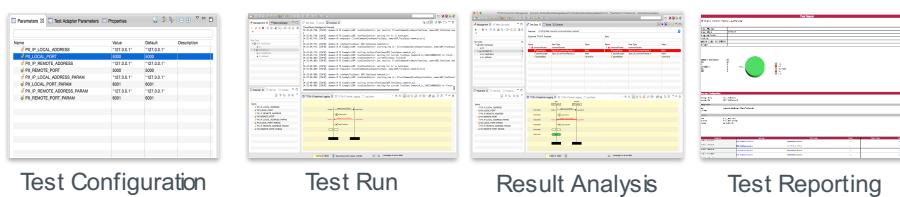
PROPRIETARY AND CONFIDENTIAL

57

Steps To Implement TTCN-3



- Translate TTCN-3 into executable code
- **Adapt runtime environment to test management**
- Implement communication and test platform aspects



Test Configuration

Test Run

Result Analysis

Test Reporting

Spirent Communications

PROPRIETARY AND CONFIDENTIAL

58

Steps To Implement TTCN-3



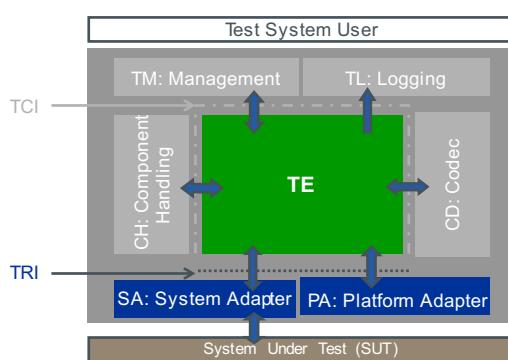
- Translate TTCN-3 into executable code
- Adapt runtime environment to test management
- **Implement communication and test platform aspects**

Spirent Communications

PROPRIETARY AND CONFIDENTIAL

59

TRI – Communication Adaptation



- Facts on the TTCN-3 Runtime Interfaces (TRI)
 - Standardized (part 5)
 - Language independent specification
- Implementation Languages
 - Java, C, C++, C# (standardized)
 - Remote API via messages (not standardized)

Spirent Communications

PROPRIETARY AND CONFIDENTIAL

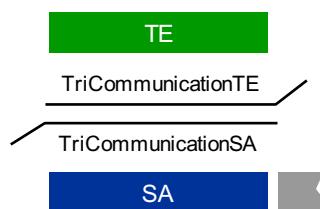
60

Why TRI ?



- Abstract Test Specifications (ATS) have to run on different test devices of different vendors
 - Different access to underlying protocol stacks
- ATS shall runs against systems in different development stages
 - Simulation
 - Software only
 - Embedded in hardware
- ATS can use different communications mechanisms and dynamic test configurations

The TRI Communication Interface

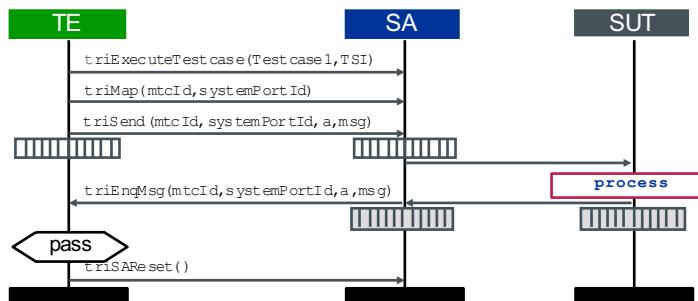


- Interface structure
 - Bi-Directional Interface
 - Applies to all TRI interfaces
 - SA reports status back
 - TE indicates error
- Languages
 - Java, C, C++, C# (*standardized*)
 - *Remote API via messages (not standardized)*

Dynamics of TRI SA



```
 testcase Testcase1 () runs on DNSTester system TSI {
    map(mtc:P, system:P);
    P.send(query);
    P.receive(answer);
    setverdict(pass);
}
```



Spirent Communications

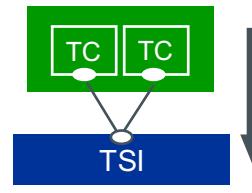
PROPRIETARY AND CONFIDENTIAL

63

TriCommunicationSA Interface



- Defines setting up configuration and sending of message to and/or calling of operations in the SUT
- Complete set of operations
 - `TriStatusType triSUTActionInformal (...);`
 - `TriStatusType triExecuteTestCase(...);`
 - `TriStatusType triMap(...);`
 - `TriStatusType triUnmap(...);`
 - `TriStatusType triSend(...);`
 - `TriStatusType triCall(...);`
 - `TriStatusType triReply(...);`
 - `TriStatusType triRaise(...);`



Spirent Communications

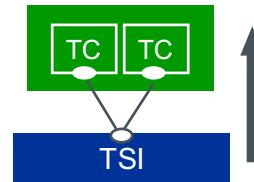
PROPRIETARY AND CONFIDENTIAL

64

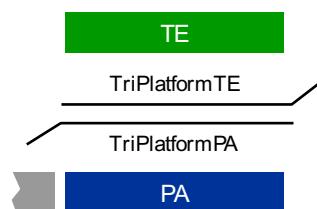
TriCommunicationTE Interface



- Defines receiving of messages and/or calling of operations in the TE
- Complete set of operations
 - `void triEnqueueMsg(...);`
 - `void triEnqueueCall(...);`
 - `void triEnqueueReply(...);`
 - `void triEnqueueException(...);`



The TRI Platform Interface

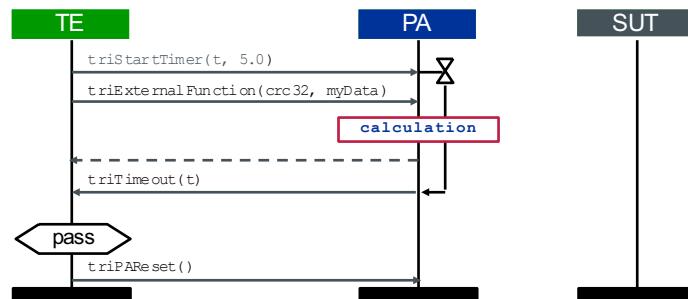


- Interface structure
 - Implementation of time and external functions
 - PA reports status back
 - TE indicates error
- Languages
 - Java, C, C++, C# (*standardized*)
 - *Remote API via messages (not standardized)*

Dynamics of TRI PA



```
 testcase Testcase2 () runs on DNSTester system TSI{
    var timer t := 5.0 ;
    t.start ;
    var octetstring crc := crc32(myData) ;
    t.timeout;
    setverdict(pass);
}
```



Spirent Communications

PROPRIETARY AND CONFIDENTIAL

67

TriPlatform Interface



- Defines control of time and calling of external functions
- Complete set of operations (PA)
 - `TriStatusType triPAREset();`
 - `TriStatusType triStartTimer(...);`
 - `TriStatusType triStopTimer(...);`
 - `TriStatusType triReadTimer(...);`
 - `TriStatusType triTimerRunning(...);`
 - `TriStatusType triExternalFunction(...);`
- Complete set of operations (TE)
 - `void triTimeout(...);`

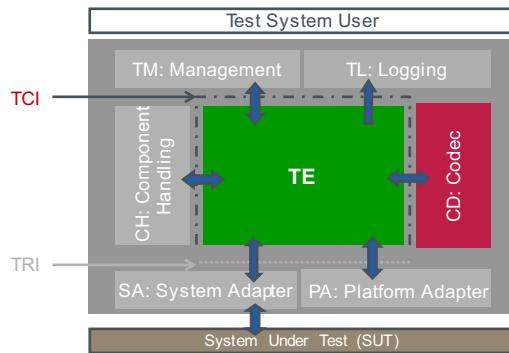


Spirent Communications

PROPRIETARY AND CONFIDENTIAL

68

A TTCN-3 Test System



TE – TTCN-3 Executable
TM – Test Management
TL – Test Logging
CD – Codec
CH – Component Handling
SA – System Adapter
PA – Platform Adapter
SUT – System Under Test

ETSI ES 201 873-1 TTCN-3 Core Language (CL)
ETSI ES 201 873-5 TTCN-3 Runtime Interface (TRI)
ETSI ES 201 873-6 TTCN-3 Control Interfaces (TCI)

Spirent Communications

PROPRIETARY AND CONFIDENTIAL

32

Why Codec Interface?



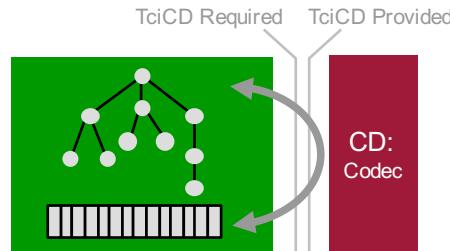
- TTCN-3 data has to be translated into a representation the SUT understands
- Two different tasks
 - Encoding
 - Internal TTCN-3 data representation to bitstring
 - Needs access to the TTCN-3 type and value system
 - Decoding
 - Bitstring to TTCN-3 data representation
 - Based upon a decoding hypothesis
 - TE may query multiple times for the decoding of the same bitstring

Spirent Communications

PROPRIETARY AND CONFIDENTIAL

70

The Codec and Value Interface



- Facts on the TTCN-3 Type and Value Interface
 - TE maintains abstract type and data presentation
 - Codec translates between abstract and concrete presentation

- Management of different codecs
- Complete set of provided (TciCDProvided) operations
 - TriMessageType encode (in Value value)
 - Value decode (in TriMessageType message, in Type hyp)

Spirent Communications

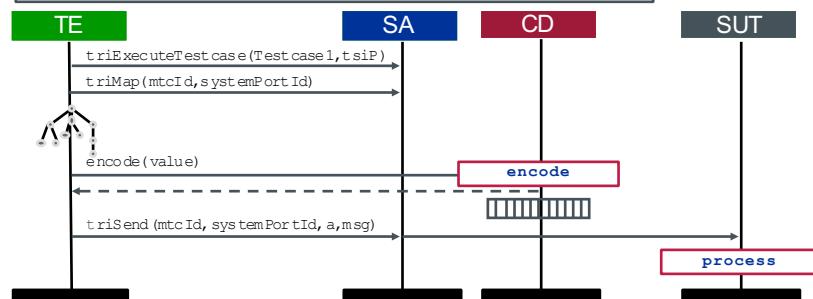
PROPRIETARY AND CONFIDENTIAL

71

Dynamics of the Codec (Sending)



```
testcase Testcase1 () runs on DNSTester system TSI {
    map(mtc:P, system:P);
    P.send(query);
    P.receive(answer);
    setverdict(pass);
}
```



Spirent Communications

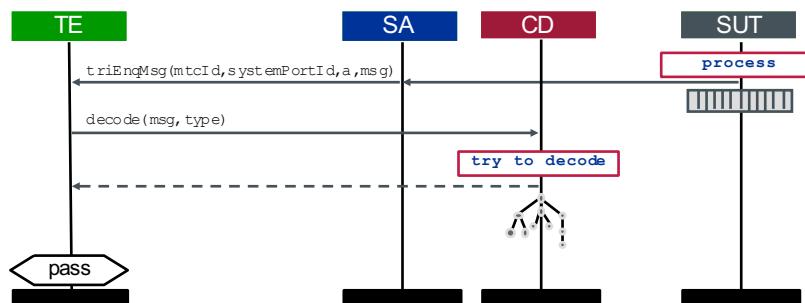
PROPRIETARY AND CONFIDENTIAL

72

Dynamics of the Codec (Receiving)



```
testcase Testcasel () runs on DNSTester system TSI{
    map(mtc:P, system:P);
    P.send(query);
    P.receive(answer);
    setverdict(pass);
}
```



Spirent Communications

PROPRIETARY AND CONFIDENTIAL

73

The decoding Hypothesis



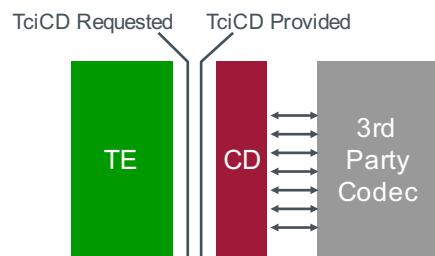
- The interpretation of an arbitrary bitstring is context sensitive
- Example: What is '56455300'?
 - Four bytes as one octetstring: '56455300'0
 - An integer: 1447383808
 - A charstring: "YES"
- Decode() can be read as follows:
 - Try to decode the provided bitstring, with the appropriated decoding rules into a value of given type
 - If you succeed, return the value
 - If you fail, return **NULL**

Spirent Communications

PROPRIETARY AND CONFIDENTIAL

74

3rd Party Codecs



- 3rd party codec provides
 - Operations to construct values
 - Operations to query values
 - Operations to encode values
 - Operations to decode messages in data structures
- CD provides
 - Operations to construct values
 - Operations to query values
- CD implementation maps TCI value structures into codec value structures

Coding Examples - SA

Taken From the AddressBook Example



Import Example Projects



- TTworkbench contains several example projects. Each TTplugin contains at least one example project too.
- Import over File > Import > TTCN-3 > TTCN-3 Examples > Address Book Example – message based.
- In the folder AddressBookMsgRuntime there are codec (...\\codec) and port plugin(...\\tri) implementation example
- In port plugin there are several functions

Test Adapter Practically

AddressBookTestAdapter



- Extends basic test adapter class
`com.testingtech.ttcn.tri.TestAdapter`
- Overrides the following methods
 - `public TriStatus triExecuteTestCase (TriTestCaseId tc, TriPortIdList tsiPortIdList)`
 - `public TriStatus triMap (TriPortId compPortId, TriPortId tsiPortId)`
 - `public TriStatus triUnmap (TriPortId compPortId, TriPortId tsiPortId)`
 - `public TriStatus triSend (TriComponentId componentId, TriPortId tsiPortId, TriAddress address, TriMessage sendMessage)`
 - `public TciCDProvided getCodec (String encodingName)`
 - `public void triSAReset()`

public TriStatus triExecuteTestcase (TriTestCaseId tc, TriPortIdList tsilist)



```
public TriStatus triExecuteTestcase(final TriTestCaseId testcase, final TriPortIdList tsilist) {
    // get the parameter values from the management (TA)
    PluginIdentifier pluginIdentifier = new
        PluginIdentifier("com.testingtech.ttcn.example.AddressBookMsgRuntime");
    // read the remote IP address
    remoteIPAddress = etAParameter(pluginIdentifier, "myPort", "REMOTE_IP_ADDRESS", "");
    if (remoteIPAddress.equals("")) {
        return new TriStatusImpl("could not resolve remote IP address");
    }
    // read the remotePortNumber and localPortNumber in the same way
    ...
    rxSocket = null;
    txSocket = null;
    return new TriStatusImpl();
}
```

- Called just before a test cases starts execution
- **triExecuteTestcase** gets TA parameter values
- If succeeds returns **TRI_OK**, Usage of predefined class **TriStatusImpl()**

public TriStatus triMap(TriPortId compPortId, TriPortId tsiPortId)



- Maps a test component port to a test system interface port
- In dynamic configurations typically receiver loops are started
- Parameter **compPortId** is a port reference to the test component port
- Parameter **tsiPortId** is a port reference to the test system interface port

public TriStatus triMap(TriPortId compPortId, TriPortId tsiPortId)



```
public TriStatus triMap (final TriPortId compPortId, final TriPortId tsiPortId)
{
    // prepare to be ready to communicate, define the sockets for sending and
    // receiving
    rxSocket = new DatagramSocket(localPortNumber);
    txSocket = new DatagramSocket();

    ...
    // Define and start a thread for listening on the receiver socket
    ...
    while (mylock) { ...
        rxSocket.receive(packet);
        if (runThread) {
            triEnqueueMsg(tsiPortId, new TriAddressImpl(new byte[] {}),
                          compPortId.getComponent(), rcvMessage);
        }
    }
}
```

- Call **triMap** in the default SUT adapter
- **triMap** receives portIds of ports being mapped
- If succeeds, returns **TRI_OK**

public TriStatus triUnmap(TriPortId compPortId, TriPortId tsiPortId)



- Unmaps a test component port from a test system interface port
- Stops the receiver loop
- Parameter **compPortId** is a port reference to the test component port
- Parameter **tsiPortId** is a port reference to the test system interface port

```
public TriStatus triSend(TriComponentId
componentId, TriPortId tsiPortId, TriAddress
address, TriMessage sendMessage) (1)
```



- Is called when it executes a TTCN-3 send operation on a test component port
- Performs the real sending on the respective test system interface port
- Encoding of **sendMessage** has to be done prior to this operation call
- Parameter **componentId** is the sending test component
- Parameter **tsiPortId** is the test system interface port via the message is sent
- Parameter **SUTaddress** is the optional destination address within the SUT
- Parameter **sendMessage** is the encoded message to be sent

```
public TriStatus triSend(TriComponentId
componentId, TriPortId tsiPortId, TriAddress
address, TriMessage sendMessage) (2)
```



```
public TriStatus triSend(final TriComponentId componentId, final TriPortId tsiPortId,
final TriAddress address, final TriMessage sendMessage) {
try {
    final byte[] msg = sendMessage.getEncodedMessage();
    final InetAddress addr = InetAddress.getByName(remoteIP);
    final DatagramPacket packet = new DatagramPacket(msg, msg.length, addr,
remotePortNumber);
    // Sending message
    txSocket.send(packet);
    return new TriStatusImpl();
} catch (final IOException ioex) {
    return new TriStatusImpl(ioex.getMessage());
}
}
```

- Send the already encoded data using UDP

Agenda

Our path through the workshop



TRI & TCI

What is required to make a test specification executable?

TRI / TCI

TTCN-3
Introducing TTCN-3 with a short example to define a common language.

Tools

TTCN-3 Tools
What you get when you are using a TTCN-3 tool

Spirent Communications

PROPRIETARY AND CONFIDENTIAL

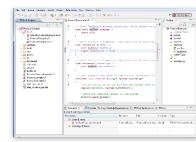
85

OneM2M Meeting Introduction to TTCN-3

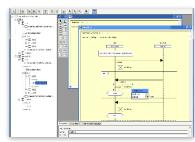
TTCN-3 Tools / TTworkbench



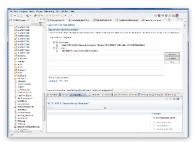
Test development



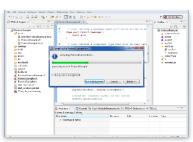
Test Scripting



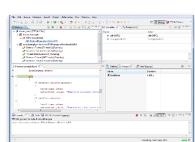
Graphical Test Specification



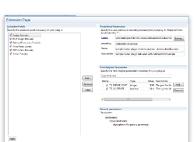
Test Capturing



Test Compiling



Test Debugging



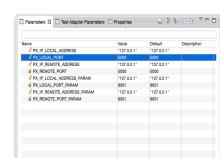
Test Adaptation Development

Spirent Communications

PROPRIETARY AND CONFIDENTIAL

87

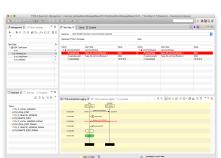
Test execution



Test Configuration



Test Run



Result Analysis



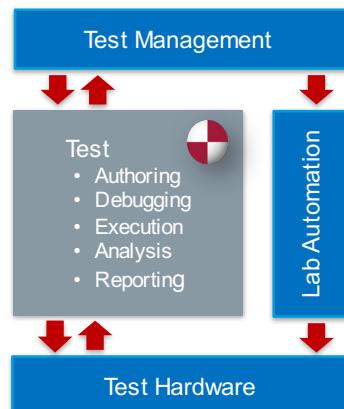
Test Reporting

Spirent Communications

PROPRIETARY AND CONFIDENTIAL

88

Test automation eco system



- **Test Management** – When and how to test
 - Interfacing to test management tools like Rational Quality Manager or continuous integration platforms like Jenkins
- **Lab Automation** – Setting up the environment
 - Lab Automation takes care of the whole setup, in particular network configuration, but also including resource planning
- **Test Hardware** – Realizing the test access to the SUT
 - Easy adaptation to different test hardware, due to standardized interfaces (TRI/TCI)
 - Ready to use plugins for easy test access on standard PC platforms

TTplugin concept



- Fast and easy extension of TTworkbench capabilities
- No implementation effort
- Automatic import of specifications into TTCN-3
- Automatic codec generation
- Freely combinable with additional test access methods

List of plugins



- Languages
 - XSD / XML / JSON
 - WSDL / SOAP
 - Google's Protobuf
 - ASN.1 / CSN.1
 - IDL
- Database / Mgmt
 - SNMP
 - SQL
- H2M Automation
 - Selenium
 - Telnet / SSH
 - SCP / FTP
 - RS232
 - GPIB
 - Spirent TestCenter
- IP / Datacom
 - RTP
 - HTTP / TLS
 - TCP / UDP
 - Ethernet / IP
- Automotive
 - CAN
 - MOST
- Mobility
 - VoiceQualityRTP
 - LTE NAS
- Misc
 - ReleaseManagement
 - TTtwo2three
 - Distribution via TTmex



Demo!

FAQ



- What skills do I need to write TTCN-3 code?
- Can I learn TTCN-3 easily?
- What skill do I need to develop a TTCN-3 based test system?

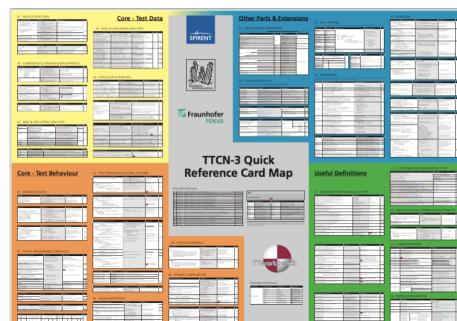
- Why shoud I write TTCN-3 code?
- Can't you write it for me?

- Does the TTCN-3 tools XYZ compile to
 - C?
 - C++?
 - Java?
- Can I implement my adapters in C?
 - Or in C++?
 - Or in Python?
- Can I get some examples?

Some references



- The language
 - www.ttcn-3.org
 - <http://www.spirent.com/go/TTCN-3>
 - de.wikipedia.org/wiki/TTCN-3
 - en.wikipedia.org/wiki/TTCN-3
- The Quick Reference Card
 - www.blukaktus.com/card.html
- TTCN-3 / TTworkbench
 - <http://www.spirent.com/TTworkbench>





Theofanis Vassiliou-Gioles

spirent.com

© Spirent Communications, Inc. All of the company names and/or brand names and/or product names and/or logos referred to in this document, in particular the name "Spirent" and its logo device, are either registered trademarks or trademarks pending registration in accordance with relevant national laws. All rights reserved. Specifications subject to change without notice.