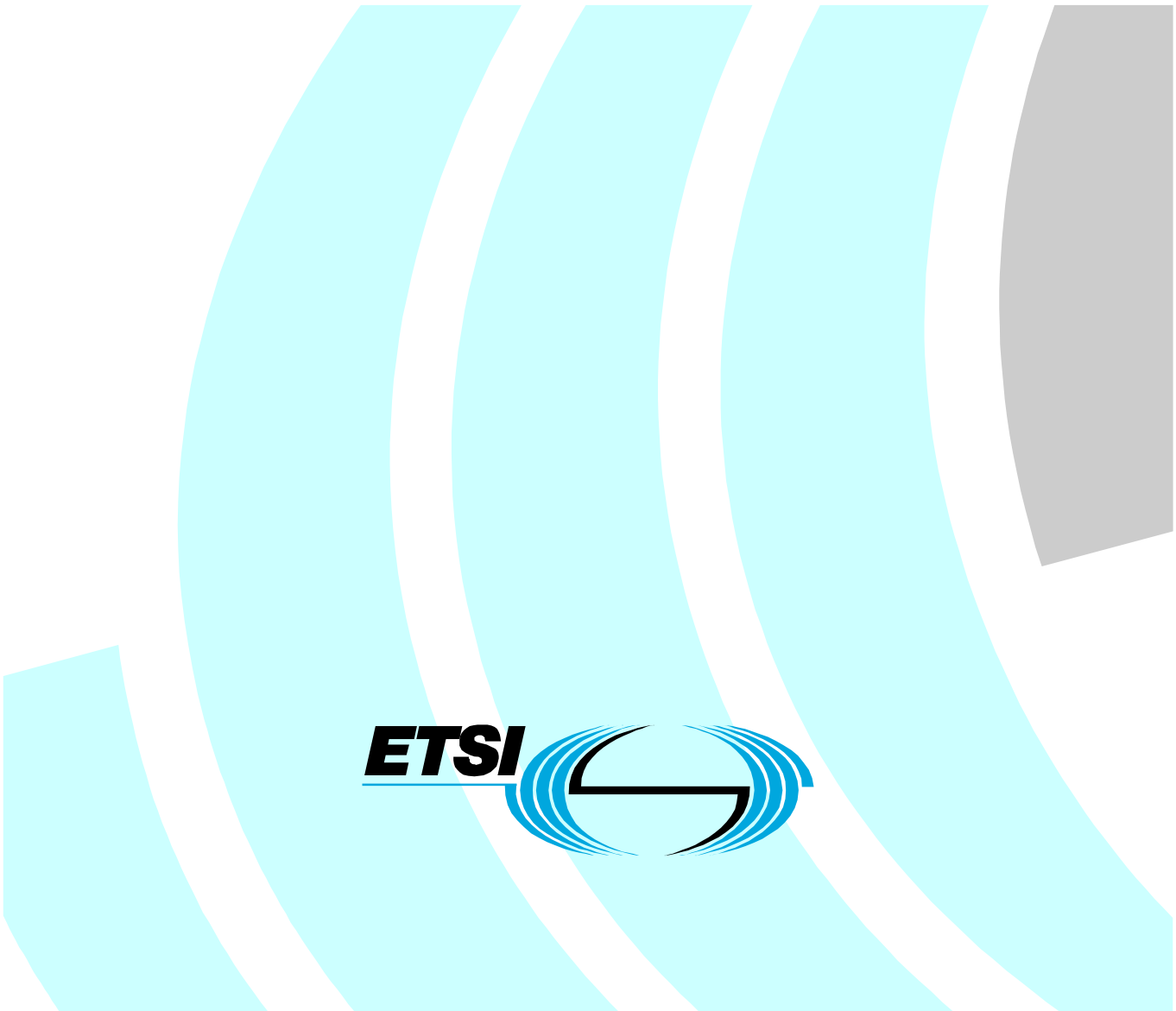
ETSI TS 102 027-2 V1.1.1 (2002-09)

Technical Specification

Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); Technology Compliance Specification; Draft IETF SIP RFC2543bis-04; Part 2: Abstract Test Suite (ATS) and partial Protocol Implementation eXtra Information for Testing (PIXIT) proforma specification



Reference
DTS/TIPHON-06021-2

Keywords
IP, PIXIT, SIP, telephony, testing, VoIP

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI Project Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON).

The present document is part 2 of a multi-part deliverable covering Technology Compliance Specification; Draft IETF SIP RFC2543bis-04, as identified below:

Part 1: "Test Suite Structure and Test Purposes (TSS&TP) specification";

Part 2: "Abstract Test Suite (ATS) and partial Protocol Implementation eXtra Information for Testing (PIXIT) proforma specification".

1 Scope

The present document specifies the Abstract Test Suite (ATS) and partial Protocol Implementation eXtra Information for Testing (PIXIT) proforma for the Session Initiation Protocol (SIP) as defined in Draft IETF SIP RFC 2543bis-04 [1].

The objective of this test specification is to provide a basis for conformance tests for SIP equipment giving a high probability of inter-operability between different manufacturer's SIP equipments.

Annex A provides the Tree and Tabular Combined Notation (TTCN-3) part of the ATS.

Annex B provides the Partial Protocol Implementation Extra Information for Testing (PIXIT) proforma of the ATS.

Annex C provides the Protocol Conformance Test Report (PCTR) proforma of the ATS.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] Draft IETF SIP RFC 2543bis-04 (2001): "SIP: Session Initiation Protocol".

[2] ETSI ES 201 873-1(V1.1.2): "Methods for Testing and Specification (MTS); The Tree and Tabular Combined Notation version 3; Part 1: TTCN-3 Core Language".

[3] ETSI ES 201 873-2 (V1.1.2): "Methods for Testing and Specification (MTS); The Tree and Tabular Combined Notation version 3; Part 2: TTCN-3 Tabular Presentation Format (TFT)".

[4] ISO/IEC 9646-4 (1994): "Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework, Part 4: Test realization".

[5] ISO/IEC 9646-5 (1994): "Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework, Part 5: Requirements on test laboratories and clients for the conformance assessment process".

[6] ISO/IEC 9646-6 (1994): "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 6: Protocol profile test specification".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in [1] and [2] apply.

3.2 Abbreviations

For the purposes of the present document, the abbreviations defined in [1], [2] and the following apply:

ATM	Abstract Test Method
ATS	Abstract Test Suite
IUT	Implementation Under Test
MTC	Main Test Component
PCTR	Protocol Conformance Test Report
PDU	Protocol Data Unit
PICS	Protocol Implementation Conformance Statement
PIXIT	Protocol Implementation eXtra Information for Testing
PTC	Parallel Test Component
SUT	System Under Test
TA	Test Adapter
TC	Test Cases
TE	TTCN-3 Execution Environment
TP	Test Purpose
TS	Test System
TSS	Test Suite Structure
TTCN	Testing and Test Control Notation version 3

4 Abstract Test Method (ATM)

This clause describes the ATM used to test Draft IETF SIP RFC as defined in [1].

4.1 Network architecture

The basic SIP network architecture is defined in figure 1. The ATS defines test cases for the IUT being in the role of each displayed entity.

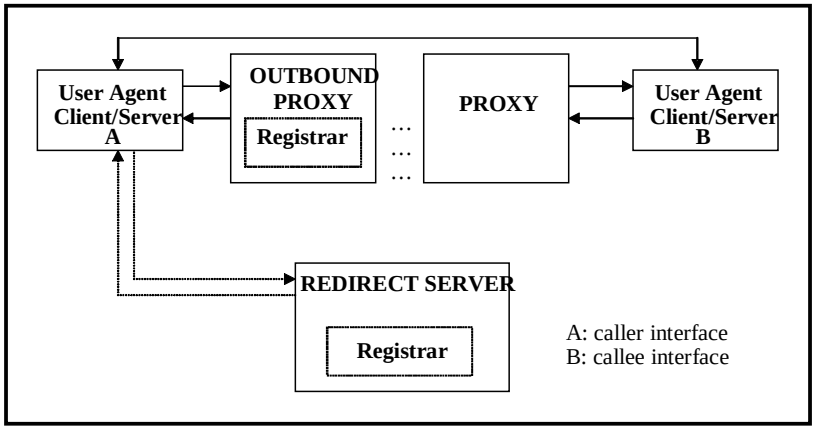


Figure 1: SIP network architecture

4.2 Protocol architecture

The Implementation Under Test (IUT) for which this Test case specification applies consists of the SIP protocol (see figure 2).

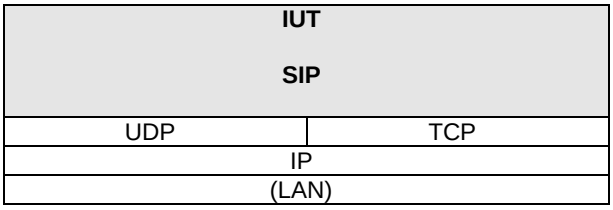


Figure 2: SIP protocol architecture

4.3 Test system architecture

Test systems that implement this ATS shall conform to the requirements as defined in this clause.

4.3.1 Structure

An abstract architecture for a Test System (TS) implementing a TTCN-3 ATS is displayed in figure 3.

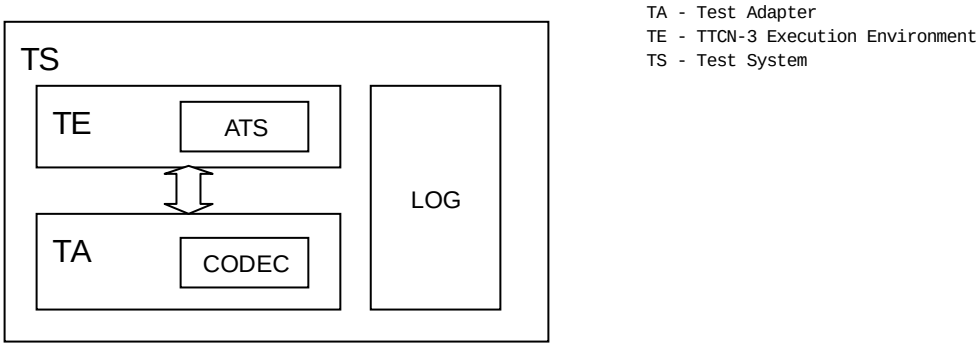


Figure 3: Abstract Test System Architecture

The TS consists at least of a TTCN-3 Execution Environment (TE) that implements the TTCN-3 ATS as defined by the semantics definition in TTCN-3 [2] and a Test Adapter (TA), which performs the adaptation of the "abstract" test concepts like sending and receiving to a concrete test device and configuration.

The encoding/decoding (CODEC) as well as the logging (LOG) functionality of TS shall comply with the conventions as defined in the clauses below.

4.3.2 Codec conventions

SIP is a text-based protocol that allows different syntactical presentations of the same information. In general, an implementation of this ATS should use a codec to parse received message and encode structured messages. This codec is not part of the ATS. However, all encoded messages, i.e. messages as they are transmitted or received shall be logged.

The following terms shall be used for the conventions defined below:

Syntactic delimiter	syntactic delimiters are characters like "=" or ";" that are used to separate.
LWS	linear white spaces as defined SIP [1].
Parameter name	name of header parameters as defined in SIP [1].

Parameter value	the value of a parameter as defined in SIP.
Undefined method	an undefined method is a method other than: "INVITE", "ACK", "OPTIONS", "BYE", "CANCEL" and "REGISTER".
Undefined header	an undefined header is a header other than general-header, entity-header, request-header and response header as defined in SIP [1].
Unexpected header	an unexpected header is a header, which shall not be present in a specific request message. This definition complies with the definition of NOT APPLICABLE in SIP [1]/10 for request messages.

4.3.2.1 Decoding conventions

TTCN-3 fields should not contain syntactic delimiters like white space, semicolon, equal characters etc. Instead the information provided by a parser shall be used to build the decoded message in TTCN-3. Decoded messages shall use the TTCN-3 enumeration types where ever appropriate, e.g. for the method and the header field name.

For **`charstring`** fields the following decoding rules shall be applied:

1) Subsequent LWS shall compress to a single space character " ".

2) Decoded parameter names shall use only lower case letters.

3) Parameter values containing an integer value shall be decoded to a TTCN-3 integer value where a TTCN-3 **`integer`** type is used for a SIP parameter value.

The following decoding rules shall be applied to each received message in the following order:

1) In case a request message indicating an undefined method is received by the test system, the message shall not be passed to the TE. However the message is subject to logging as defined in clause 4.3.3 ("Logging conventions").

2) In case an undefined header has been received the header field shall be decoded as **`UndefinedHeader`** field.

3) In case an unexpected header has been received a **`RawMesage`** shall be constructed and passed to the TE.

4) In any other case where a received message cannot be decoded according to the ATS's type definitions a message of type **`RawMessage`** shall be constructed and passed to the TE.

4.3.2.2 Encoding conventions

Encoders shall follow all encoding rules that are defined in SIP [1] as MUST or SHOULD rules for the sending of structured templates. This applies in particular to but it is not restricted to clause3 "SIP Message Overview" and 10.5 "Header Field Format" of [1].

Templates of type **`RawMessage`** shall be sent to the SUT without any modification.

4.3.3 Logging conventions

As the ATS defines on an abstract level the message exchange between TS and SUT the messages encoded messages send and received shall be logged. The TS shall provide access to this log.

5 Untestable Test Purposes (TP)

This clause gives a list of TP, which are not implemented in the ATS due to the chosen ATM or other restrictions: None.

6 ATS conventions

The ATS conventions are intended to give a better understanding of the ATS, but they also describe the conventions made for the development of the ATS. These conventions shall be considered during any later maintenance or further development of the ATS.

The ATS conventions contain two clauses, the naming conventions and the implementation conventions. The naming conventions describe the structure of the naming of all ATS elements. The implementation conventions describe the functional structure of the ATS.

6.1 Naming conventions

6.1.1 Type definitions

This clause describes the naming conventions used for structured and unstructured types as well as for the field names of structured types.

6.1.1.1 Generals

Type identifiers use mixed cased with the first letter of each internal word capitalized. In case single internal words consist completely of uppercase letter the following word is separated with an underscore "_".

EXAMPLE 1:
```
RequestLine
ACK_RequestHeader
```

Field identifiers use mixed cased with a lowercase first letter. Internal words start with a capital letter.

EXAMPLE 2:
```
requestLine
```

In case type and identifier names should give a hint on their structure the term describing the structure should be separated with an underscore ("_") at the end of the name.

EXAMPLE 3:
```
GenericParam_List
ContactBody_Choice
```

6.1.1.2 PDU Type Definition

Names of types used as PDUs follow the general conventions as defined in clause 6.1.1.1. In case the PDU type denotes a SIP request message the method name for the request as defined in SIP [1] is prepended.

EXAMPLE: `ACK_Request`

6.1.2 Template definition

Template identifiers consist of the type name, an identifier denoting whether the template is for sending or receiving and a sequential number.

EXAMPLE: `ACK_Request_r_1` denotes a template from type `ACK_Request` that is intended for reception.
`GenericParam_List_s_25` denotes a template from type `GenericParam_List` that is intended for sending.

The sequential number is used only to distinguish between templates for the same type and direction and includes no other information.

6.1.3 Constant declarations

Identifiers for either internal or external constants, use only uppercase letters. Internal words are separated by an underscore ("_").

EXAMPLE: `SIP_VERSION`

6.1.4 Enumeration declarations

While identifiers for the enumeration type follow the conventions as defined in clause 6.1.1.1. "Type definitions", enumerations elements use only uppercase letters, which are suffixed by "_E" to distinguish them from constants. Internal words are separated by an underscore character ("_").

EXAMPLE: Enumeration type
```
HeaderType
```

Enumeration value
```
FROM_E
```

6.1.5 Module parameter declarations

Identifiers for module parameters follow the general rules as defined in clause 6.1.3 "Constant declarations". Numbers are separated from words using the underscore character "_".

EXAMPLE: CAP_1

6.1.6 Variable declarations

Identifiers for variables follow the general rules for field names as defined in clause 6.1.1 "Type definitions".

6.1.7 Function declarations

Identifiers for either internal or external functions use mixed case with a lowercase first letter. Internal words start with a capital letter.

EXAMPLE: strToInt()

6.1.8 Test Case declarations

6.1.8.1 General

All test cases are listed in the order in which they appear in the Test Suite Structure (TSS) and TP document. Grouping is used to reflect the TSS.

6.1.8.2 Test Case (TC) identifier

The identifier of the test case is constructed in the same way as for the test purpose described in TS 102 027-1. The identifier of a TC is built according to table 1.

Table 1: TP identifier naming convention scheme

Identifier: **<protocol>_<main functionality>_<role>_<functionality>_<type>_<nn>**	
<protocol>	SIP
<main functionality>	Registration (RG), Session (SS), Message Parsing (MP
<role> =	Registrant (RT), Registrar (RR) Originating Endpoint (OE), Terminating Endpoint (TE), Proxy (PR), Redirect (RD)
<functionality> (optional)	Call Establishment (CE), Call Release (CR)
<type>	Valid Behaviour (V), Invalid Behaviour (I), Inopportune Behaviour (O)
<nnn> =	sequential number (001-999)

If variants for one TP have been specified a lower case letter, starting with "a" has been appended to each of the variants.

EXAMPLE:
```
SIP_RG_RT_V_001
SIP_RG_RT_V_004a
SIP_RG_RT_V_004c
```

If more than 26 variants have been defined an additional lower case letter, starting with "a" will be appended.

6.1.9 Timer declarations

Two types of timers can be identified:

1) Standardized:

   - Those defined in SIP [1], e.g. `T1`. They use exactly the same name as in the standard.

   As there is a tolerance margin accepted for these timers, three values are needed:

   - The maximum value allowed, which will use the suffix "`_max`";
   - The minimum value allowed, which will use the suffix "`_min`";
   - The value actually implemented, with no suffix;

EXAMPLE 1: `T1 _max`, `T1_min`, and `T1`.

2) Not standardized:

   - Those not defined in the protocol standard, i.e. for execution use, e.g. a timer waiting for a response. These timers begin with the prefix "`T_`", followed by a string in lowercase letters.

EXAMPLE 2: `T_resp` represents a timer for controlling the response time of the IUT.

6.1.10 Group names

Group names follow the same general conventions as defined in clause 6.1.1.1.

EXAMPLE: SubtypesTemplateDeclarations.

Where appropriate group names reflect the hierarchic group structure.

6.2 Implementation conventions

6.2.1 Type definitions

The following clause gives an overview on the mapping of SIP messages and structures as defined in SIP [1] and their corresponding TTCN-3 types.

6.2.1.1 Messages

Distinct types have been defined for SIP request and response messages, as they have a different internal structure. Messages are defined as a record structure containing three fields, a request/status line field, a header field and a message body field.

EXAMPLE: A generic request message is:

```
type record Request {
    RequestLine     requestLine,
    RequestHeader   reqHeader,
    charstring      messageBody optional
}
```

For syntactic delimiters, like Carriage Return Line Feed (CRLF), colon ":", etc. no extra fields are defined as they are already removed by the codec.

The definition of invalid messages for sending and receiving is discussed in clause 6.2.1.1.3.

6.2.1.1.1 Request messages

For each method defined in SIP an own structured type has been defined. This is analogous to the TTCN-2 PDU type. As the request line is for each request message identical no extra types have been defined than the generic one.

EXAMPLE: A type for a message indicating the ACK method is:

```
type record ACK_Request {
    RequestLine         requestLine,
    ACK_RequestHeader   reqHeader,
    charstring          messageBody optional
}
```

6.2.1.1.2 Response messages

For all possible valid responses one generic type has been defined.

EXAMPLE: The generic response message:

```
type record Response {
    StatusLine      statusLine,
    ResponseHeader  resHeader,
    charstring      messageBody optional,
    Payload         payload optional
}
```

The payload field contains the whole message as it has been received in its text format.

6.2.1.1.3 Raw messages

For defining syntactic torture or syntactic invalid messages a distinct type **RawMessage** has been defined. This type is defined as a **charstring**. Messages using this type define exactly how the message shall be transmitted, thus giving the possibility to define the message on a character level.

EXAMPLE: A torture message:

```
template RawMessage rawMessage_s_1 =
    "INVITE sip:joe@foo.com SIP/2.0" & CRLF &
    "TO     :   " & CRLF
    " sip:joe@foo.com   ;" & TAB & "    tag   = 1918181833n" & CRLF &
```

```
    "Via   : SIP   /   2.0" ;
```

The **RawMessage** type has not been used to describe valid receiving templates, as each received message will be parsed by the codec. Therefore, valid receiving templates will always use structured messages.

The reception of a **RawMessage** indicates that the codec was unable to parse the received message according to the type definitions given in this specification. Typically the RawMessage will match to the TTCN-3 otherwise construct, i.e. the receive construct that matches everything.

6.2.1.2 Headers

The following clause defines the mapping of the header field section of a SIP message as defined in [1].

For each message a distinct message header type has been defined. As the header fields can appear in any order in a SIP message header types are defined as **set**s. The message header type includes all possible header fields that are allowed to be present in the in the header field section of the message. Optional header fields are using the **optional** keyword.

EXAMPLE 1: A header structure for an ACK request.

```
type set ACK_RequestHeader
{
    Allow               allow optional,
    Authorization       authorization optional,
    CallId              callId,
    Contact             contact optional,
    ContentDispostion   contentDisposition optional,
    ContentEncoding     contentEncoding optional,
    ContentLanguage     contentLanguage optional,
    ContentLength       contentLength,
    ContentType         contentType optional,
    CSeq                cSeq,
    Date                date optional,
    Encryption          encryption optional,
    From                fromField,
    MaxForwards         maxForwards optional,
    MimeVersion         mimeVersion optional,
    ProxyAuthorization  proxyAuthorization optional,
    ProxyRequire        proxyRequire optional,
    RecordRoute         recordRoute optional,
    Require             require optional,
    Route               route optional,
    Timestamp           timestamp optional,
    To                  toField,
    UserAgent           userAgent optional,
    Via                 via
}
```

Header fields are sub structured as records and consist of at least two fields, the fieldName and the fieldValue.

EXAMPLE 2: General structure of a header field type

```
type record HeaderField {
    FieldName   fieldName,
    FieldValue  fieldValue
}
```

The field name type is an enumeration type that contains an entry for each header field type. All enumerations are capitalized and suffixed with "_E" to distinguish enumeration values from constants.

EXAMPLE 3:

```
type enumerated FieldName {
    TO_E, FROM_E
}
```

The field value type can be:

a) a simple type, like **charstring**

EXAMPLE 4:

```
type record Priority {
    FieldName   fieldName,
    charstring  priorityValue
}
```

b)sub structured, or

EXAMPLE 5:

```
type record FieldValue {
    charstring  mediaRanage,
    charstring  acceptParam optional
}
```

c)consists of several parts

EXAMPLE 6:

```
type record MimeVersion {
    FieldName   fieldName,
    integer     majorNumber,
    integer     minorNumber
}
```

The level of substructure is header field depended.

In case a SIP header field might contain several types of information **union**s have been used.

EXAMPLE 7: sub structured union field value type

```
type union ContactBody
{
charstring      wildcard,
ContactAddress_List contactAddress_List
}
```

The contact body can contain either a wildcard or a contact address list.

A list, i.e. the **set of** construct has been used for collecting information of the same type. In case the order is important the **record of** construct has been used. Lists have been created for header fields that may appear multiple times and for parameters of header fields.

EXAMPLE 8: Collection of multiple header fields of the same type

```
type record Via {
    FieldName       fieldName,
    ViaBody_List    viaBody
}

type record of ViaBody ViaBody_List;
```

`ViaBody_List` contains all occurrences of the `Via` field, either explicit as separate header field or implicit as via parameter.

EXAMPLE 9: Collection of multiple parameters of the same type for one header field

```
type set of LanguageBody LanguageBody_List;
```

6.2.2 Constant definitions

6.2.2.1 Constants

Syntactic delimiters like "<", or "@" have been defined as constants.

EXAMPLE 1: `const charstring AT = "@";`

To represent non-printable characters like horizontal tabulator or a carriage return in2char() function is used.

EXAMPLE 2: `const` char LF: = int2char(10); `// the ASCII character 0x0a`

6.2.2.2 External Constants

No external constants were used in the ATS.

6.2.3 Module Parameters

PICS/PIXIT parameters are defined as module parameters.

6.2.4 Template definitions

Modifications of templates have been used to allow focusing on the essential parts of a test message. However these mechanisms have been used very carefully as they can be counter intuitive.

6.2.5 Dynamic part

No conventions for the dynamic part have been defined yet.

7 PCTR conformance

A test laboratory, when requested by a client to produce a PCTR, is required, as specified in ISO/IEC 9646-5 [5], to produce a PCTR conformant with the PCTR template given in annex B of ISO/IEC 9646-5 [5].

Furthermore, a test laboratory, offering testing for the ATS specification contained in annex C, when requested by a client to produce a PCTR, is required to produce a PCTR conformant with the PCTR proforma contained in annex A.

A PCTR which conforms to this PCTR proforma specification shall preserve the content and ordering of the clauses contained in annex A. Clause A.6 of the PCTR may contain additional columns. If included, these shall be placed to the right of the existing columns. Text in italics may be retained by the test laboratory.

8 PIXIT conformance

A test realizer, producing an executable test suite for the Abstract Test Suite (ATS) specification contained in annex C, is required, as specified in ISO/IEC 9646-4 [4], to produce an augmented partial PIXIT proforma conformant with this partial PIXIT proforma specification.

An augmented partial PIXIT proforma which conforms to this partial PIXIT proforma specification shall, as a minimum, have contents which are technically equivalent to annex B. The augmented partial PIXIT proforma may contain additional questions that need to be answered in order to prepare the Means Of Testing (MOT) for a particular Implementation Under Test (IUT).

A test laboratory, offering testing for the ATS specification contained in annex C, is required, as specified in ISO/IEC 9646-5 [5], to further augment the augmented partial PIXIT proforma to produce a PIXIT proforma conformant with this partial PIXIT proforma specification.

A PIXIT proforma which conforms to this partial PIXIT proforma specification shall, as a minimum, have contents which are technically equivalent to annex B. The PIXIT proforma may contain additional questions that need to be answered in order to prepare the test laboratory for a particular IUT.

9 ATS Conformance

The test realizer, producing a Means Of Testing (MOT) and Executable Test Suite (ExTS) for this Abstract Test Suite (ATS) specification, shall comply with the requirements of ISO/IEC 9646-4 [4]. In particular, these concern the realization of an Executable Test Suite (ExTS) based on each ATS. The test realizer shall provide a statement of conformance of the MOT to this ATS specification.

An ExTS which conforms to this ATS specification shall contain test groups and test cases which are technically equivalent to those contained in the ATS in annex C. All sequences of test events comprising an abstract test case shall be capable of being realized in the executable test case. Any further checking which the test system might be capable of performing is outside the scope of this ATS specification and shall not contribute to the verdict assignment for each test case.

Test laboratories running conformance test services using this ATS shall comply with ISO/IEC 9646-5 [5].

A test laboratory which claims to conform to this ATS specification shall use an MOT which conforms to this ATS.

Annex A (normative): Abstract Test Suite (ATS)

This ATS has been produced using the Testing and Test Control Notation version 3 (TTCN-3) according to ES 201 873-2 [3].

The TTCN.3MP representation corresponding to this ATS is contained in an ASCII file (SIP_Module.3MP, SIP_Templates.3MP and SIP_TypesAndConf.3MP contained in archive ts_10202702v010101p0.ZIP) which accompanies the present document.

Annex B (normative): Partial PIXIT proforma

The PIXIT proforma is based on ISO/IEC 9646-6. Any needed additional information can be found in this international standard document.

B.1 Identification summary

Table B.1

PIXIT Number:	
Test Laboratory Name:	
Date of Issue:	
Issued to:	

B.2 ATS summary

Table B.2

Protocol Specification:	
Protocol to be tested:	
ATS Specification:	
Abstract Test Method:	

B.3 Test laboratory

Table B.3

Test Laboratory Identification:	
Test Laboratory Manager:	
Means of Testing:	
SAP Address:	

B.4 Client identification

Table B.4

Client Identification:	
Client Test manager:	
Test Facilities required:	

B.5 SUT

Table B.5

Name:	
Version:	
SCS Number:	
Machine configuration:	
Operating System Identification:	
IUT Identification:	
PICS Reference for IUT:	
Limitations of the SUT:	
Environmental Conditions:	

B.6 Protocol layer information

B.6.1 Protocol identification

Table B.6

Name:	
Version:	
PICS References:	

B.6.2 IUT information

Table B.7: IP parameters

Name	Type	Comments	Value
PX_IUT_PORT	integer	default port number used by the IUT to exchange SIP messages	
PX_IUT_IPADDR	charstring	default IP address used by the ETS to exchange SIP messages	
PX_IUT_PORT2	integer	default port number used by the IUT to exchange SIP messages on PTC side	
PX_IUT_IPADDR2	charstring	default IP address used by the IUT to exchange SIP messages on PTC side	
PX_ETS_PORT	integer	default port number used by the ETS to exchange SIP messages on MTC side	
PX_ETS_IPADDR	charstring	default IP address used by the IUT to exchange SIP messages on MTC side	
PX_ETS_PORT2	integer	default port number used by the ETS to exchange SIP messages on PTC side	
PX_ETS_IPADDR2	charstring	default IP address used by the IUT to exchange SIP messages on PTC side	
PX_PROXY_PORT	integer	default port number used in 305 "Use Proxy" message	
PX_PROXY_IPADDR	charstring	default IP address used in 305 "Use Proxy" message	

Table B.8: Registration parameters

Name	Type	Comments	Value
PX_DELTA_REGISTRATION	charstring	delta-seconds used in expires header field in 200 OK message to answer REGISTRATION request	
PX_MCAST_REGISTRATION	boolean	does IUT address its registration to the well-known "all SIP servers" multicast address?	
PX_ETS_LOCAL_DOMAIN	charstring	identity of the tester local domain on MTC side	
PX_ETS_LOCAL_USER	charstring	identity of the tester local user on MTC side	
PX_ETS_LOCAL_DOMAIN2	charstring	identity of the tester local domain on MTC or PTC side	
PX_ETS_LOCAL_USER2	charstring	identity of the tester local user on PTC side	
PX_HOME_REGISTRATION	boolean	IUT needs to register itself to its home registrar first	

Table B.9: Roaming parameters

Name	Type	Comments	Value
PX_VISITED_DOMAIN	charstring	identity of the visited domain by the IUT	
PX_ETS_VISITING_DOMAIN	charstring	identity of the tester when ETS behaves as a visitor	
PX_ETS_VISITING_USER	charstring	identity of the user tester when ETS behaves as a visitor	
PX_ETS_LOCAL_THIRD_USER	charstring	identity of another tester local user (third party tester)	

Table B.10: Session parameters

Name	Type	Comments	Value
PX_OE_CALLEE_DOMAIN	charstring	hostname of the callee when IUT is the caller	
PX_OE_CALLEE_USERINFO	charstring	userinfo of the callee when IUT is the caller	
PX_TE_CALLEE_DOMAIN	charstring	hostname of the callee when IUT is the callee	
PX_TE_CALLEE_USERINFO	charstring	userinfo of the callee when IUT is the callee	
PX_UNKNOWN_DOMAIN	charstring	unknown hostname when IUT is the callee	
PX_UNKNOWN_USERINFO	charstring	unknown userinfo when IUT is the callee	

Table B.11: Header parameters

Name	Type	Comments	Value
PX_AUTHSHEME	charstring	authorization scheme understood by the IUT	
PX_REALM	charstring	realm value understood by the IUT	
PX_CONTENCOD_UNSUPPORTED	charstring	content encoding mechanism that is not supported by the IUT	
PX_OPTION_UNSUPPORTED	charstring	option set in Request header field that is not supported	

Table B.12: Body parameters

Name	Type	Comments	Value
PX_ PX_SDPBODY	charstring	SDP parameter proposed by the ETS	
PX_SDPBODY2	charstring	additional SDP parameter proposed by the ETS	
PX_SDPBODY_UNSUPPORTED	charstring	SDP parameter proposed by the ETS that is not supported by the IUT	
PX_SDPBODY_65535	charstring	SDP parameter proposed by the ETS that makes the 200 OK or INVITE request of 63.535 bytes long	
PX_SDPBODY_TOO_LARGE	charstring	SDP parameter proposed by the that makes the 200 OK or INVITE request too large	

Table B.13: Timers

Name	Type	Comments	Value
PX_TWAIT	float	TWait default value for waiting an operator action	
PX_TACK	float	default value for waiting an acknowledgement	
PX_TRESP	float	TResp default value for waiting for a response from the IUT	
PX_TINIT	float	default value to init lower stack protocol	
PX_TNOACT	float	value for waiting no message from the IUT	
PX_TSYNC	float	default value to synchronize ptc	
PX_TGUARD	float	default value for an extra long timer to limit test execution	

Annex C (normative): PCTR proforma

The PCTR proforma is based on ISO/IEC 9646-6. Any needed additional information can be found in the present document.

C.1 Identification summary

C.1.1 Protocol conformance test report

Table C.1

PCTR Number:	
PCTR Date:	
Corresponding SCTR Number:	
Corresponding SCTR Date:	
Test Laboratory Identification:	
Test Laboratory Manager:	
Signature:	

C.1.2 IUT identification

Table C.2

Name:	
Version:	
Protocol specification:	
PICS:	
Previous PCTR if any:	

C.1.3 Testing environment

Table C.3

PIXIT Number:	
ATS Specification:	
Abstract Test Method:	
Means of Testing identification:	
Date of testing:	
Conformance Log reference(s):	
Retention Date for Log reference(s):	

C.1.4 Limits and reservation

Additional information relevant to the technical contents or further use of the test report, or the rights and obligations of the test laboratory and the client, may be given here. Such information may include restriction on the publication of the report.

.....

.....

.....

.....

.....

C.1.5 Comments

Additional comments may be given by either the client or the test laboratory on any of the contents of the PCTR, for example, to note disagreement between the two parties.

.....

.....

.....

.....

.....

C.2 IUT Conformance status

This IUT has or has not been shown by conformance assessment to be non conforming to the specified protocol specification.

Strike the appropriate words in this sentence. If the PICS for this IUT is consistent with the static conformance requirements (as specified in clause D.3 in the present document) and there are no "FAIL" verdicts to be recorded (in clause D.6 in the present document) strike the words "has or", otherwise strike the words "or has not".

C.3 Static conformance summary

The PICS for this IUT is or is not consistent with the static conformance requirements in the specified protocol.

Strike the appropriate words in this sentence.

C.4 Dynamic conformance summary

The test campaign did or did not reveal errors in the IUT.

Strike the appropriate words in this sentence. If there are no "FAIL" verdicts to be recorded (in clause D.6 of the present document) strike the words "did or" otherwise strike the words "or did not".

Summary of the results of groups of test:

.....

.....

.....

.....

.....

.....

.....

.....

.....

C.5 Static conformance review report

If clause D.3 indicates non-conformance, this clause itemizes the mismatches between the PICS and the static conformance requirements of the specified protocol specification.

.....

.....

.....

.....

.....

.....

.....

.....

.....

C.6 Test campaign report

Table C.6

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_RG_RT_V_001				
SIP_RG_RT_V_002				
SIP_RG_RT_V_003				
SIP_RG_RT_V_004				
SIP_RG_RT_V_005				
SIP_RG_RT_V_006				
SIP_RG_RT_V_007				
SIP_RG_RT_V_008				
SIP_RG_RT_V_009				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_RG_RR_V_001				
SIP_RG_RR_V_002				
SIP_RG_RR_V_003				
SIP_RG_RR_V_004				
SIP_RG_RR_V_005				
SIP_RG_RR_V_006				
SIP_RG_RR_V_007				
SIP_RG_RR_V_008				
SIP_RG_RR_V_009				
SIP_RG_RR_V_010				
SIP_RG_RR_V_011				
SIP_RG_RR_V_012				
SIP_RG_RR_V_013				
SIP_RG_RR_V_014				
SIP_RG_RR_V_015				
SIP_RG_RR_V_016				
SIP_RG_RR_I_001				
SIP_RG_RR_I_002				
SIP_RG_RR_I_003				
SIP_RG_RR_O_001				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_SS_OE_CE_V_001				
SIP_SS_OE_CE_V_002				
SIP_SS_OE_CE_V_003				
SIP_SS_OE_CE_V_004				
SIP_SS_OE_CE_V_005				
SIP_SS_OE_CE_V_006				
SIP_SS_OE_CE_V_007				
SIP_SS_OE_CE_V_008				
SIP_SS_OE_CE_V_009				
SIP_SS_OE_CE_V_010				
SIP_SS_OE_CE_V_011				
SIP_SS_OE_CE_V_012				
SIP_SS_OE_CE_V_013				
SIP_SS_OE_CE_V_014				
SIP_SS_OE_CE_V_015				
SIP_SS_OE_CE_V_016				
SIP_SS_OE_CE_V_017				
SIP_SS_OE_CE_V_018				
SIP_SS_OE_CE_V_019				
SIP_SS_OE_CE_V_020				
SIP_SS_OE_CE_V_021				
SIP_SS_OE_CE_V_022				
SIP_SS_OE_CE_V_023				
SIP_SS_OE_CE_V_024				
SIP_SS_OE_CE_V_025				
SIP_SS_OE_CE_V_026				
SIP_SS_OE_CE_V_027				
SIP_SS_OE_CE_V_028				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_SS_OE_CR_V_001				
SIP_SS_OE_CR_V_002				
SIP_SS_OE_CR_V_003				
SIP_SS_OE_CR_V_004				
SIP_SS_OE_CR_V_005				
SIP_SS_OE_CR_V_006				
SIP_SS_OE_CR_I_001				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_SS_TE_CE_V_001				
SIP_SS_TE_CE_V_002				
SIP_SS_TE_CE_V_003				
SIP_SS_TE_CE_V_004				
SIP_SS_TE_CE_V_005				
SIP_SS_TE_CE_V_006				
SIP_SS_TE_CE_V_007				
SIP_SS_TE_CE_V_008				
SIP_SS_TE_CE_V_009				
SIP_SS_TE_CE_V_010				
SIP_SS_TE_CE_V_011				
SIP_SS_TE_CE_V_012				
SIP_SS_TE_CE_V_013				
SIP_SS_TE_CE_V_014				
SIP_SS_TE_CE_V_015				
SIP_SS_TE_CE_V_016				
SIP_SS_TE_CE_I_001				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_SS_TE_CR_V_001				
SIP_SS_TE_CR_V_002				
SIP_SS_TE_CR_V_003				
SIP_SS_TE_CR_I_001				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_SS_TE_SM_V_001				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_SS_PR_CE_V_001				
SIP_SS_PR_CE_V_002				
SIP_SS_PR_CE_V_003				
SIP_SS_PR_CE_V_004				
SIP_SS_PR_CE_V_005				
SIP_SS_PR_CE_V_006				
SIP_SS_PR_CE_V_007				
SIP_SS_PR_CE_V_008				
SIP_SS_PR_CE_V_009				
SIP_SS_PR_CE_V_010				
SIP_SS_PR_CE_V_011				
SIP_SS_PR_CE_V_012				
SIP_SS_PR_CE_V_013				
SIP_SS_PR_CE_V_014				
SIP_SS_PR_CE_V_015				
SIP_SS_PR_CE_V_016				
SIP_SS_PR_CE_V_017				
SIP_SS_PR_CE_V_018				
SIP_SS_PR_CE_V_019				
SIP_SS_PR_CE_V_020				
SIP_SS_PR_CE_V_021				
SIP_SS_PR_CE_V_022				
SIP_SS_PR_CE_V_023				
SIP_SS_PR_CE_V_024				
SIP_SS_PR_CE_V_025				
SIP_SS_PR_CE_V_026				
SIP_SS_PR_CE_V_027				
SIP_SS_PR_CE_V_028				
SIP_SS_PR_CE_V_029				
SIP_SS_PR_CE_V_030				
SIP_SS_PR_CE_V_031				
SIP_SS_PR_CE_V_032				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_SS_PR_CR_V_001				
SIP_SS_PR_CR_V_002				
SIP_SS_PR_CR_V_003				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_SS_PR_SM_V_001				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_SS_RD_CE_V_001				
SIP_SS_RD_CE_V_002				
SIP_SS_RD_CE_V_003				
SIP_SS_RD_CE_V_004				
SIP_SS_RD_CE_V_005				
SIP_SS_RD_CE_V_006				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_SS_RD_CR_V_001				
SIP_SS_RD_CR_V_002				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_MG_RT_V_001				
SIP_MG_RT_V_002				
SIP_MG_RT_V_003				
SIP_MG_RT_V_004				
SIP_MG_RT_V_005				
SIP_MG_RT_V_006				
SIP_MG_RT_V_007				
SIP_MG_RT_V_008				
SIP_MG_RT_V_009				
SIP_MG_RT_V_010				
SIP_MG_RT_V_011				
SIP_MG_RT_V_012				
SIP_MG_RT_V_013				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_MG_RT_I_001				
SIP_MG_RT_I_002				
SIP_MG_RT_I_003				
SIP_MG_RT_I_004				
SIP_MG_RT_I_005				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_MG_RR_V_001				
SIP_MG_RR_V_002				
SIP_MG_RR_V_003				
SIP_MG_RR_V_004				
SIP_MG_RR_V_005				
SIP_MG_RR_V_006				
SIP_MG_RR_V_007				
SIP_MG_RR_V_008				
SIP_MG_RR_V_009				
SIP_MG_RR_V_010				
SIP_MG_RR_V_011				
SIP_MG_RR_V_012				
SIP_MG_RR_V_013				
SIP_MG_RR_V_014				
SIP_MG_RR_V_015				
SIP_MG_RR_V_016				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_MG_RR_V_001				
SIP_MG_RR_V_017				
SIP_MG_RR_V_018				
SIP_MG_RR_V_019				
SIP_MG_RR_V_020				
SIP_MG_RR_V_021				
SIP_MG_RR_V_022				
SIP_MG_RR_V_023				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_MG_RR_I_001				
SIP_MG_RR_I_002				
SIP_MG_RR_I_003				
SIP_MG_RR_I_004				
SIP_MG_RR_I_005				
SIP_MG_RR_I_006				
SIP_MG_RR_I_007				
SIP_MG_RR_I_008				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_MG_OE_V_001				
SIP_MG_OE_V_002				
SIP_MG_OE_V_003				
SIP_MG_OE_V_004				
SIP_MG_OE_V_005				
SIP_MG_OE_V_006				
SIP_MG_OE_V_007				
SIP_MG_OE_V_008				
SIP_MG_OE_V_009				
SIP_MG_OE_V_010				
SIP_MG_OE_V_011				
SIP_MG_OE_V_012				
SIP_MG_OE_V_013				
SIP_MG_OE_V_014				
SIP_MG_OE_V_015				
SIP_MG_OE_V_016				
SIP_MG_OE_V_017				
SIP_MG_OE_V_018				
SIP_MG_OE_V_019				
SIP_MG_OE_V_020				
SIP_MG_OE_V_021				
SIP_MG_OE_V_022				
SIP_MG_OE_V_023				
SIP_MG_OE_V_024				
SIP_MG_OE_V_025				
SIP_MG_OE_V_026				
SIP_MG_OE_V_027				
SIP_MG_OE_V_028				
SIP_MG_OE_V_029				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_MG_OE_I_001				
SIP_MG_OE_I_002				
SIP_MG_OE_I_003				
SIP_MG_OE_I_004				
SIP_MG_OE_I_005				
SIP_MG_OE_I_006				
SIP_MG_OE_I_007				
SIP_MG_OE_I_008				
SIP_MG_OE_I_009				
SIP_MG_OE_I_010				
SIP_MG_OE_I_011				
SIP_MG_OE_I_012				
SIP_MG_OE_I_013				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_MG_TE_V_001				
SIP_MG_TE_V_002				
SIP_MG_TE_V_003				
SIP_MG_TE_V_004				
SIP_MG_TE_V_005				
SIP_MG_TE_V_006				
SIP_MG_TE_V_007				
SIP_MG_TE_V_008				
SIP_MG_TE_V_009				
SIP_MG_TE_V_010				
SIP_MG_TE_V_011				
SIP_MG_TE_V_012				
SIP_MG_TE_V_013				
SIP_MG_TE_V_014				
SIP_MG_TE_V_015				
SIP_MG_TE_V_016				
SIP_MG_TE_V_017				
SIP_MG_TE_V_018				
SIP_MG_TE_V_019				
SIP_MG_TE_V_020				
SIP_MG_TE_V_021				
SIP_MG_TE_V_022				
SIP_MG_TE_V_023				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_MG_TE_I_001				
SIP_MG_TE_I_002				
SIP_MG_TE_I_003				
SIP_MG_TE_I_004				
SIP_MG_TE_I_005				
SIP_MG_TE_I_006				
SIP_MG_TE_I_007				
SIP_MG_TE_I_008				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_MG_PR_V_001				
SIP_MG_PR_V_002				
SIP_MG_PR_V_003				
SIP_MG_PR_V_004				
SIP_MG_PR_V_005				
SIP_MG_PR_V_006				
SIP_MG_PR_V_007				
SIP_MG_PR_V_008				
SIP_MG_PR_V_009				
SIP_MG_PR_V_010				
SIP_MG_PR_V_011				
SIP_MG_PR_V_012				
SIP_MG_PR_V_013				
SIP_MG_PR_V_014				
SIP_MG_PR_V_015				
SIP_MG_PR_V_016				
SIP_MG_PR_V_017				
SIP_MG_PR_V_018				
SIP_MG_PR_V_019				
SIP_MG_PR_V_020				
SIP_MG_PR_V_021				
SIP_MG_PR_V_022				
SIP_MG_PR_V_023				
SIP_MG_PR_V_024				
SIP_MG_PR_V_025				
SIP_MG_PR_V_026				
SIP_MG_PR_V_027				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_MG_PR_I_001				
SIP_MG_PR_I_002				
SIP_MG_PR_I_003				
SIP_MG_PR_I_004				
SIP_MG_PR_I_005				
SIP_MG_PR_I_006				
SIP_MG_PR_I_007				
SIP_MG_PR_I_008				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_MG_RD_V_001				
SIP_MG_RD_V_002				
SIP_MG_RD_V_003				
SIP_MG_RD_V_004				
SIP_MG_RD_V_005				
SIP_MG_RD_V_006				
SIP_MG_RD_V_007				
SIP_MG_RD_V_008				
SIP_MG_RD_V_009				
SIP_MG_RD_V_010				
SIP_MG_RD_V_011				
SIP_MG_RD_V_012				
SIP_MG_RD_V_013				
SIP_MG_RD_V_014				
SIP_MG_RD_V_015				
SIP_MG_RD_V_016				
SIP_MG_RD_V_017				
SIP_MG_RD_V_018				
SIP_MG_RD_V_019				
SIP_MG_RD_V_020				
SIP_MG_RD_V_021				
SIP_MG_RD_V_022				

ATS Reference	Selected?	Run?	Verdict	Observations (Reference to any observations made in clause 8 of this annex)
SIP_MG_RD_I_001				
SIP_MG_RD_I_002				
SIP_MG_RD_I_003				
SIP_MG_RD_I_004				
SIP_MG_RD_I_005				
SIP_MG_RD_I_006				
SIP_MG_RD_I_007				
SIP_MG_RD_I_008				

C.8 Observations

Additional information relevant to the technical content of the PCTR is given here.

.....

.....

.....

.....

.....

.....

.....

.....

.....

Annex D (informative): Bibliography

- ISO/IEC 9646-1 (1994): "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 1: General concepts".
- ISO/IEC 9646-2 (1994): "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 2: Abstract Test Suite specification".
- ISO/IEC 9646-3 (1998): "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 3: The Tree and Tabular Combined Notation (TTCN)".
- ISO/IEC 9646-7 (1995): "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 7: Implementation Conformance Statements".

History

Document history		
V1.1.1	September 2002	Publication