

# Experiences of Using TTCN-3 for Testing SIP and OSP

## Authors

Anthony Wiles, ETSI, France  
Theofanis Vassiliou-Gioles, Testing Technologies, Germany  
Scott Moseley, FSCOM, France  
Sebastian Mueller, FSCOM, France

## Other Contributors

Frederique Aurouet, ACACIA, France  
Csaba Koppany, Ericsson, Hungary

**Abstract.** The ultimate aim of standardisation is to enable interoperability of ITC products, ideally on a global scale. ETSI members have long-recognized that well-defined, unambiguous and validated standards, together with testing, are crucial elements in achieving that goal. Rigorous testing of VoIP-related protocols is becoming increasingly important and TTCN-3, the next generation testing language developed by ETSI, is a key component in achieving this. This engineering paper describes how ETSI Project TIPHON™ is using TTCN-3 to develop test suites for SIP (IETF Session Initiation Protocol) and TIPHON™ OSP (Open Settlement Protocol). This paper also describes the execution and validation of the test suites on real TTCN-3 test platforms and describes experiences of running the tests against actual implementations.

## 1 What are SIP and OSP?

The IETF Session Initiation Protocol [1], or SIP, has been developed as a control protocol for establishing, changing and tearing down multimedia calls between one or more endpoints in an IP network. These calls may range from simple person-to-person voice calls to full Internet multimedia conferencing. Conceptually, there are two SIP protocol components: The SIP User Agent (UA) and the SIP Network Server. The UA is essentially the end system component comprising two elements: the UA Client (UAC) for initiating calls and the UA Server (UAS) for answering calls. In the network a SIP server may be a proxy server or a redirect server. SIP messages, such as INVITE and ACK, are text-based and are defined using BNF (Backus-Naur Form). We shall see later in this paper how this can pose particular challenges when specifying tests for SIP.

The ETSI TIPHON™ Open Settlement Protocol [2], or OSP, permits the exchange of inter-domain pricing, authorization, and settlement information between Internet telephony operators. OSP fulfils the essential requirements of such services by providing appropriate functionality between multiple administrative domains in a secure manner. OSP also provides for non-standard extensions that permit co-operating parties to augment or replace the basic functionality. OSP messages are defined in XML. Again, we shall show that this has an impact on how we specify tests for OSP.

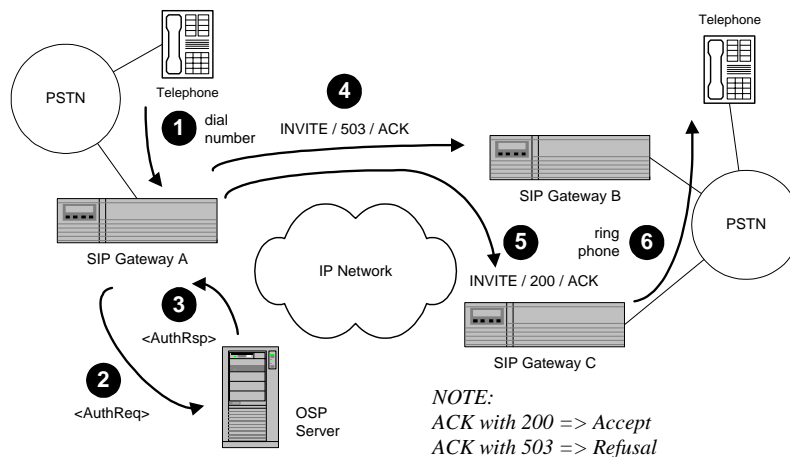


Figure 1: Example of how OSP may be used by SIP gateways to find and authorize each other.

## 2 The ETSI Approach to Testing

ETSI members have long recognized the important role that validation and testing play in the development of ETSI standards, and of the products based on those standards. Interoperability is a key issue, and activities such as conformance and interoperability testing are crucial to achieving this goal.

ETSI has two permanent entities involved in testing activities. The ETSI Protocol and Testing Competence Centre (<http://www.etsi.org/ptcc>) and the Plugtests Service ([www.etsi.org/plugtests](http://www.etsi.org/plugtests)).

The PTCC provides support and services to ETSI Technical Bodies on the application of modern techniques for specifying protocols and test specifications. The PTCC is also responsible for the technical management of the ETSI Specialist Task Forces (STFs) that develop conformance test specifications for ETSI standards. In the past 10 years test suites have been produced for many leading ETSI technologies, including 3G UMTS, GSM, DECT™, TETRA, Hiperlan/2, TIPHON™, INAP, B-ISDN etc.

While certain areas quite rightly require regulatory testing the policy today is to keep this to a minimum. As a consequence, there has been a subtle but fundamental change in the nature of ETSI test suites over the last few years. ETSI test specifications concentrate on maximizing the chances of interoperability, for example by increased focus on critical functionality or error recovery behaviour. The test suites are not part of a bureaucratic testing program or certification scheme and are increasingly used by manufacturers and fora as a key step in product development.

The ETSI Plugtests service organizes interoperability events for ETSI members and non-members alike. It provides the logistical and organizational support (often at ETSI premises) for such events. Recent events include SynchFest, IPv6 InterOp, IMTC SuperOp and Bluetooth UnplugFest. The intention with plugtests is to validate (debug) both the standards and early products or prototypes of those standards as they are developed.

ETSI sees conformance testing and interoperability events as complementary and not competing activities. Rigorous conformance testing contributed to the success of GSM and this experience is now being applied to 3G, where 3GPP is developing comprehensive conformance test specifications for 3G UAs (terminals). In the IP-oriented world, where interoperability events are commonplace, a focused set of conformance tests can provide a useful complement to these events, as ETSI has already demonstrated in the area of IPv6 and SIP.

### 2.1 EP TIPHON™ and TTCN-3

The ETSI Project for *Telecommunications and Internet Protocol Harmonization Over Networks* (EP TIPHON™) is ETSI's main force in the standardisation of VoIP and multimedia calls and services across different kinds of networks (IP, SCN, PSTN etc.). ETSI STF176 is producing the TIPHON™ conformance test suites for H.225, H.245, H.248, BICC, SIP and OSP.

Nearly all ETSI conformance test specifications are written in the standardized test language TTCN. The latest version of this language, known as the *Testing and Test Control Notation version 3* or TTCN-3 [3], has been developed by the ETSI Technical Committee on *Methods for Testing and Specification* (TC MTS). In addition to the core language TTCN-3 has a tabular format [4] and a graphical format [5].

TTCN-3 can be used for many types of reactive testing and is not necessarily restricted to conformance testing. Typical areas of application are protocol testing, including mobile and Internet protocols, supplementary service testing, module testing, testing of CORBA based platforms, testing of APIs etc.

The syntax of TTCN-3 is similar to that of modern programming languages, making it attractive to engineers and developers familiar with those environments, including those from the datacom world. The language includes features that directly simplify some of the problems identified in this paper. For these and other reasons (see the TIPHON™ Technical Report on the use of TTCN-3 [6]) EP TIPHON™ WG6 chose TTCN-3 for writing the SIP and OSP test suites.

Currently there are over 430 SIP Test Cases in TTCN-3, while for OSP, which has less requirements, the number is around 30. These test suites concentrate on 'pure' SIP and OSP. Future versions of the tests will take into account the upcoming TIPHON™ profiles for both protocols. The H.225, H.245, H.248 and BICC tests are written in TTCN-2. This is mainly due to timing (the work was started several months prior to the publication of TTCN-3) and the fact that they are 'traditional' protocols (for example H.225 is very close to Q.931). It is also more likely that the test systems for these protocols will be based on TTCN-2.

### 3 Considerations when Testing SIP

SIP has very simple dynamic behaviour. The TTCN-3 communication and timer mechanisms etc. are entirely adequate to specify the exchange of SIP messages. For testing, the main problem is that SIP messages are text-based and do not have the explicit structure that is normally found in telecommunication protocols. In talking of SIP, one often hears "parsing SIP messages". This forces answering the question "Where does syntax checking stop, and protocol testing begin?". It is not the purpose of this paper to argue the merits of one approach over the other. However, it does mean that when writing requirements-based tests we are faced with a fundamental dilemma: *control* versus *readability*.

The TIPHON™ test team decided that it is necessary to give the messages at least some level of structuring. Highly structured SIP data will give a good degree of control but will probably lead to a test suite that is difficult to read (at least for someone who is used to reading text-based messages). When used wisely, pattern matching (using regular expressions) can assist readability but provides less control and less explicit debugging.

NOTE: In researching this issue, we found a regular expression that checks a BNF-conforming e-mail address (with a few simplifications) to be 6,598 ASCII bytes without any formatting or symbolic naming. This is approximately one page of pure regular expression fully occupying each line of the page - not exactly readable and in a testing context very hard to debug! Regular expressions are a powerful and useful feature when testing SIP, but they need to be used with discretion.

STF 176 has tried to go for the middle ground. In the current test suite the SIP messages are represented by over 100 structures representing the different SIP messages and headers. These are defined in a separate TTCN-3 module. Wide use is made of template parameterisation but there is relatively little pattern matching. For example:

```
template REGISTER_Request REGISTER_Request_r_1 :=
{
  requestLine :=
  {
    method := REGISTER_E,
    requestUri := ?,
    sipVersion := SIP_VERSION
  },
  reqHeader := *,
  messageBody := *
}
```

NOTE:  
A TTCN-3 template specifies the expected contents of an incoming SIP message. Templates can be substructured.

In TTCN-2 a template is known as a constraint.

However, as we gain more experience and feedback from implementers and users updates of the test suite may tend towards a less structured approach with more use of regular expressions. This debate is still open. For example it can be questioned whether the method field in the above shown example is necessary as the type of the template (REGISTER\_Request) already implicitly defines the method that shall be indicated in the method field. STF 176 has tried to find the balance between readability and complexity and it was decided for example that this type of information, although potentially superfluous will increase the readability and therefore the acceptance of test results.

The open specification of the SIP message contents also increases complexity. A single message type, such as REGISTER, may be present in several forms, each of which could be tested. Another increase in complexity is syntax-related; e.g., there are several possibilities in a text-based message to indicate a new line. These line terminators do not have to be the same across one message. Another example is the possibility of different formats for the same e-mail address; e.g., [john.doe@etsi.fr](mailto:john.doe@etsi.fr) and [john.doe@etsi.fr](mailto:john.doe@etsi.fr) are both acceptable.

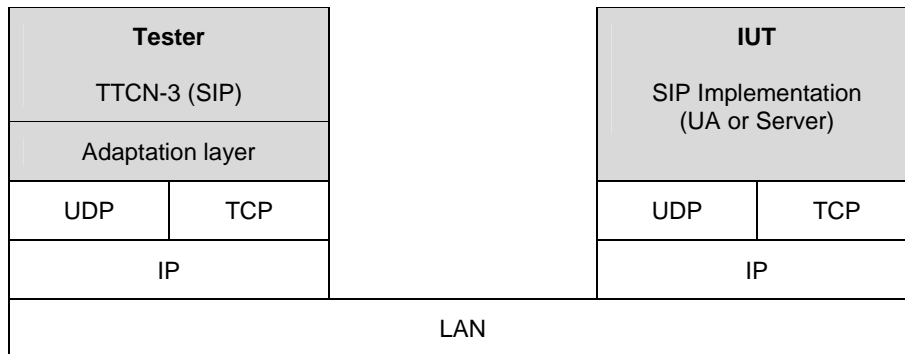
Given limited resources a decision has to be made on how much needs to be explicitly tested and how much can be left to the test system. The following section looks at these issues.

#### 3.1 SIP Test Architecture

The conceptual TIPHON™ SIP test architecture has two basic components: the Test Execution Environment that contains the SIP-specific TTCN-3 test suite itself and the Test Adapter that contains elements inherent to SIP that can be excluded from explicit testing such as coding, decoding, encoding rules, syntax productions, segmentation and reassembly, etc.

In essence, the Test Adapter extracts essential (to the test writer) SIP elements from a SIP message and passes them up to the test suite as values of TTCN-3 types for analysis. The adapter also receives these essential elements from the test suite and formulates them into a complete SIP message to be passed on to the lower protocol layer (TCP or UDP).

Such a tester-defined architecture allows definition of the degree of abstraction of the test suite. The advantages of this architecture are that the Test Execution Environment matches the level of abstraction in the test purposes and that the test writer's effort is considerably reduced. Also, the coding/decoding issues are passed on to the test equipment manufacturer who has better experience and tools in this domain.



**Figure 2:** Abstract SIP testing architecture.

Because a two-component test architecture was chosen, debate has naturally focused upon what the ATS (Abstract Test Suite) should contain; in other words, the degree of abstraction shown in the ATS versus the amount of pre/post-processing that the Test Adapter should do. On the one hand, if the Test Adapter contains more "lower level" functions, the test writer will have less to write in the ATS, while the test equipment manufacturer will have more to incorporate. This brings its own problems of maintenance and general portability of the test suite across different test platforms.

On the other hand, there will be less items tested in an implementation to determine its conformance to the specification. A requirement can become "buried" in the Test Adapter where, naturally, there is no test case nor verdict for the requirement. Thus, one could conceivably have an implementation that passes conformance testing but still contains a non-conforming specification function that was not tested because the function was abstracted into the Test Adapter.

NOTE: The approach taken by STF 176 for the definition of types and templates is that they do not contain syntactic delimiters and are canonicalized, i.e., where possible more abstract data types can be used and where this is not desirable, only one syntactic variation will be used for *send* templates. Analogously, different possible syntactical variations of received messages are reduced to one abstract presentation. Thus, templates describe the *information* and not their *presentation*. This is the approach frequently taken in traditional telecommunication systems. However, the precise distinction between information and presentation of human-readable protocols like SIP vanishes.

### 3.2 SIP Test Configurations

The TIPHON™ SIP test suite considers three kinds of entities as possible IUTs (Implementation Under Test):

- User Agent behaving as a Client or Server.
- Proxy (outbound and simple proxy).
- Redirect server.

The corresponding test purposes match this structure, with groups to reflect Registration (RG), Session (SS), Messaging (MG) and Transport and further sub-grouping to cover Call Establishment (CE), Session Modification (SM) and Call Release (CR). Further groups divide the tests into valid and invalid behaviour.

The number of test cases increases significantly because the system under test has several configurations. For example, if there are, say, five different configurations that receive a REGISTER message and must respond with an OK message, a test needs to be written for each of the five configurations even though the message exchange and its parameters are identical.

A sample of a valid (V) messaging (MG) test case for a SIP registrant (RT) is shown below. For simplicity the underlying declarations are not shown.

```

testcase SIP_MG_RT_V_008()
runs on SipComponent
system SipInterfaces
// Status: Mandatory
// SUT: A UA.
// Precondition: None
// Ref: 9.1.1 [1]
// Purpose: Ensure that the IUT having sent a REGISTER request on receipt of
// a valid Success (200 PERFECT) response with an unknown reason phrase
// considers to have received a Success (200 OK).
{
    var REGISTER_Request v_REGISTER_Request;
    var integer i :=0;

    initUDPPort (mtc, system);
    activate(defaultRGRT());

    havingSentARegister(v_REGISTER_Request);
    //Send a 200 PERFECT Answer to the UA
    SIPP.send (Response_2xx_REGISTER_s_1(200, "PERFECT",
                                         v_CallId, v_CSeq,
                                         v_From, v_To, v_Via,
                                         v_Contact, SHORT_REGISTRATION ))
              to sent_label;

    checkConsideredToBeRegistered();

} // end testcase SIP_MG_RT_V_008

```

## 4 Considerations when Testing OSP

The OSP is also a text-based and datagram-oriented protocol but it is found at a higher layer than SIP and must always be run over XML. The OSP messages are transported as one or more components in an XML message. If there is more than one component, this is called a multi-message.

The protocol itself is rudimentary consisting of 7 independent client-server transactions. There is no state machine, no timers or counters and only valid behaviour is defined. One or more OSP messages can be contained in the same XML data message. Thus, the test equipment as a client can receive responses to the messages it sent in an indeterminate order from the server. An OSP message's data structure is relatively straightforward with the complication that, at times and under indeterminate conditions, zero, one, or more data elements may be present in the message. All this can be handled in TTCN-3.

OSP also defines security and authentication mechanisms, optionally encoded using ASN.1. Encoding/decoding adds complexity because three schemes are possible: XML (ASCII or binary) or ASN.1 (PER). Currently, security features are not tested but will probably be addressed in later versions of the test suite. This is not a problem in TTCN-3 which is harmonized with ASN.1

For OSP the problem of structuring versus pattern matching is not the issue it is in SIP. XML and the OSP DTD (Data Type Definition) fully define the structure and content of each OSP message. Mapping the OSP XML messages to equivalent TTCN-3 types is a relatively simple task. STF176 has tried to follow a generic approach that works for OSP but which could be extended to a general TTCN-3 - XML mapping. For example:

```

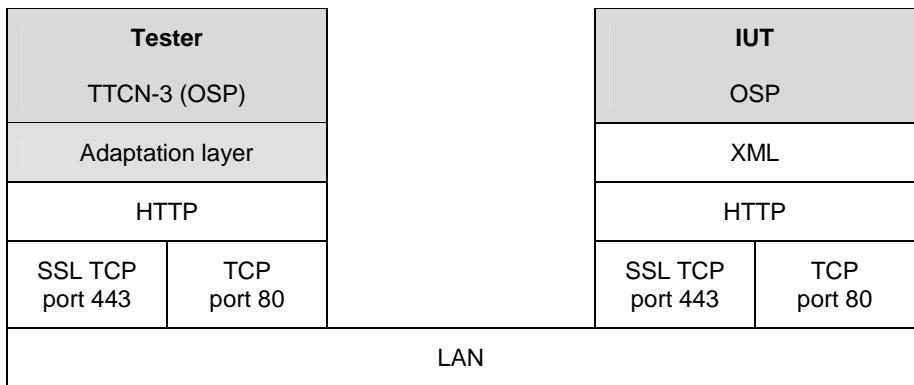
<!ELEMENT PricingIndication ( Timestamp, SourceInfo, DestinationInfo, Currency, Amount,
Increment,
  Unit, Service, ValidAfter, ValidUntil )>
<!ATTLIST PricingIndication componentId ID #REQUIRED>
type record PricingIndication
{
  ID_DTD          msgIdField,
  CDATA_DTD       randField,
  ID_DTD          compIdField,
  Timestamp       timestamp,
  SourceInfo      sourceInfo,
  DestinationInfo destinationInfo,
  Currency        currency,
  Amount          amount,
  Increment       increment,
  Unit            unit,
  Service         service,
  ValidAfter      validAfter,
  ValidUntil      validUntil
} with {encode (msgIdField, randField, compIdField) "XML_ATTRIBUTE"};

<!ELEMENT Timestamp (#PCDATA)>
<!ATTLIST Timestamp critical (true | false) "true">
type record Timestamp
{
  ENUM_DTD        criticalField ("true","false"),
  PCDATA_DTD      valueField
} with {encode (criticalField) "XML_ATTRIBUTE"};

```

#### 4.1 OSP Test Architecture

The conceptual TIPHON™ OSP test architecture includes an OSP-specific Adaptation Layer below the TTCN-3 layer. This is the equivalent to the SIP Test Adapter discussed earlier. This layer performs the following functions: coding and decoding of the three different XML formats and separation of multi-messages into single message units to be passed to the TTCN-3 ATS. This latter abstraction matches the test purposes that deal only with single messages.



**Figure 3:** Abstract OSP testing architecture.

A discussion between test writers, similar to the SIP debate on the level of abstraction done by the adaptation layer, occurred for OSP as well. The handling of multi-messages was an important topic. The decision made was that the Adaptation Layer would break the multi-messages into individual messages.

When writing the tests for OSP two TTCN-3 features were found to be especially useful:

- A response to an OSP routing request may require, for example, 1 or 100 destinations. The interleaving execution algorithms of TTCN-3 allow more elegant processing of these responses compared to TTCN2 loop structures.
- TTCN-3 also provides an elegant encoding/decoding method selection mechanism to convert the OSP messages into XML.

## 4.2 OSP Test Configurations

The OSP Test Suite Structure has two levels. The first level is divided into two IUT types: an OSP client or an OSP server.

At the second level, each IUT type contains procedures : General OSP requirements (GOS), Pricing Exchange procedure (PRI), Authorization Exchange procedure (EXC), Authorization Validation procedure (VAL), Reauthorization Exchange procedure (REA), Usage Reporting procedure (REP), Subscriber Authentication procedure (SUB), Capabilities Exchange procedure (CAP).

Because OSP does not define error recovery behaviour there are no invalid tests.

The following TTCN-3 fragment shows a valid (V) test case for an OSP server (SV) for Capabilities Exchange (CAP). For simplicity the underlying declarations are not shown.

```
testcase    OSP SV CAP BV 001() runs on MTC OSP
// Selection: Pics Table I.1/2 [1] AND Pics Table I.20/13 [1] AND Pics Table I.20/1
// 4 [1]
// Precondition:    None
// Ref:             Clause 6.2.13 [1]
// Purpose:         Ensure that the IUT, on receipt of a CapabilitiesIndication with an OSP
// Version element that contains a supported OSP version,
// sends a CapabilitiesConfirmation with a componentID attribute associated to the
// CapabilitiesIndication, with the critical attribute set to FALSE, with a Timestamp,
// Status and an OSP version element.
// The Timestamp element shall contain the definition of the time according to ISO
// 8601 [8]. The Status element shall contain the Code element. The Code element sh
// all contain the code value 2xx. The critical attribute of the OSPVersion element
// shall be set to FALSE. The OSPVersion element shall include a version number eq
// ual or less than the version number proposed in the CapabilitiesIndication.
{
    var default TCV_Def:= activate( DF_Server());

    var CDATA_DTD TCV_Rand:="000";
    var PCDATA_DTD TCV_Timestamp:="000";
    var ID_DTD     TCV_MsgId:="0";
    var ID_DTD     TCV_CompId:="0";
    var ENUM3     TCV_Result:={"0","0","0"};
    var PCDATA_DTD TCV_Code:="000";

    var PCDATA_DTD TCV_OSPVersion;
    var CapabilitiesConfirmation TCV_CapConf;

    TCV_Result:= getIds();
    TCV_MsgId:= TCV_Result.msgId1;
    TCV_CompId:= TCV_Result.compId1;
    TCV_Rand:= getRand();

    OSP_Init();

    Ll.send(CapInd_S1(TCV_MsgId, TCV_Rand, TCV_CompId, PX_OSPVersion_Client));
    TAC.start;
    alt
    {
        [] Ll.receive(CapConf_R1(TCV_MsgId, TCV_CompId)) -> value TCV_CapConf
        {
            TCV_Code:= TCV_CapConf.status.code.valueField;
            TCV_OSPVersion:= TCV_CapConf.ospVersion.valueField;
            TAC.stop;
        }
        [] TAC.timeout
        {
            verdict.set(fail);
            stop;
        }
    }

    if ((200 <= str2int(TCV_Code))and (str2int(TCV_Code) <= 299))
    {verdict.set(pass);}
    else
    {verdict.set(fail);}
} // end testcase    OSP_SV_CAP_BV_001
```

## 5 Actual TTCN-3 Test Platforms for SIP and OSP

The structure of a TTCN-3 test system has been described in the TTCN-3 Runtime Interface specification defined by ETSI [7].

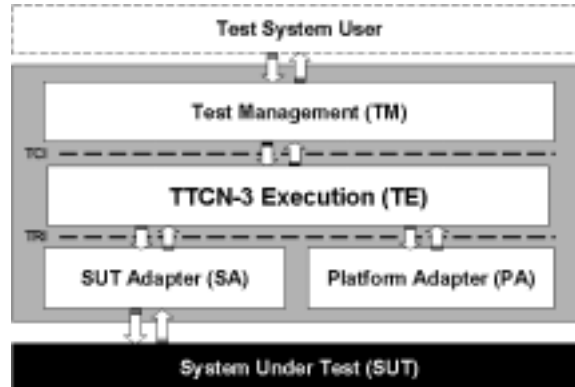


Figure 4: General Structure of a TTCN-3 Test System.

Figure 4 displays the general structure of a TTCN-3 test system. The test system is conceptually designed as a set of interacting and communicating entities. Each entity implements one certain aspect required by a test system. The Test Management (TM) entity is responsible for the overall test management. The Test System user interacts at a user interface with the TM and uses the general test execution management functionality of the TM, e.g., the preparation of the test system for a test execution, the collection of final verdicts, and propagation of module parameters to the TTCN-3 Executable (TE) via the Test Control Interface (TCI).

The TTCN-3 Executable (TE) is responsible for the implementation of the functionality as defined in the TTCN-3 module, like control of test execution, propagation and matching of test events, and management of TTCN-3 test configurations. The remaining aspects of the test system that are required to execute a TTCN-3 test suite are represented by the SUT Adapter (SA) and the Platform Adapter (PA) entities. Communication with the SA and PA is performed via the TTCN-3 Runtime Interface. The implementation of the test suites communication means is performed by the SA while the PA implements TTCN-3 external functions and provides the test system with a unique notion of time.

Currently the TTCN-3 type and value representation on the test system has not been specified. [7] specifies that all interaction at the TRI shall use encoded data only, and therefore the encoding of TTCN-3 messages and the decoding of received messages has to be performed within the TE by a codec.

In order to provide a test platform for the SIP and OSP test suite both the codecs and the SA had to be implemented for a specific test system. For both test platforms a Java based TTCN-3 execution environment was chosen. Besides its availability the transportability of the resulting executable test suites on different operating systems and available freeware modules to support the implementation of the codec and the SA were the main reasons for choice.

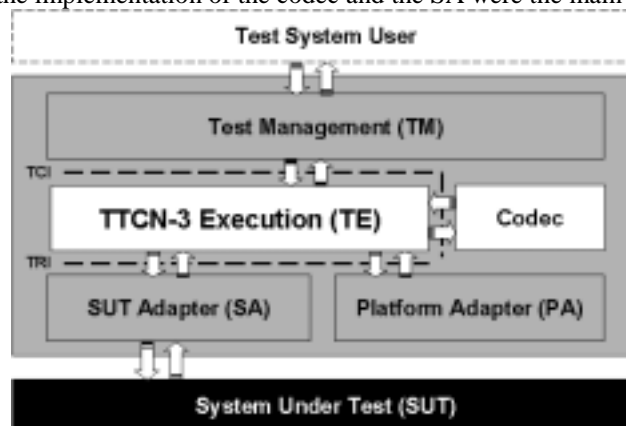


Figure 5: Decomposition of the TE.

Figure 5 shows a possible decomposition into the core TE and a codec entity. Although the TCI interface as depicted in Figure 5 is currently a proprietary addition to the TTCN-3 Runtime Environment. An ETSI work group has been established to standardize the TCI interface and therefore enable the user to write portable codecs.

## 5.1 The SIP Platform

In Section 3 it has been shown that finding the right level of abstraction was one of the major tasks in defining the SIP test suite. The design decisions made there are directly related to the implementation of the codec for the SIP platform. The adaptation layer introduced in section 3 consists of a codec and an SA implementation.

To cope with the syntactical complexity of SIP messages, a modified freeware SIP parser, taken from the NIST IP telephony project (<http://www.antd.nist.gov/proj/iptel>) has been adapted to act as a codec. The codec validates syntactically the incoming SIP messages and constructs the appropriate TTCN-3 values needed within the TE. In addition the SIP parser has been used to encode the TTCN-3 templates into SIP messages suitable for sending. The parser provided therefore a canonical view on SIP messages and abstracts from the actual syntactic variations.

NOTE: This property, i.e., having syntactical variations in the encoding of a message is common to many higher-layer IP based protocols but differs from common practice in traditional telecommunication system where a precise encoding for messages is defined.

To be able to handle syntactic invalid messages special TTCN-3 values are generated when the parser finds syntactic errors. A single syntactic error renders therefore a complete message invalid. To address sending syntactically invalid messages or messages with syntactic variations, dedicated TTCN-3 values will be used. They will be transmitted "as is", i.e., without using the SIP parser that canonicalizes the send events.

The SA, the second entity needed in order to implement a specific test suite, had to implement the specific operations as defined by the Java-language mapping in [7]. ETSI's first experiences with the usage of the TRI for a specific implementation had shown that the TRI interface fulfils the expectations, to provide all the functionality needed to implement the communication with the SUT. The use of the TTCN-3 `map()` and `unmap()` operations and their TRI counterparts has shown to be an elegant way in establishing and configuring on-the-fly communication means without using the traditional way of abstract services primitives well known from telecommunications.

The actual communication is done by using standard communication operations provided by the Java SDK. Therefore an easy and reusable test access for the testing of IP based protocols could be achieved.

The SIP test platform was initially developed on a Solaris 5.6 workstation using a Java 2 SDK. Besides the usage of the TTCN-3 Runtime Environment and the above mentioned freeware NIST SIP parser no other external modules were needed for the test platform.

As all parts are implemented in pure Java, a test platform has been designed and implemented that could run on virtually every test device that provides a Java Virtual Machine. For the time being the test platform has been executed on PCs running Linux and Windows 2000, and on Solaris 5.6 running on Sun Sparcs and Intel CPUs. Implementations of different vendors have been used as IUT, freely available and commercial implementations, User Agents and Servers.

## 5.2 OSP Platform

Like for the SIP platform the implementation of an OSP test platform requires the implementation of codec and an SA.

As it has been shown in section 4 a canonical mapping for OSP messages defined in XML to TTCN-3 has been used. With such kind of mapping a codec has been implemented that uses the information of the TTCN-3 abstract test suite to encode TTCN-3 templates into XML messages without special protocol knowledge.

The same of course applies also for the decoding of received XML messages carrying OSP information. To handle the different presentation formats allowed in XML the codec uses a XML DOM-conformant (Document Object Model) implementation to achieve abstraction from the syntactical presentation of XML.

For the implementation of the SA either built-in operations of the used Java SDK or publicly available packages have been used.

An experimental OSP test platform was built on PC/Windows 2000 using Java 2 SDK with the TTCN-3 runtime environment (TTrun) from TestingTech. TTrun implements the TTCN-3 Runtime Interface specified by ETSI [7]. TTrun also provides a TTCN-3 type and value interface so the above-described routines could be implemented. Freeware such as an XML parser and the Tomcat server was also used. The IUT was a remote OSP server kindly supplied by TransNexus, USA.

## 6 Validation of the Test Suites

One of the main concerns of STF 176 has been to validate the test suites before publication by ETSI. The test platforms described above are being used to validate both test suites. For SIP this is expected to be completed by May 2002, for OSP by June 2002.

Initial SIP conformance testing based on the TIPHON™ test suite has been made available at events such as SIPit and IMTC WinterOp. The SIP testing service was considered useful by many of the participants at the events by providing an initial demonstration of interoperability and conformance to the SIP RFC and by highlighting problems that were not detected by interoperability testing alone. Many manufacturers of ICT equipment are realising that including the possibility of conformance testing at interoperability events gives real added value to the entire process of validating products and standards.

A positive side effect could be observed from the sole existence of a conformance test suite. As specifications often leave room for interpretation the conformance test suite could give guidance for resolving interoperability issues. Typically the most restrictive interpretation are chosen, if possible. Therefore manufacturers can use the interpretation of the conformance test suite as guidance to resolve interoperability issues while still meeting conformance.

## 7 Conclusions

ETSI has long been committed to improving the quality of its standards and related products through validation and testing. To this end ETSI actively contributes to the development and standardization of modern testing techniques such as TTCN-3.

The work of STF 176 has been a good opportunity to evaluate TTCN-3. Our experience is that the language fulfils its promise of application to new areas of testing and that its modern syntax makes it more acceptable to the Internet community. TTCN-3 features such as the trigger mechanism, interleaving, regular expressions, interface mapping, address types and powerful type and template matching mechanism make TTCN-3 more adapted to expressing tests for SIP than TTCN-2.

NOTE: There is a significant investment in TTCN-2 test suites both in industry and in standardization bodies such as the ITU-T and ETSI. The purpose of TTCN-3 is to expand the use of the language into areas where it is currently not used. It is ETSI's policy that the two versions of the language will exist for several years to come. It is probable that technologies that have made extensive use of TTCN-2 in the past (such as ISDN, GSM, Hiperlan/2, IN and DECT™) will continue to do use TTCN-2 in order to preserve that investment. However, as more TTCN-3 test systems are developed it is expected that new ETSI test specifications will be written in TTCN-3. 3GPP are already preparing for the use of TTCN-3 for some test specifications. There are no plans for a wholesale upgrade of existing TTCN-2 specifications to TTCN-3, though this may be done on an as needed basis for individual test suites.

The use of TTCN-3 is still at an early stage and success of the language will be closely linked to the availability of TTCN-3 tools and compilers. We anticipate that 2002 will see a number of sophisticated TTCN-3 tools and test systems coming onto the market.

There are still open issues on style and the best way to present a readable test suite. These are expected to be resolved as we gain experience. Annex A of ETSI TR 102 026 [6] makes a first attempt at this. ETSI TC MTS is also planning to produce some simple guidelines in 2002.

Experiences from the implementation of the SIP and OSP codec lead to valuable input to the ongoing standardization work of the TCI interface that is currently under definition at ETSI.

Finally, we are seeing a growing interest from various communities on the use of conformance test suites in parallel with interoperability events. This is an activity ETSI will encourage in 2003.

## Acknowledgements

The authors of this paper thank EP TIPHON™, especially the TIPHON™ Chairman Mr. Hans van der Veer and the TIPHON™ WG6 Chairman Mr. Klaus Sambor, for their support and encouragement given to STF 176 during the development of all the TIPHON™ test suites (<http://portal.etsi.org/>).

Similarly, EP TIPHON™ WG6 thanks ACACIA (<http://www.acacia-net.com/>), FSCOM (<http://www.fscom.fr/>) and Testing Technologies (<http://www.testingtech.de/>) for the considerable additional voluntary effort they have contributed to the work of STF176.

## References

- [1] Draft IETF RFC 2543 bis04  
Session Initiation Protocol (SIP)
- [2] ETSI TS 101 321 V.2.1.1  
Telecommunications and Internet Protocol;  
Harmonization Over Networks (TIPHON);  
Open Settlement Protocol (OSP) for Inter-Domain pricing, authorization, and usage exchange
- [3] ETSI ES 201 873-1  
Methods for Testing and Specification;  
The Testing and Test Control version 3;  
Part 1: TTCN-3 Core Language
- [4] ETSI ES 201 873-2  
Methods for Testing and Specification;  
The Testing and Test Control Notation version 3;  
Part 2: TTCN-3 Tabular Presentation Format (TFT)
- [5] ETSI TR 101 873-3  
Methods for Testing and Specification;  
The Testing and Test Control Notation version 3;  
Part 3: TTCN-3 Graphical Presentation Format (GFT)
- [6] ETSI TR 102 026  
Telecommunications and Internet Protocol;  
Harmonization Over Networks (TIPHON);  
Study on the use of TTCN-3 for SIP and for OSP test specifications
- [7] ETSI TRI V4.2  
The TTCN-3 Runtime Interface (TRI)